



System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

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

Part No: 816-7511-10
December 2002

Copyright 2002 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

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

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, Sun ONE and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

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

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions.

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

Copyright 2002 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

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

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

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

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



030326@5533



Contents

Preface 17

Part I About Naming and Directory Services

- 1 Naming and Directory Services (Overview) 23**
 - What Is a Naming Service? 23
 - Solaris Naming Services 29
 - DNS 30
 - /etc Files 30
 - NIS 30
 - NIS+ 31
 - FNS 31
 - LDAP Naming Services 32
 - Naming Services: A Quick Comparison 32

- 2 The Name Service Switch (Overview) 33**
 - About the Name Service Switch 33
 - Format of the `nsswitch.conf` File 34
 - Comments in `nsswitch.conf` Files 38
 - Keyserver and `publickey` Entry in the Switch File 38
 - The `nsswitch.conf` Template Files 38
 - The Default Switch Template Files 39
 - The `nsswitch.conf` File 42
 - Selecting a Different Configuration File 43
 - ▼ Modifying the name service switch 43

DNS and Internet Access	44
IPv6 and Solaris Naming Services	44
Ensuring Compatibility With +/- Syntax	45
The Switch File and Password Information	46

Part II DNS Setup and Administration

3 Domain Name System (Overview)	49
DNS Basics	49
Name-to-Address Resolution	50
DNS Administrative Domains	52
<i>in.named</i> and DNS Name Servers	53
Server Configuration and Data File Names	53
Configuration File	53
Names of DNS Data Files	54
Domain Names	56
Default Domain Name	56
Trailing Dots in Domain Names	56
DNS Clients and the Resolver	56
The <i>resolv.conf</i> File	57
The <i>named.conf</i> File	58
DNS Hierarchy in a Local Domain	60
DNS Hierarchy and the Internet	61
Zones	64
Reverse Mapping	65
4 Administering DNS (Tasks)	67
Setting Up the <i>resolv.conf</i> File	67
Configuring a Network For DNS	68
Setting Up a DNS Client	68
Setting Up a DNS Server	70
How to Specify a Master Server	71
How to Specify a Slave Server	72
How to Specify a Cache-Only or Stub Server	73
How to Add DNS Compatibility and +/- Syntax	74
Setting up DNS Servers	75
Initializing the Server	75

Testing Your Installation	75
Adding Additional Servers	77
Modifying DNS Data Files	77
How to Change the SOA Serial Number	78
Forcing <code>in.named</code> to Reload DNS Data	78
Adding and Deleting Clients	79
Adding a Client	79
Removing a Client	80
Enabling a Client to Use IPv6	80
▼ How to Enable a Client to Use IPv6	80
Creating DNS Subdomains	81
Planning Your Subdomains	81
Setting Up a Subdomain	83
Solaris DNS BIND 8.2.4 Implementation	84
▼ How to Migrate from BIND 4.9.x to BIND 8.2.4	85
DNS Forwarding	85
▼ How to Enable DNS Forwarding Capabilities on an NIS+ Client	85
▼ How to enable DNS forwarding capabilities on an older NIS client	85
5 DNS Administration (Reference)	87
Implementing DNS	87
A Practical Example	87
Setting Up the Data Files	93
Resource Record Types	93
Setting Up Subdomains	94
Setting Up Subdomains—Same Zone	94
Setting Up Subdomains—Different Zones	95
The DNS Namespace Hierarchy	96
Domains and Subdomains	96
How DNS Affects Mail Delivery	97
DNS Configuration and Data Files	97
Names of DNS Data Files	97
The <code>named.conf</code> File	98
The <code>named.ca</code> File	101
The <code>hosts</code> File	103
The <code>hosts.rev</code> File	104
The <code>named.local</code> File	105

\$INCLUDE Files	106
Data File Resource Record Format	106
Standard Resource Record Format	107
Special Resource Record Characters	108
Control Entries	109
Resource Record Types	109
6 DNS Troubleshooting (Reference)	117
Clients Can Find Machine by Name but Server Cannot	117
Changes Do Not Take Effect or Are Erratic	118
DNS Client Cannot Lookup “Short” Names	119
Reverse Domain Data Not Correctly Transferred to slave	119
Server Failed and Zone Expired Problems	119
rlogin, rsh, and ftp Problems	120
Other DNS Syntax Errors	121

Part III NIS Setup and Administration

7 Network Information Service (NIS) (Overview)	125
NIS Introduction	125
NIS Architecture	126
NIS Machine Types	127
NIS Servers	127
NIS Clients	127
NIS Elements	128
The NIS Domain	128
NIS Daemons	128
NIS Utilities	129
NIS Maps	129
NIS-Related Commands	133
NIS Binding	134
Server-List Mode	135
Broadcast Mode	135
Differences in NIS Solaris 2.6 NIS and Earlier NIS Versions	136
NSKit Discontinued	136
The ypupdated Daemon	136
/var/yp/securenets	136

	Multihomed Machine Support	137
	SunOS 4 Compatibility Mode	137
8	Setting Up and Configuring NIS Service	139
	Configuring NIS — Task Map	139
	Before You Begin Configuring NIS	140
	Planning Your NIS Domain	140
	Identify Your NIS Servers and Clients	141
	Preparing the Master Server	141
	Source Files Directory	141
	Passwd Files and Namespace Security	141
	Preparing Source Files for Conversion to NIS Maps	142
	Preparing the Makefile	143
	How to Set Up the Master Server With ypinit	144
	Starting NIS Service on the Master Server	146
	Starting NIS Service Automatically	146
	Starting and Stopping NIS From the Command Line	146
	Setting Up NIS Slave Servers	147
	Preparing a Slave Server	147
	Setting Up a Slave Server	147
	Setting Up NIS Clients	149
	Configuring a Machine to Use NIS	149
9	Administering NIS (Tasks)	151
	Password Files and Namespace Security	151
	Administering NIS Users	152
	Adding a New User to an NIS Domain	152
	Setting User Passwords	153
	Netgroups	154
	Working With NIS Maps	155
	Obtaining Map Information	155
	Changing a Map's Master Server	156
	Modifying Configuration Files	157
	Modifying and Using the Makefile	158
	Updating and Modifying Existing Maps	161
	Updating Maps Supplied with the Default Set	161
	Modifying Default Maps	164

Using makedbm to Modify a Non-Default Map	164
Creating New Maps from Text Files	165
Adding Entries to a File-Based Map	165
Creating Maps From Standard Input	165
Modifying Maps Made From Standard Input	165
Adding a Slave Server	166
▼ How to Add a Slave Server	166
Using NIS With C2 Security	167
Changing a Machine's NIS Domain	168
▼ How to Change a Machine's NIS Domain Name	168
Using NIS in Conjunction With DNS	168
▼ Configuring Machine Name and Address Lookup Through NIS and DNS	168
Dealing with Mixed NIS Domains	169
Turning Off NIS Services	170

10 NIS Troubleshooting	171
NIS Binding Problems	171
Symptoms	171
NIS Problems Affecting One Client	172
NIS Problems Affecting Many Clients	175

Part IV Configuring Sun ONE Directory Server 5.1

11 Configuring Sun ONE Directory Server 5.1 (Tasks)	183
Preparing for Configuration	184
Configuration Components	184
Configuration Choices	185
Choosing Unique Port Numbers	185
Choosing User and Group	186
Defining Authentication Entities	186
Choosing Your Directory Suffix	187
Choosing the Location of the Configuration Directory	188
Choosing the Location of the User Directory	189
Choosing the Administration Domain	189
Configuration Process Overview	190
Selecting an Configuration Process	190
Using Express and Typical Configuration	191

Using Express Configuration	191
Using Typical Configuration	192

Part V LDAP Naming Services Setup and Administration

12 Introduction to LDAP Naming Services (Overview/Reference)	197
Audience Assumptions	197
Suggested Background Reading	198
Additional Prerequisites	198
LDAP Naming Services Compared to Other Naming Services	198
Using Fully Qualified Domain Names	199
Advantages of LDAP Naming Services	200
Restrictions of LDAP Naming Services	200
New LDAP Naming Services Features for Solaris 9	200
Transitioning from NIS+ to LDAP	201
LDAP Naming Services Setup (Task Map)	201
13 Basic Components and Concepts (Overview)	203
LDAP Data Interchange Format (LDIF)	203
Default Directory Information Tree (DIT)	206
Default Schema	207
Service Search Descriptors (SSDs) and Schema Mapping	208
SSDs	208
Client Profiles	210
Client Profile Attributes	210
ldap_cachemgr Daemon	213
LDAP Naming Services Security Model	214
Introduction	214
Transport Layer Security (TLS)	214
Assigning Client Credential Levels	215
Choosing Authentication Methods	217
Pluggable Authentication Methods	219
Password Management	221
14 Planning for the LDAP Naming Service (Tasks)	223
Overview	223

	Planning the Network Model	224
	Planning the Directory Information Tree (DIT)	224
	Multiple Directory Servers	225
	Data Sharing With Other Applications	225
	Choosing the Directory Suffix	226
	Replica Servers	226
	Planning the Security Model	227
	Planning Client Profiles and Default Attribute Values	227
	Planning the Data Population	228
	▼ How to Populate a Server With host Entries Using ldapaddent	229
15	Setting Up Sun ONE Directory Server 5.1 (Tasks)	231
	Configuring Sun ONE Directory Server 5.1 Using idsconfig	232
	Creating a Checklist Based on Your Server Installation	232
	Schema Definitions	234
	Using Browsing Indices	234
	Using Service Search Descriptors to Modify Client Access to Various Services	235
	Setting Up SSDs Using idsconfig	235
	Running idsconfig	236
	▼ How to Configure Sun ONE Directory Server Using idsconfig	237
	Populating the Directory Server Using ldapaddent	240
	▼ How to Populate Sun ONE Directory Server 5.1 With User Password Data Using ldapaddent	241
	Managing Printer Entries	241
	Adding Printers	241
	Using lpget	241
	Populating the Directory Server With Additional Profiles	242
	▼ How to Populate the Directory Server With Additional Profiles Using ldapclient	242
	Configuring the Directory Server to Enable Password Management	242
16	Setting Up Clients (Task)	245
	Prerequisites	245
	Initializing a Client	246
	Using Profiles to Initialize a Client	246
	Using Proxy Credentials	247
	Initializing a Client Manually	247

	Modifying a Manual Client Configuration	248
	Uninitializing a Client	248
	Setting Up TLS Security	249
	Configuring PAM	250
	Retrieving LDAP Naming Services Information	251
	Using <code>ldaplist</code>	251
	Customizing the Client Environment	252
	Modifying the <code>nsswitch.conf</code> File	253
17	LDAP Troubleshooting (Reference)	255
	Monitoring Client Status	255
	Verifying <code>ldap_cachemgr</code> is running	255
	Checking the Current Profile Information	256
	Verifying Basic Client/Server Communication	256
	Checking Server Data From a Non-client Machine	257
	Configuration Problems and Solutions	257
	Unresolved Hostname	257
	Unable to Reach Systems in the LDAP Domain Remotely	257
	Login Does Not Work	257
	Lookup Too Slow	258
	<code>ldapclient</code> Cannot Bind to Server	258
	Using <code>ldap_cachemgr</code> for Debugging	259
	<code>ldapclient</code> Hangs During Setup	259
	Frequently Asked Questions	259
18	LDAP General Reference (Reference)	261
	Blank Checklists	261
	Upgrade Information	262
	New <code>automount</code> Schema	263
	LDAP Commands	263
	General LDAP Tools	263
	LDAP Tools Requiring LDAP Naming Services	264
	Example <code>pam.conf</code> File for <code>pam_ldap</code>	264
	Example <code>pam_conf</code> file for <code>pam_ldap</code> Configured for Password Management	266
	IETF Schemas	267
	RFC 2307 Network Information Service Schema	268
	Mail Alias Schema	273

Directory User Agent Profile (DUAProfile) Schema	274
Solaris Schemas	276
Solaris Projects Schema	276
Role-Based Access Control and Execution Profile Schema	276
Internet Print Protocol Information	278
Internet Print Protocol (IPP) Attributes	278
Internet Print Protocol (IPP) ObjectClasses	284
Sun Printer Attributes	285
Sun Printer ObjectClasses	286
Generic Directory Server Requirements	286
Default Filters Used by LDAP Naming Services	287
19 Transitioning From NIS+ to LDAP	291
Overview	291
Configuration Files	292
Creating Attributes and Object Classes	293
Getting Started	293
/etc/default/rpc.nisd	293
/var/nis/NIS+LDAPmapping	297
NIS+ to LDAP Migration Scenarios	302
Merging NIS+ and LDAP Data	303
Masters and Replicas	306
Replication Timestamps	306
The Directory Server	307
Configuring the Sun ONE Directory Server 5.1	307
Assigning Server Address and Port Number	308
Security and Authentication	308
Performance and Indexing	310
Mapping NIS+ Objects Other Than Table Entries	311
NIS+ Entry Owner, Group, Access, and TTL	313
▼ How to Store Additional Entry Attributes in LDAP	313
Principal Names and Netnames	316
client_info and timezone Tables	318
client_info Attributes and Object Class	318
timezone Attributes and Object Class	319
Adding New Object Mappings	320
▼ How to Map Non-Entry Objects	320

Adding Entry Objects 322
Storing Configuration Information in LDAP 325

A System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) Updates 331
Solaris 9 12/02 Updates 331

Glossary 333

Index 339

Figures

FIGURE 3-1	Name to Address Resolution	50
FIGURE 3-2	Name to Address Resolution for a Remote Host	51
FIGURE 3-3	Hierarchy of DNS Domains in a Single Organization	60
FIGURE 3-4	Hierarchy of Internet Domains	61
FIGURE 3-5	Ajax Domain's Position in the DNS Namespace	62
FIGURE 3-6	Domains and Zones	64
FIGURE 5-1	Domains and Subdomains	96

Preface

Solaris Administration Guide: Naming and Directory Services (DNS, NIS and LDAP) describes the set up, configuration, and administration of the Solaris 9 operating environment naming and directory services: DNS, NIS, and LDAP. This manual is part of the Solaris™ 9 Release System and Network Administration manual set.

Who Should Use This Book

This manual is written for experienced system and network administrators.

Although this book introduces networking concepts relevant to Solaris™ naming and directory services, it explains neither the networking fundamentals nor the administration tools in the Solaris operating environment.

How This Book Is Organized

This manual is divided into parts according to the respective naming services.

Part I: About Naming and Directory Services

Part II: DNS Setup and Administration

Part III: NIS Setup Administration

Part IV: Sun™ ONE Directory Server 5.1 (formerly iPlanet™ Directory Server 5.1) Configuration

Related Books

- *DNS and Bind*, by Cricket Liu and Paul Albitz, (O' Reilly, 1992)
- *Understanding and Deploying LDAP Directory Services* by Timothy A. Howes, Ph.D and Mark C. Smith

In addition to providing a thorough treatment of LDAP naming services, this book includes useful case studies on deploying LDAP at a large university, a large multinational enterprise, and an enterprise with an extranet.

- *Sun ONE Directory Server 5.1 Deployment Guide* which is included in the Documentation CD.

This guide provides a foundation for planning your directory, including directory design, including schema design, the directory tree, topology, replication, and security. The last chapter provides sample deployment scenarios to help you plan simple deployments as well as complex deployments designed to support millions of users distributed worldwide.

- *Sun ONE Directory Server 5.1 Administrator's Guide*
-

Accessing Sun Documentation Online

The docs.sun.comSM Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is <http://docs.sun.com>.

Typographic Conventions

The following table describes the typographic changes used in this book.

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. machine_name% you have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	machine_name% su Password:
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type rm <i>filename</i> .
<i>AaBbCc123</i>	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in <i>User's Guide</i> . These are called <i>class</i> options. You must be <i>root</i> to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P-2 Shell Prompts

Shell	Prompt
C shell prompt	machine_name%
C shell superuser prompt	machine_name#
Bourne shell and Korn shell prompt	\$
Bourne shell and Korn shell superuser prompt	#

PART I About Naming and Directory Services

This part introduces the naming and directory services for the Solaris Operating Environment. It also describes the `nsswitch.conf` file that you use to coordinate the use of the different services.

Naming and Directory Services (Overview)

This chapter provides an overview of *namespaces* and *naming services* are and what they do. This chapter also briefly describes in brief the Solaris naming services: DNS, NIS, NIS+ and LDAP naming services. See *System Administration Guide: Naming and Directory Services (FNS and NIS+)* for detailed information about NIS+ and FNS.

What Is a Naming Service?

Naming services store information in a central place, which enables users, machines, and applications to communicate across the network. This information can include the following.

- Machine (host) names and addresses
- User names
- Passwords
- Access permissions
- Group membership, printers, and so on

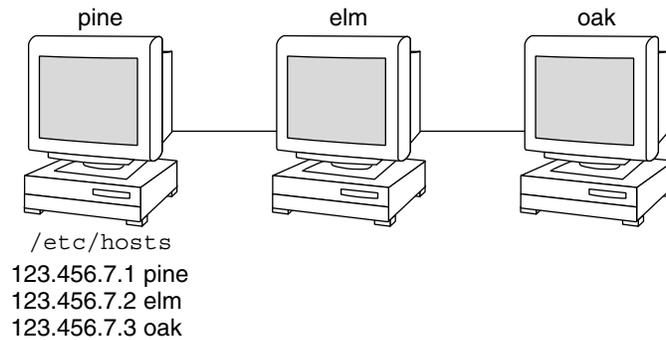
Without a central naming service, each machine would have to maintain its own copy of this information. naming service information can be stored in files, maps, or database tables. Centrally locating this data makes it easier to administer large networks.

Naming services are fundamental to any computing network. Among other features, naming service provide functionality that does the following.

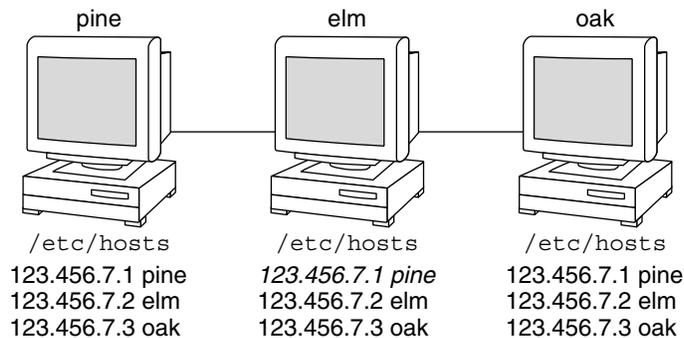
- Associates (*binds*) names with objects
- Resolves names to objects
- Removes bindings
- Lists names
- Renames

A network information service enables machines to be identified by common names instead of numerical addresses. This makes communication simpler because users do not have to remember and try to enter cumbersome numerical addresses like 192.168.00.00.

For example, take a network of three machines named, `pine`, `elm`, and `oak`. Before `pine` can send a message to either `elm` or `oak`, it must know their numerical network addresses. For this reason, it keeps a file, `/etc/hosts` or `/etc/inet/ipnodes`, that stores the network address of every machine in the network, including itself.



Likewise, in order for `elm` and `oak` to communicate with `pine` or with each other, they must keep similar files.

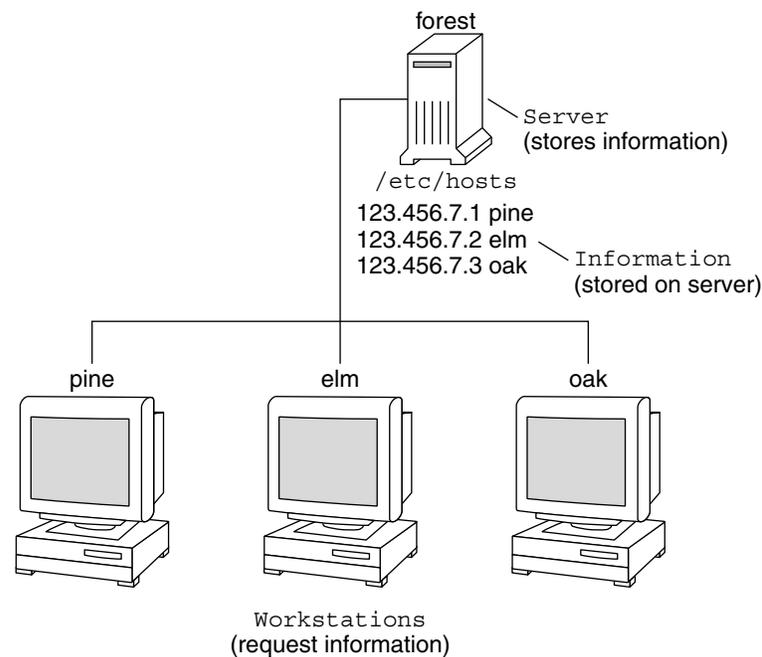


In addition to addresses, machines store security information, mail data, information about their Ethernet interfaces, network services, groups of users allowed to use the network, services offered on the network, and so on. As networks offer more services, the list grows. As a result, each machine might need to keep an entire set of files similar to `/etc/hosts` or `/etc/inet/ipnodes`.

As this information changes, administrators must keep it current on every machine in the network. In a small network, this is tedious. On a medium or large network, the job becomes not only time-consuming but nearly unmanageable.

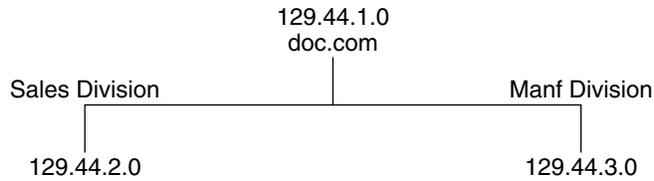
A network information service solves this problem. It stores network information on a server, which provides the information to any machine that queries it.

The machines are known as *clients* of the server. The following figure illustrates the client-server arrangement. Whenever information about the network changes, instead of updating each client's local file, an administrator updates only the information stored by the network information service. This reduces errors, inconsistencies between clients, and the sheer size of the task.

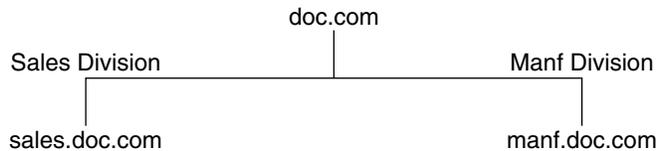


This arrangement, of a server providing centralized services to clients across a network, is known as *client-server computing*.

Although the main purpose of a network information service is to centralize information, another is to simplify network names. For example, assume your company has set up a network and connected it to the Internet. The Internet has assigned your network the network number 192.68.0.0 and the domain name doc.com. Your company has two divisions, Sales and Manufacturing (Manf), so its network is divided into a main net and two subnets, one for each division. Each net has its own address.



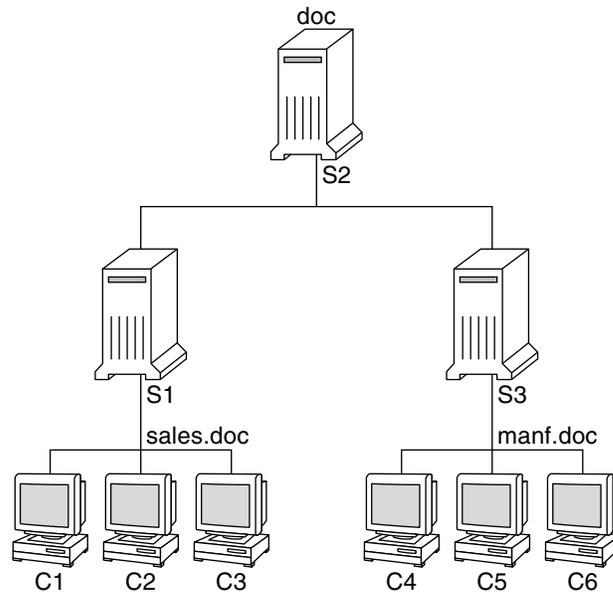
Each division could be identified by its network address, as shown above, but descriptive names made possible by naming services would be preferable.



Instead of addressing mail or other network communications to 129.44.1.0, they could be addressed to doc. Instead of addressing them to 192.68.2.0 or 192.68.3.0, they could be addressed to sales.doc or manf.doc.

Names are also more flexible than physical addresses. Physical networks tend to remain stable, but the organizations that use them tend to change. A network information service can act as a buffer between an organization and its physical network, as it is mapped and not hard-wired to it.

For example, assume that the doc.com network is supported by three servers, S1, S2, and S3, and that two of those servers, S1 and S3, support clients.

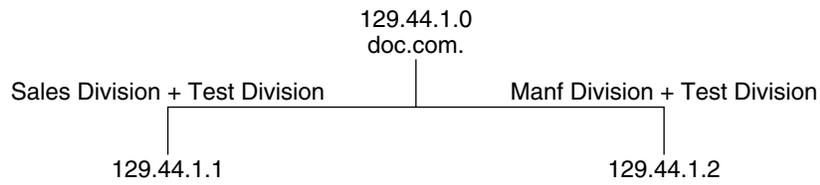


Clients C1, C2, and C3 would obtain their network information from server S1. Clients C4, C5, and C6 would obtain it from server S3. The resulting network is summarized in the following table. (The table is a generalized representation of that network but does not resemble an actual network information map.)

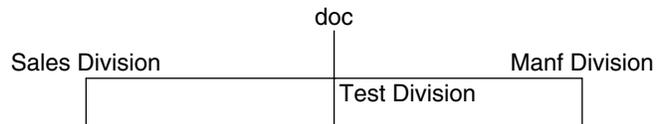
TABLE 1-1 Representation of doc.com Network

Network Address	Network Name	Server	Clients
192.68.1.0	doc	S1	
192.68.2.0	sales.doc	S2	C1, C2, C3
192.68.3.0	manf.doc	S3	C4, C5, C6

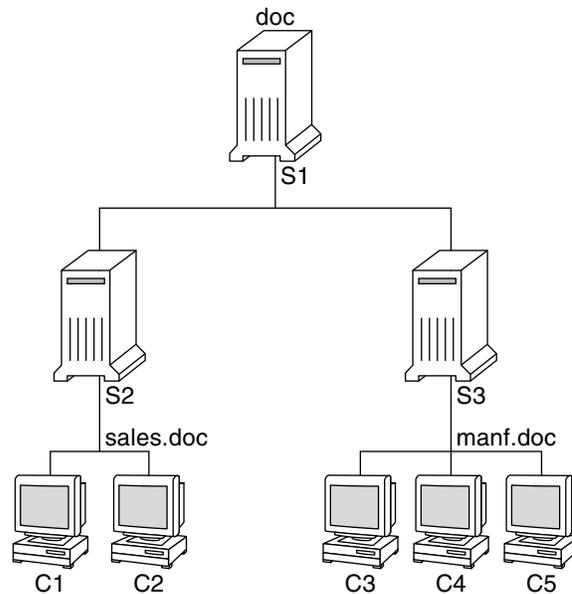
Now assume that you create a third division, Testing, which borrowed some resources from the other two divisions, but did not create a third subnet. The physical network would then no longer parallel the corporate structure.



Traffic for the Test Division would not have its own subnet, but would instead be split between 192 . 68 . 2 . 0 and 192 . 68 . 3 . 0. However, with a network information service, the Test Division traffic could have its own dedicated network.



Thus, when an organization changes, its network information service can change its mapping as shown here.



Now clients C1 and C2 would obtain their information from server S2 and C3, C4 and C5 from server S3.

Subsequent changes in your organization would continue to be accommodated by changes to the “soft” network information structure without reorganizing the “hard” network structure.

Solaris Naming Services

The Solaris operating environment provides the following naming services.

- DNS, the *Domain Name System* (see “DNS” on page 30)
- `/etc` files, the original UNIX™ naming system (see “`/etc` Files” on page 30)
- NIS, the *Network Information Service* (see “NIS” on page 30)
- NIS+, the *Network Information Service Plus* (see *System Administration Guide: Naming and Directory Services (FNS and NIS+)*)
- FNS, the *Federated Naming Service* (see *System Administration Guide: Naming and Directory Services (FNS and NIS+)*)

Most modern networks use two or more of these services in combination. When more than one service is used, they are coordinated by the `nsswitch.conf` file which is discussed in Chapter 2.

DNS

DNS is the naming service provided by the Internet for TCP/IP networks. It was developed so that machines on the network could be identified with common names instead of Internet addresses. DNS performs naming between hosts within your local administrative domain and across domain boundaries.

The collection of networked machines that use DNS are referred to as the *DNS namespace*. The DNS namespace can be divided into a hierarchy of *domains*. A DNS domain is a group of machines. Each domain is supported by two or more *name servers*, a principal server and one or more secondary servers. Each server implements DNS by running a daemon called `in.named`. On the client's side, DNS is implemented through the "resolver." The resolver's function is to resolve users' queries. It queries a name server, which then returns either the requested information or a referral to another server.

/etc Files

The original host-based UNIX naming system was developed for standalone UNIX machines and then adapted for network use. Many old UNIX operating systems and machines still use this system, but it is not well suited for large complex networks.

NIS

The *Network Information Service* (NIS) was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information about machine names and addresses, users, the network itself, and network services. This collection of network information is referred to as the *NIS namespace*.

NIS namespace information is stored in NIS maps. NIS maps were designed to replace UNIX `/etc` files, as well as other configuration files, so they store much more than names and addresses. As a result, the NIS namespace has a large set of maps. See "Working With NIS Maps" on page 155 for more information.

NIS uses a client-server arrangement similar to DNS. Replicated NIS servers provide services to NIS clients. The principal servers are called *master servers*, and for reliability, they have backup, or *slave servers*. Both master and slave servers use the NIS information retrieval software and both store NIS maps. For more information on NIS Architecture and NIS Administration, see Chapter 8 and Chapter 9.

NIS+

The *Network Information Service Plus* (NIS+) is similar to NIS but with many more features. NIS+ is not an extension of NIS. It is a different software program.

The NIS+ naming service is designed to conform to the shape of the organization that installs it for almost any network configuration. Unlike NIS, the NIS+ namespace is dynamic because updates can occur and be put into effect at any time by any authorized user.

NIS+ enables you to store information about machine addresses, security information, mail information, Ethernet interfaces, and network services in central locations where all machines on a network can have access to it. This configuration of network information is referred to as the NIS+ *namespace*.

The NIS+ namespace is hierarchical and is similar in structure to the UNIX directory file system. The hierarchical structure allows an NIS+ namespace to be configured to conform to the logical hierarchy of an organization. The namespace's layout of information is unrelated to its *physical* arrangement. Thus, an NIS+ namespace can be divided into multiple domains that can be administered autonomously. Clients might have access to information in other domains in addition to their own if they have the appropriate permissions.

NIS+ uses a client-server model to store and have access to the information contained in an NIS+ namespace. Each domain is supported by a set of servers. The principal server is called the *primary* server and the backup servers are called *secondary servers*. The network information is stored in 16 standard NIS+ tables in an internal NIS+ database. Both primary and secondary servers run NIS+ server software and both maintain copies of NIS+ tables. Changes made to the NIS+ data on the master server are incrementally propagated automatically to the secondary servers.

NIS+ includes a sophisticated security system to protect the structure of the namespace and its information. It uses authentication and authorization to verify whether a client's request for information should be fulfilled. *Authentication* determines whether the information requester is a valid user on the network. *Authorization* determines whether a particular user is allowed to have or modify the information requested. See *System Administration Guide: Naming and Directory Services (FNS and NIS+)* for a more detailed description of NIS+ security and administering it.

For information on making the transition from NIS+ to LDAP, see Chapter 19.

FNS

See *System Administration Guide: Naming and Directory Services (FNS and NIS+)* for information about FNS.

LDAP Naming Services

Solaris 9 supports LDAP (Lightweight Directory Access Protocol) in conjunction with the Sun ONE™ Directory Server 5.1, as well as other LDAP Directory Servers.

See Chapter 12 for more information.

Naming Services: A Quick Comparison

	DNS	NIS	NIS+	FNS	LDAP
NAMESPACE	Hierarchical	Flat	Hierarchical	Hierarchical	Hierarchical
DATA STORAGE	Files/resource records	2 column maps	Multi-columned tables	Maps	Directories [varied]
SERVER NAMES	Master/slave	Master/slave	Root master/non-root master primary/secondary cache/stub	N/A	Master/replica
SECURITY	SSL	None (root or nothing)	DES Authentication	None (root or nothing)	SSL
TRANSPORT	TCP/IP	LAN	LAN	LAN	TCP/IP
SCALE	Global	LAN	LAN	Global (w/ DNS)/LAN	Global

The Name Service Switch (Overview)

This chapter describes the name service switch, what it does, and how clients use it to obtain naming information from one or more sources. You use the name service switch to coordinate usage of different naming services.

About the Name Service Switch

The name service switch is a file named `nsswitch.conf`(4). It controls how a client machine or application obtains network information. It is used by client applications that call any of the `getXbyY()` interfaces such as the following.

- `gethostbyname()`
- `getpwuid()`
- `getpwnam()`
- `getipnodebyname()`

Each machine has a switch file in its `/etc` directory. Each line of that file identifies a particular type of network information, such as `host`, `password`, and `group`, followed by one or more sources where the client is to look for that information.

A client can obtain naming information from one or more of the switch's sources. For example, an NIS+ client could obtain its hosts information from an NIS+ table and its password information from a local `/etc` file. In addition, it could specify the conditions under which the switch must use each source. See Table 2-1.

The Solaris operating environment automatically loads an `nsswitch.conf` file into every machine's `/etc` directory as part of the installation process. Four alternate (template) versions of the switch file are also loaded into `/etc` for LDAP, NIS, NIS+, or files. See "The `nsswitch.conf` Template Files" on page 38.

These four files are alternate default switch files. Each one is designed for a different primary naming service: `/etc` files, NIS, NIS+, or LDAP. When the Solaris software is first installed on a machine, the installer selects the machine's default naming service: NIS+, NIS, local files, or LDAP. During installation, the corresponding template file is copied to `nsswitch.conf`. For example, for a machine client using LDAP, the installation process copies `nsswitch.ldap` to `nsswitch.conf`. Unless you have an unusual namespace, the default template file as copied to `nsswitch.conf` should be sufficient for normal operation.

No default file is provided for DNS or IPv6, but you can edit any of these files to use DNS or IPv6. See "DNS and Internet Access" on page 44 or "IPv6 and Solaris Naming Services" on page 44.

If you later change a machine's primary naming service, you copy the appropriate alternate switch file to `nsswitch.conf`. See "The `nsswitch.conf` Template Files" on page 38. You can also change the sources of particular types of network information used by the client by editing the appropriate lines of the `/etc/nsswitch.conf` file. The syntax for doing this is described below, and additional instructions are provided in "Modifying the name service switch" on page 43.

Format of the `nsswitch.conf` File

The `nsswitch.conf` file is essentially a list of 16 types of information and the sources that `getXXbyYY()` routines search for that information. The 16 types of information, not necessarily in this order, are the following.

- `aliases`
- `bootparams`
- `ethers`
- `group`
- `hosts`
- `ipnodes`
- `netgroup`
- `netmasks`
- `networks`
- `passwd` (includes shadow information)
- `protocols`
- `publickey`
- `rpc`
- `services`
- `automount`
- `sendmailvars`

The following table provides a description of the kind of sources that can be listed in the switch file for the information types above.

TABLE 2-1 Switch File Information Sources

Information Sources	Description
files	A file stored in the client's <code>/etc</code> directory. For example, <code>/etc/passwd</code>
nisplus	An NIS+ table. For example, the <code>hosts</code> table.
nis	An NIS map. For example, the <code>hosts</code> map.
compat	<code>compat</code> can be used for password and group information to support old-style <code>+</code> or <code>-</code> syntax in <code>/etc/passwd</code> , <code>/etc/shadow</code> , and <code>/etc/group</code> files.
dns	Can be used to specify that host information be obtained from DNS.
ldap	Can be used to specify entries be obtained from the LDAP directory.

Search Criteria

Single Source. If an information type has only one source, such as `nisplus` a routine using the switch searches for the information in that source *only*. If it finds the information, it returns a `success` status message. If it does not find the information, it stops searching and returns a different status message. What the routine does with the status message varies from routine to routine.

Multiple Sources. If a table has more than one source for a given information type, the switch directs the routine to start searching for the information in the first source that is listed. If it finds the information, it returns a `success` status message. If it does not find the information in the first source, it tries the next source. The routine will search through all of the sources until it has found the information it needs, or it is halted by encountering a `return` specification. If all of the listed sources are searched without finding the information, the routine stops searching and returns a `non-success` status message.

Switch Status Messages

If a routine finds the information, it returns a `success` status message. If it does not find the information for which it is looking, it returns one of three unsuccessful status messages, depending on the reason for not finding the information. Possible status messages are listed in the following table.

TABLE 2-2 Switch Search Status Messages

Status Message	Meaning of Message
SUCCESS	The requested entry was found in the specified source.

TABLE 2-2 Switch Search Status Messages (Continued)

Status Message	Meaning of Message
UNAVAIL	The source is not responding or is unavailable. That is, the NIS+ table, or NIS map, or <code>/etc</code> file could not be found or accessed.
NOTFOUND	The source responded with "No such entry." In other words, the table, map, or file was accessed but it did not contain the needed information.
TRYAGAIN	The source is busy; it might respond next time. In other words, the table, map, or file was found, but it could not respond to the query.

Switch Action Options

You can instruct the switch to respond to status messages with either of these two *actions* shown in the following table.

TABLE 2-3 Responses to Switch Status Messages

Action	Meaning
return	Stop looking for the information.
continue	Try the next source, if there is one.

Default Search Criteria

The combination of `nsswitch.conf` file status message and action option determines what the routine does at each step. This combination of status and action is called the search *criteria*.

The switch's default search criteria are the same for every source. Described in terms of the status messages listed above, they are the following.

- `SUCCESS=return`. Stop looking for the information and proceed using the information that has been found
- `UNAVAIL=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status
- `NOTFOUND=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status
- `TRYAGAIN=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status

Because these are the default search criteria, they are assumed. That is, you do not have to explicitly specify them in the switch file. You can change these default search criteria by explicitly specifying some other criteria using the `STATUS=action` syntax show above. For example, the default action for a `NOTFOUND` condition is to continue

the search to the next source. To specify that for a particular type of information, such as `networks`, the search is to halt on a `NOTFOUND` condition, you would edit the `networks` line of the switch file to read as follows.

```
networks: nis [NOTFOUND=return] files
```

The `networks: nis [NOTFOUND=return] files` line specifies a non-default criterion for the `NOTFOUND` status. Non-default criteria are delimited by square brackets.

In this example, the search routine behaves as follows

- If the `networks` map is available and contains the needed information, the routine returns with a `SUCCESS` status message
- If the `networks` map is not available, the routine returns with an `UNAVAIL` status message and by default continues on to search the appropriate `/etc` file
- If the `networks` map is available and found, but it does not contain the needed information, the routine returns with a `NOTFOUND` message. But, instead of continuing on to search the appropriate `/etc` file, which would be the default behavior, the routine stops searching
- If the `networks` map is busy, the routine returns with an `TRYAGAIN` status message and by default continues on to search the appropriate `/etc` file

What if the Syntax is Wrong?

Client library routines contain compiled-in default entries that are used if an entry in the `nsswitch.conf` file is either missing or syntactically incorrect. These entries are the same as the switch file's defaults.

The name service switch assumes that the spelling of table and source names is correct. If you misspell a table or source name, the switch uses default values.

Auto_home and Auto_master

The switch search criteria for the `auto_home` and `auto_master` tables and maps is combined into one category called `automount`.

Timezone and the Switch File

The `timezone` table does not use the switch, so it is not included in the switch file's list.

Comments in `nsswitch.conf` Files

Any `nsswitch.conf` file line beginning with a comment character (`#`) is interpreted as a comment line and is ignored by routines that search the file.

When a comment character (`#`) is included in the middle of the line, characters preceding the comment mark *are* interpreted by routines that search the `nsswitch.conf` file. Characters to the right of the comment mark are interpreted as comments and ignored.

TABLE 2-4 Switch File Comment Examples

Type of Line	Example
Comment line (not interpreted).	<code># hosts: nisplus [NOTFOUND=return] files</code>
Fully interpreted line.	<code>hosts: nisplus [NOTFOUND=return] file</code>
Partially interpreted line (the <code>files</code> element not interpreted)	<code>hosts: nisplus [NOTFOUND=return] # files</code>

Keyserver and `publickey` Entry in the Switch File



Caution – You must restart the keyserver after you make a change to `nsswitch.conf`

The keyserver reads the `publickey` entry in the name service switch configuration file only when the keyserver is started. As a result, if you change the switch configuration file, the keyserver does not become aware of changes to the `publickey` entry until it is restarted.

The `nsswitch.conf` Template Files

Four `nsswitch.conf(4)` template files are provided with the Solaris operating environment to accommodate different naming services. Each of them provides a different default set of primary and subsequent information sources.

The four template files are the following.

- *LDAP template file.* The `nsswitch.ldap` configuration file specifies the LDAP directory as the primary source of information for the machine.

Note – In order to use LDAP naming services, you must also properly configure all LDAP client machines, in addition to modifying the `nsswitch.conf`. See Chapter 16 for more information.

- *NIS+ template file.* The `nsswitch.nisplus` configuration file specifies NIS+ as the primary source for all information except `passwd`, `group`, `automount`, and `aliases`. For those four files, the primary source is local `/etc` files and the secondary source is an NIS+ table. The `[NOTFOUND=return]` search criterion instructs the switch to stop searching the NIS+ tables if it receives a “No such entry” message from them. It searches through local files only if the NIS+ server is unavailable
- *NIS template file.* The `nsswitch.nis` configuration file is almost identical to the NIS+ configuration file, except that it specifies NIS maps in place of NIS+ tables. Because the search order for `passwd` and `group` is `files nis`, you don’t need to place the `+` entry in the `/etc/passwd` and `/etc/group` files
- *Files template file.* The `nsswitch.files` configuration file specifies local `/etc` files as the only source of information for the machine. There is no “files” source for `netgroup`, so the client will not use that entry in the switch file

Copy the template file that most closely meets your requirements to `thenswitch.conf` configuration file and then modify the file as needed.

For example, to use the LDAP template file, you would type the following command.

```
mymachine# cp nsswitch.ldap nsswitch.conf
```

The Default Switch Template Files

The following are the four switch files supplied with Solaris operating environment.

EXAMPLE 2-1 NIS+ Switch File Template (`nsswitch.nisplus`)

```
#
#
# /etc/nsswitch.nisplus:
#
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it uses NIS+ (NIS Version 3) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
#
# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
```

EXAMPLE 2-1 NIS+ Switch File Template (nsswitch.nisplus) (Continued)

```
passwd: files nisplus
group: files nisplus
# consult /etc "files" only if nisplus is down.
hosts: nisplus [NOTFOUND=return] files
# Uncomment the following line, and comment out the above, to use
# both DNS and NIS+. You must also set up the /etc/resolv.conf
# file for DNS name server lookup. See resolv.conf(4).
# hosts: nisplus dns [NOTFOUND=return] files
services: nisplus [NOTFOUND=return] files
networks: nisplus [NOTFOUND=return] files
protocols: nisplus [NOTFOUND=return] files
rpc: nisplus [NOTFOUND=return] files
ethers: nisplus [NOTFOUND=return] files
netmasks: nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files
publickey: nisplus
netgroup: nisplus
automount: files nisplus
aliases: files nisplus
sendmailvars: files nisplus
```

EXAMPLE 2-2 NIS Switch File Template

```
#
# /etc/nsswitch.nis:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it uses NIS (YP) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
#
# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
passwd: files nis
group: files nis
# consult /etc "files" only if nis is down.
hosts: nis [NOTFOUND=return] files
networks: nis [NOTFOUND=return] files
protocols: nis [NOTFOUND=return] files
rpc: nis [NOTFOUND=return] files
ethers: nis [NOTFOUND=return] files
netmasks: nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey: nis [NOTFOUND=return] files
netgroup: nis
automount: files nis
aliases: files nis
# for efficient getservbyname() avoid nis
services: files nis
sendmailvars: files
```

EXAMPLE 2-3 Files Switch File Template

```
#
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
passwd: files
group: files
hosts: files
networks: files
protocols: files
rpc: files
ethers: files
netmasks: files
bootparams: files
publickey: files
# At present there isn't a 'files' backend for netgroup;
# the system will figure it out pretty quickly, and will not use
# netgroups at all.
netgroup: files
automount: files
aliases: files
services: files
sendmailvars: files
```

EXAMPLE 2-4 LDAP Switch File Template

```
#
# /etc/nsswitch.ldap:
#
# An example file that could be copied over to /etc/nsswitch.conf; it
# uses LDAP in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet" transports.

# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
passwd:    files ldap
group:     files ldap

hosts:     ldap [NOTFOUND=return] files

networks:  ldap [NOTFOUND=return] files
protocols: ldap [NOTFOUND=return] files
rpc:       ldap [NOTFOUND=return] files
ethers:    ldap [NOTFOUND=return] files
netmasks:  ldap [NOTFOUND=return] files
bootparams: ldap [NOTFOUND=return] files
publickey: ldap [NOTFOUND=return] files
```

EXAMPLE 2-4 LDAP Switch File Template (Continued)

```
netgroup:    ldap

automount:  files ldap
aliases:    files ldap

# for efficient getservbyname() avoid ldap
services:   files ldap
sendmailvars: files
```

The `nsswitch.conf` File

The default `nsswitch.conf` file that is installed when you install the Solaris operating environment for the first time is determined by which naming service you select during the Solaris software installation process. Each line of that file identifies a particular type of network information, such as host, password, and group, followed by one or more sources, such as NIS+ tables, NIS maps, the DNS hosts table, or local `/etc`, where the client is to look for that information. When you chose a naming service, the switch template file for that service is copied to create the new `nsswitch.conf` file. For example, if you choose NIS+, the `nsswitch.nisplus` file is copied to create a new `nsswitch.conf` file.

An `/etc/nsswitch.conf` file is automatically loaded into every machine's `/etc` directory by the Solaris 9 release software, along with the following alternate (template) versions.

- `/etc/nsswitch.nisplus`
- `/etc/nsswitch.nis`
- `/etc/nsswitch.files`
- `/etc/nsswitch.ldap`

These alternate template files contain the default switch configurations used by the NIS+ and NIS services, local files, and LDAP. No default file is provided for DNS, but you can edit any of these files to use DNS. See Chapter 5. When the Solaris operating environment is first installed on a machine, the installer selects the machine's default naming service: NIS+, NIS, local files, or LDAP. During installation, the corresponding template file is copied to `/etc/nsswitch.conf`. For example, for a machine client using NIS+, the installation process copies `nsswitch.nisplus` to `nsswitch.conf`.

If your network is connected to the Internet and you want users to be able to access Internet hosts using DNS, you must enable DNS forwarding.

Unless you have an unusual namespace, the default template file as copied to `nsswitch.conf` should be sufficient for normal operation.

Selecting a Different Configuration File

When you change a machine's naming service, you need to modify that machine's switch file accordingly. For example, if you change a machine's naming service from NIS to NIS+, you need to install a switch file appropriate for NIS+. You change switch files by copying the appropriate template file to `nsswitch.conf`.

If you are installing NIS+ on a machine using the NIS+ installation scripts, the NIS+ template script is copied to `nsswitch.conf` for you. In this case, you do not have to configure the switch file unless you want to customize it.

Before proceeding to change switch files, make sure the sources listed in the file are properly set up. In other words, if you are going to select the NIS+ version, the client must eventually have access to NIS+ service; if you are going to select the local files version, those files must be properly set up on the client.

▼ Modifying the name service switch

To change to a switch file, follow these steps.

1. **Become superuser.**
2. **Copy the alternate file appropriate for the machine's naming service over the `nsswitch.conf` file.**

NIS+ Version (done automatically for you by NIS+ scripts)

```
client1# cd /etc
```

```
client1# cp nsswitch.nisplus nsswitch.conf
```

NIS Version

```
client1# cd /etc
```

```
client1# cp nsswitch.nis nsswitch.conf
```

Local /etc Files Version

```
client# cd /etc
```

```
client# cp nsswitch.files nsswitch.conf
```

3. **Reboot the machine.**

The `nscd` naming service cache daemon caches switch information. Some library routines do not periodically check the `nsswitch.conf` file to see whether it has been changed. You must reboot the machine to make sure that the daemon and those routines have the latest information in the file.

Note – In order to use LDAP naming services, you must also properly configure all LDAP client machines, in addition to modifying the `nsswitch.conf`. See for Chapter 16 more information.

DNS and Internet Access

The `nsswitch.conf` file also controls DNS forwarding for clients as described in the following subsections. DNS forwarding grants Internet access to clients. For information on how to set DNS forwarding for NIS and NIS+, see *System Administration Guide: Naming and Directory Services (DNS and NIS+)*.

IPv6 and Solaris Naming Services

Note – DNS and LDAP are IPv6 “compatible” in the sense that one can *store* IPv6 addresses. However, as of Solaris 9, one cannot use an IPv6 transport for client-server DNS or LDAP traffic. The LDAP naming service cannot yet function on an IPv6-only network.

NIS and NIS+ support storing IPv6 data, as well as using IPv6 transports for NIS/NIS+ protocol traffic.

The `nsswitch.conf` file controls search criteria for IPv6 addresses. IPv6 increases the IP address size from 32 bits to 128 bits to support more levels of addressing hierarchy and provide a greater number of addressable nodes. For more information about IPv6, its configuration and implementation, see *System Administration Guide: IP Services*.

Use the new `ipnodes` source for IPv6 addresses. The `/etc/inet/ipnodes` file stores both IPv4 and IPv6 addresses. The `/etc/inet/ipnodes` file uses the same format convention as the `/etc/hosts` file.

IPv6 aware naming services use the new `ipnodes` source for its search forwarding. For instance, if LDAP is aware of IPv6 addresses, specify the following.

```
ipnodes: ldap [NOTFOUND=return] files
```



Caution – `ipnodes` defaults to `files`. During the transition from IPv4 to IPv6, where all naming services are not aware of IPv6 addresses, accept the `files` default. Otherwise, unnecessary delays (such as boot timing delays) might result during the resolution of addresses.



Caution – An application searches all `ipnodes` databases for IPv4 addresses before searching for IPv4 addresses in the `hosts` databases. Before specifying `ipnodes`, consider the inherent delay of searching both databases for IPv4 addresses.

Ensuring Compatibility With +/- Syntax

If +/- is used in `/etc/passwd`, `/etc/shadow`, and `/etc/group` files, you will need to modify the `nsswitch.conf` file to insure compatibility.

- **NIS+**. To provide +/- semantics with NIS+, change the `passwd` and `groups` sources to `compat` and add a `passwd_compat: nisplus` entry to the `nsswitch.conf` file after the `passwd` or `group` entry as shown below.

```
passwd: compat
passwd_compat: nisplus
group: compat
group_compat: nisplus
```

The above specifies that client routines obtain their network information from `/etc` files and NIS+ tables as indicated by the +/- entries in the files.

- **NIS**. To provide the same syntax as in the Sun Operating Environment 4.x release, change the `passwd` and `groups` sources to `compat`.

```
passwd: compat
group: compat
```

This specifies that `/etc` files and NIS maps as indicated by the +/- entries in the files.

Note – Users working on a client machine being served by an NIS+ server running in NIS compatibility mode cannot run `ypcat` on the `netgroup` table. Doing so will give you results as if the table were empty even if it has entries.

The Switch File and Password Information



Caution – `files` should be the first source in the `nsswitch.conf` file for `passwd` information. If `files` is not the first source, network security could be weakened and users could encounter log in difficulty.

For example, in an NIS+ environment, the `passwd` line of the `nsswitch.conf` file should look like the following.

```
passwd: files nisplus
```

In an NIS environment, the `passwd` line of the `nsswitch.conf` file should look like the following.

```
passwd: files nis
```

PART II DNS Setup and Administration

This part describes the setup, configuration, administration and troubleshooting of DNS naming service in the Solaris operating environment.

Domain Name System (Overview)

This chapter describes the structure and provides an overview of the Domain Name System (DNS).

Note – One of the most common and important uses of DNS is connecting your network to the global Internet. To connect to the Internet, your network IP address must be registered with whomever is administering your parent domain.

This chapter covers the following topics.

- “DNS Basics” on page 49
- “Server Configuration and Data File Names” on page 53
- “Domain Names” on page 56
- “The `resolv.conf` File” on page 57
- “The `named.conf` File” on page 58
- “Zones” on page 64

DNS Basics

The Domain Name System (DNS) is an application-layer protocol that is part of the standard TCP/IP protocol suite. This protocol implements the DNS naming service, which is the naming service used on the Internet.

This section introduces the basic DNS concepts. It assumes that you have some familiarity with network administration, particularly TCP/IP, and some exposure to other naming services, such as NIS+ and NIS.

Refer to Chapter 4 for information regarding initial setup and configuration of DNS.

Note – DNS, NIS+, NIS, and FNS provide similar functionality and sometimes use the same terms to define different entities. Thus, this chapter takes care to define terms like domain and name server according to their DNS functionality, a very different functionality than NIS+ and NIS domains and servers.

Name-to-Address Resolution

Though it supports the complex, worldwide hierarchy of computers on the Internet, the basic function of DNS is actually very simple: providing *name-to-address resolution* for TCP/IP-based networks. Name-to-address resolution, also referred to as mapping, is the process of finding the IP address of a computer in a database by using its host name as an index.

Name-to-address mapping occurs when a program running on your local machine needs to contact a remote computer. The program most likely will know the host name of the remote computer but might not know how to locate it, particularly if the remote machine is in another company, miles from your site. To get the remote machine's address, the program requests assistance from the DNS software running on your local machine, which is considered a *DNS client*.

Your machine sends a request to a *DNS name server*, which maintains the distributed DNS database. The files in the DNS database bear little resemblance to the NIS+ *host* or *ipnodes* Table or even the local `/etc/hosts` or `/etc/inet/ipnodes` file, though they maintain similar information: the host names, the *ipnode* names, IPv4 and IPv6 addresses, and other information about a particular group of computers. The name server uses the host name your machine sent as part of its request to find or “resolve” the IP address of the remote machine. It then returns this IP address to your local machine *if* the host name is in its DNS database.

The following figure shows name-to-address mapping as it occurs between a DNS client and a name server, probably on the client's local network.

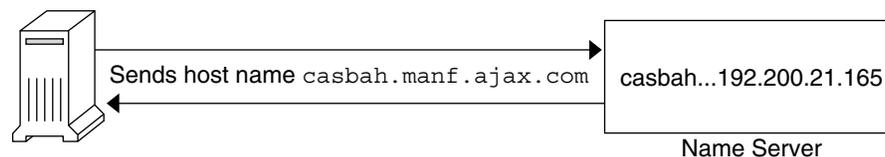


FIGURE 3-1 Name to Address Resolution

If the host name is not in that name server's DNS database, this indicates that the machine is outside of its authority, or, to use DNS terminology, outside the *local administrative domain*. Thus, each name server is spoken of as being “authoritative” for its local administrative domain.

Fortunately, the local name server maintains a list of host names and IP addresses of *root domain name servers*, to which it will forward the request from your machine. These root name servers are authoritative for huge organizational domains, as explained fully in “DNS Hierarchy and the Internet” on page 61. These hierarchies resemble UNIX file systems, in that they are organized into an upside down tree structure.

Each root name server maintains the host names and IP addresses of top level domain name servers for a company, a university, or other large organizations. The root name server sends your request to the top-level name servers that it knows about. If one of these servers has the IP address for the host you requested, it will return the information to your machine. If the top-level servers do not know about the host you requested, they pass the request to second-level name servers for which they maintain information. Your request is then passed on down through the vast organizational tree. Eventually, a name server that has information about your requested host in its database will return the IP address back to your machine.

The following figure shows name-to-address resolution outside the local domain.

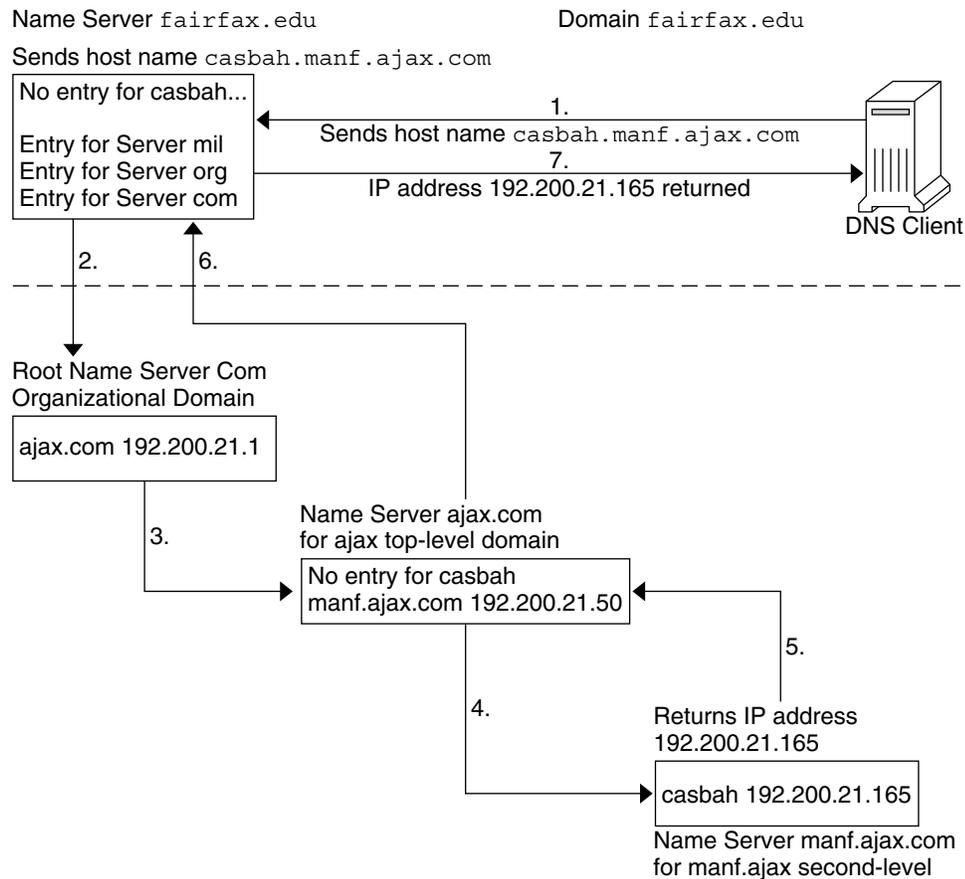


FIGURE 3-2 Name to Address Resolution for a Remote Host

DNS Administrative Domains

From a DNS perspective, an *administrative domain* is a group of machines which are administered as a unit. Information about this domain is maintained by at least two name servers, which are “authoritative” for the domain. The DNS domain is a logical grouping of machines. The domain groupings could correspond to a physical grouping of machines, such as all machines attached to the Ethernet in a small business. Similarly, a local DNS domain could include all machines on a vast university network that belong to the computer science department or to university administration.

For example, suppose the Ajax company has two sites, one in San Francisco and one in Seattle. The `Retail.Sales.Ajax.com` domain might be in Seattle and the `Wholesale.Sales.Ajax.com` domain might be in San Francisco. One part of the `Sales.Ajax.com` domain would be in one city, the other part in the second city.

Each administrative domain must have its own unique subdomain name. Moreover, if you want your network to participate in the Internet, the network must be part of a registered administrative domain. The section “Joining the Internet” on page 62 has full details about domain names and domain registration.

`in.named` and DNS Name Servers

As mentioned previously, name servers in an administrative domain maintain the DNS database. They also run the `in.named` daemon, which implements DNS services. `in.named` is a public domain TCP/IP program and is included with the Solaris operating environment.

Note – The `in.named` daemon is also called the Berkeley Internet Name Domain service, or BIND, because it was developed at the University of California at Berkeley.

There are three types of DNS name servers.:

- Master server
- Slave server
- Stub server

Each domain must have one master server and should have at least one slave server to provide backup. “Implementing DNS” on page 87 explains primary and secondary servers in detail.

Server Configuration and Data File Names

To function correctly, the `in.named` daemon requires a configuration file and four data files.

Configuration File

The master server configuration file is `/etc/named.conf`. The configuration file contains a list of domain names and the file names containing host information. See “The `named.conf` File” on page 98 for additional information on the `named.conf` file.

Names of DNS Data Files

If you are internally consistent, you can name the zone data files anything you want. This flexibility might lead to some confusion when working at different sites or referring to different DNS manuals and books.

For example, the file names used in Sun manuals and at most many Solaris sites vary from those used in the book *DNS and BIND* published by O'Reilly & Associates and both of those nomenclatures have some differences from that used in the public-domain *Name Server Operations Guide for BIND*.

In addition, this manual and other DNS documentation use generic names that identify a file's main purpose, and specific example names for that file in code samples. For example, this manual uses the generic name `hosts` when describing the function and role of that file, and the example names `db.doc` and `db.sales` in code samples.

The required data files are the following.

- `/var/named/named.ca` See "The `named.ca` File" on page 101 for additional information on the `named.ca` file. As long as you are internally consistent, you can name this file anything you want.
- `/var/named/hosts` See "The `hosts` File" on page 103 for additional information on `hosts` files.

The name `hosts` is a generic name indicating the file's purpose and content. But to avoid confusion with `/etc/hosts`, you should name this file something other than `hosts`. The most common naming convention is `db.domainname`. Thus, the `hosts` file for the `doc.com` domain would be called `db.doc`.

If you have more than one zone, each zone must have its own `hosts` file and each of these zone `hosts` files must have a unique name. For example, if your DNS domain is divided into `doc.com` and `sales.doc.com` zones, you could name one `hosts` file `db.doc` and the other `db.sales`.

- `/var/named/hosts.rev` See "The `hosts.rev` File" on page 104 for additional information on the `hosts.rev` file.
The name `hosts.rev` is a generic name indicating the file's purpose and content. If you have more than one zone, each zone must have its own `hosts.rev` file and each of these zone `hosts.rev` files must have a unique name. For example, if your DNS domain is divided into `doc.com` and `sales.doc.com` zones, you could name one `hosts.rev` file `doc.rev` and the other `sales.rev`.
- `/var/named/named.local` See "The `named.local` File" on page 105 and for additional information on the `named.local` file. As long as you are internally consistent, you can name this file anything you want.

\$INCLUDE Files

An include file is any file named in an `$INCLUDE()` statement in a DNS data file. `$INCLUDE` files can be used to separate different types of data into multiple files for your convenience. See “`$INCLUDE Files`” on page 106.

For reference purposes, the following table compares BIND file names from the above mentioned sources.

TABLE 3-1 File Name Examples

Solaris Names	O'Reilly Names or Other Names	U.C. Berkeley Names	Content and Purpose of File
/etc/named.conf, same file name for all three sources			BIND 8.1 adds a new named.conf file to replace the earlier named.boot file. This configuration file adds security, startup options, logging. It specifies the type of server it is running on and selectively applies options on a per-zone or per-server basis, rather than all zones or servers. It contains a list of domain names and the names of the data files.
/etc/resolv.conf, same file name for all three sources			This file resides on every DNS client (including DNS servers) and designates the servers that the client queries for DNS information.
named.ca	db.cache db.root	root.cache	This file establishes the names of root servers and lists their addresses.
Generic: hosts Examples: db.doc, db.sales	Generic: db.domain Examples: db.movie, db.fx	Generic: hosts Example: ucbhosts	This file contains all the data about the machines in the local zone that the server serves.
Generic: hosts.rev Examples: doc.rev	Generic: db.ADDR Examples db.192.249.249 db.192.249.253	hosts.rev	This file specifies a zone in the in-addr.arpa. domain, a special domain that allows reverse (address-to-name) mapping.
named.local	Generic: db.cache Example: db.127.0.0	named.local	This file specifies the address for the local loopback interface, or local host.
\$INCLUDE files, same convention for all three sources			Any file identified by an <code>\$INCLUDE()</code> statement in a data file.

Domain Names

A *domain name* is the name assigned to a group of systems on a local network that share DNS administrative files. A domain name is required for the network information service database to work properly.

Default Domain Name

DNS obtains your default domain name from your `resolv.conf` file.

- If the `resolv.conf` file is not available, or does not identify a default domain, and if your enterprise-level naming service is either NIS+ or NIS, the Sun implementation of DNS obtains the default domain name from those services.
- If `resolv.conf` is not available or does not provide a domain name and you are *not* running either NIS+ or NIS, you must either provide a `resolv.conf` file on each machine that does specify the domain or set the `LOCALDOMAIN` environment variable.

Trailing Dots in Domain Names

When working with DNS-related files, follow these rules regarding the trailing dot in domain names:

- Use a trailing dot in domain names in `hosts`, `hosts.rev`, `named.ca`, and `named.local` data files. For example, `sales.doc.com.` is correct for these files.
- Do not use a trailing dot in domain names in `named.boot` or `resolv.conf` files. For example, `sales.doc.com` is correct for these files.

DNS Clients and the Resolver

To be a DNS client, a machine must run the *resolver*. The resolver is neither a daemon nor a single program. It is a set of dynamic library routines used by applications that need to know machine names. The resolver's function is to resolve users' queries. To do that, it queries a name server, which then returns either the requested information or a referral to another server. Once the resolver is configured, a machine can request DNS service from a name server.

The DNS name server uses several files to load its database. At the resolver level, it needs the file `/etc/resolv.conf` listing the addresses of the servers where it can obtain its information. The resolver reads this `resolv.conf` file to find the name of

the local domain and the location of name servers. It sets the local domain name and instructs the resolver routines to query the listed name servers for information. Normally, each DNS client system on your network has a `resolv.conf` file in its `/etc` directory. If a client does not have a `resolv.conf` file, it defaults to using a server at IP address `127.0.0.1`.

Whenever the resolver has to find the IP address of a host (or the host name corresponding to an address), the resolver builds a query package and sends it to the name servers listed in `/etc/resolv.conf`. The servers either answer the query locally or contact other servers known to them, ultimately returning the answer to the resolver.

When a machine's `/etc/nsswitch.conf` file specifies `hosts: dns` (or any other variant that includes `dns` in the `hosts` line), the resolver libraries are automatically used. If the `nsswitch.conf` file specifies some other naming service before `dns`, that naming service is consulted first for host information and only if that naming service does not find the host in question are the resolver libraries used.

For example, if the `hosts` line in the `nsswitch.conf` file specifies `hosts: nisplus dns`, the NIS+ naming service will first be searched for host information. If the information is not found in NIS+, then the DNS resolver is used. Since naming services such as NIS+ and NIS only contain information about hosts in their own network, the effect of a `hosts: nisplus dns` line in a switch file is to specify the use of NIS+ for local host information and DNS for information on remote hosts out on the Internet.

There are two kinds of DNS clients.

- *Client-only*

A client-only DNS client does not run `in.named`. Instead, it consults the resolver. The resolver knows about a list of name servers for the domain, to which queries are then directed.

- *Client-server*

A client-server uses the services provided by `in.named` to resolve queries forwarded to it by client-machine resolvers.

The `resolv.conf` File

For a detailed description of what the `resolv.conf` file does, see `resolv.conf(4)`.

See “Setting Up the `resolv.conf` File” on page 67 for a discussion on how to set up the `resolv.conf` file.

The named.conf File

BIND 8.1 added a new configuration file, `/etc/named.conf`, that replaces the `/etc/named.boot` file. The `/etc/named.conf` file establishes the server as a master, slave, or cache-only name server. It also specifies the zones over which the server has authority and which data files it should read to get its initial data.

The `/etc/named.conf` file contains statements that implement:

- Security through an access control list (ACL) that defines a collection of IP addresses that an NIS+ host can read and write
- Logging specifications
- Selectively applied options for a set of zones, rather than to all zones

The configuration file is read by `in.named` when the daemon is started by the server's startup script, `/etc/init.d/inetsvc`. The configuration file directs `in.named` to other servers or to local data files for a specified domain.

The `named.conf` file contains statements and comments. Statements end with a semicolon. Some statements can contain a block of statements. Again, each statement in the block is terminated with a semicolon.

TABLE 3-2 `named.conf` Statements

<code>acl</code>	Defines a named IP address match list used for access control. The address match list designates one or more IP addresses (dotted-decimal notation) or IP prefixes (dotted-decimal notation followed with a slash and the number of bits in the netmask). The named IP address match list must be defined by an <code>acl</code> statement before it can be used elsewhere. No forward references are allowed.
<code>include</code>	Inserts an include file at the point where the <code>include</code> statement is encountered. Use <code>include</code> to break up the configuration into more easily managed chunks.
<code>key</code>	Specifies a key ID used for authentication and authorization on a particular name server. See the <code>server</code> statement.
<code>logging</code>	Specifies what information the server logs and the destination of log messages.
<code>options</code>	Controls global server configuration options and sets default values for other statements.
<code>server</code>	Sets designated configuration options associated with a remote name server. Selectively applies options on a per-server basis, rather than to all servers.

TABLE 3-2 named.conf Statements (Continued)

zone	Defines a zone. Selectively applies options on a per-zone basis, rather than to all zones.
------	--

EXAMPLE 3-1 Example Master Configuration File for a Master Server

```

options {
    directory "/var/named";
    datasize 2098;
    forward only;
    forwarders {
        99.11.33.44;
    };
    recursion no;
    transfers-in 10;
    transfers-per-ns 2;
    allow-transfer {
        127.0.1.1/24;
    };
};

logging {
    category queries { default_syslog; };
};

include "/var/named/abcZones.conf"

// here are the names of the master files
zone "cities.zn" {
    type master;
    file "db.cities.zn";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "db.127.cities.zn";
};

zone "168.192.in-addr.arpa" {
    type master;
    file "db.cities.zn.rev";
};

zone "sales.doc.com" {
    type slave;
    file "slave/db.sales.doc";
    masters {
        192.168.1.151;
    };
};

zone "168.192.in-addr.arpa" {

```

EXAMPLE 3-1 Example Master Configuration File for a Master Server (Continued)

```
        type slave;
        file "slave/db.sales.doc.rev";
        masters {
            192.168.1.151;
        };
};
```

DNS Hierarchy in a Local Domain

If your company is large enough, it might support a number of domains, organized into a local namespace. The following figure shows a domain hierarchy that might be in place in a single company. The top-level, or “root” domain for the organization is `ajax.com`, which has three subdomains, `sales.ajax.com`, `test.ajax.com`, and `manf.ajax.com`.

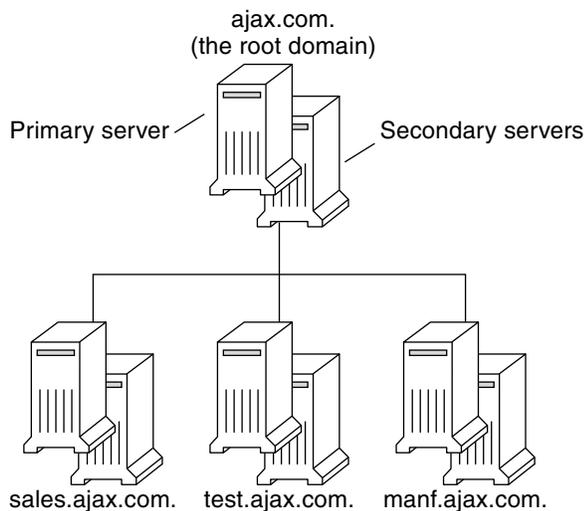


FIGURE 3-3 Hierarchy of DNS Domains in a Single Organization

DNS clients request service only from the servers that support their domain. If the domain’s server does not have the information the client needs, it forwards the request to its parent server, which is the server in the next higher domain in the hierarchy. If the request reaches the top-level server, the top-level server determines whether the domain is valid. If it is *not* valid, the server returns a “not found” type message to the client. If the domain is valid, the server routes the request down to the server that supports that domain.

DNS Hierarchy and the Internet

The domain hierarchy shown in the following figure is a “leaf” of the huge DNS namespace supported on the global Internet.

It consists of the root directory, represented as a dot (.), and two top level domain hierarchies, one organizational and one geographical. Note that the `com` domain introduced in this figure is one of a number of top-level organizational domains in existence on the Internet.

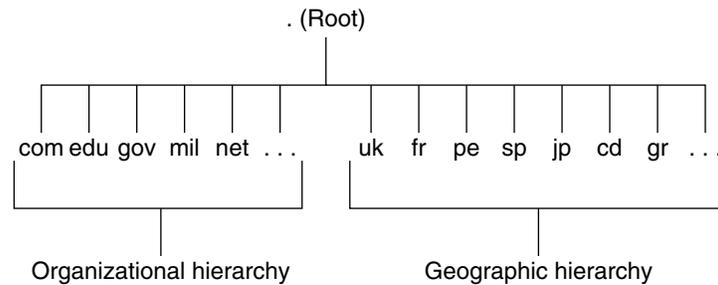


FIGURE 3-4 Hierarchy of Internet Domains

At the present time, the organizational hierarchy divides its namespace into the top-level domains listed shown in the following table. It is probable that additional top-level organizational domains will be added in the future.

TABLE 3-3 Internet Organizational Domains

Domain	Purpose
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
mil	Military groups
net	Major network support centers
org	Nonprofit organizations and others
int	International organizations

The geographic hierarchy assigns each country in the world a two or three-letter identifier and provides official names for the geographic regions within each country. For example, domains in Britain are subdomains of the `uk` top-level domain, Japanese domains are subdomains of `jp`, and so on.

Joining the Internet

The Internet root domain, top-level domains (organizational and geographical) are maintained by the various Internet governing bodies. People with networks of any size can “join” the Internet by registering their domain name in either the organizational or the geographical hierarchy.

Every DNS domain must have a domain name. If your site wants to use DNS for naming service *without* connecting to the Internet, you can use any name your organization wants for its your domains and subdomains, if applicable. However, if your site plans wants to join the Internet, it *must* register its domain name with the Internet governing bodies.

To join the Internet, do the following.

- Register your DNS domain name with the an appropriate Internet governing body.
- Obtain a network IP address from that governing body.

There are two ways to accomplish this.

- You can communicate directly with the appropriate Internet governing body or their agent.
- You can contract with an Internet Service Provider (ISP) to assist you. ISPs provide a wide range of services from consulting to actually hosting your Internet presence.

Domain Names

Domain names indicate a domain’s position in the overall DNS namespace, much as path names indicate a file’s position in the UNIX file system. After your local domain is registered, its name is added to the name of the Internet hierarchy to which it belongs. For example, the ajax domain shown in Figure 3–5 has been registered as part of the Internet com hierarchy. Therefore, its Internet domain name becomes `ajax.com`.

The following figure shows the position of the `ajax.com` domain in the DNS namespace on the Internet.

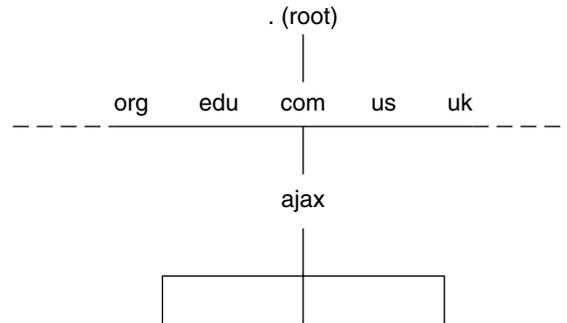


FIGURE 3-5 Ajax Domain's Position in the DNS Namespace

The `ajax.com` subdomains now have the following names.

```
sales.ajax.com
test.ajax.com
manf.ajax.com
```

DNS does not require domain names to be capitalized, though they can be. Here are some examples of machines and domain names.

```
boss.manf.ajax.com
quota.sales.ajax.com
```

The Internet organization regulates administration of its domains by granting each domain authority over the names of its hosts and by expecting each domain to delegate authority to the levels below it. Thus, the `com` domain has authority over the names of the hosts in its domain. It also authorizes the formation of the `ajax.com` domain and delegates authority over the names in that domain. The `ajax.com` domain, in turn, assigns names to the hosts in its domain and approves the formation of the `sales.ajax.com`, `test.ajax.com`, and `manf.ajax.com` domains.

Fully Qualified Domain Names (FQDNs)

A domain name is said to be *fully-qualified* when it includes the names of every DNS domain from the local domain on up to `."`, the DNS root domain. Conceptually, the fully qualified domain name indicates the path to the root, as does the absolute path name of a UNIX file. However, fully qualified domain names are read from lowest, on the left, to highest, on the right. Therefore, a fully-qualified domain name has the following syntax.

local_domain_name>.<*Internet_Org_name*>.
root domain ↗

The fully qualified domain names for the `ajax` domain and its subdomains are:

```
ajax.com.  
sales.ajax.com.  
test.ajax.com.  
manf.ajax.com.
```

Note the dot at the furthest right position of each name.

Zones

DNS service for a domain is managed on the set of name servers. Name servers can manage a single domains or multiple domains, or domains and some or all of their corresponding subdomains. The part of the namespace that a given name server controls is called a *zone*. Therefore, the name server is said to be authoritative for the zone. If you are responsible for a particular name server, you might be given the title “Zone Administrator”.

The data in a name server’s database are called *zone files*. One type of zone file stores IP addresses and host names. When someone attempts to connect to a remote host using a host name by a utility like `ftp` or `telnet`, DNS performs name-to-address mapping, by looking up the host name in the zone file and converting it into its IP address.

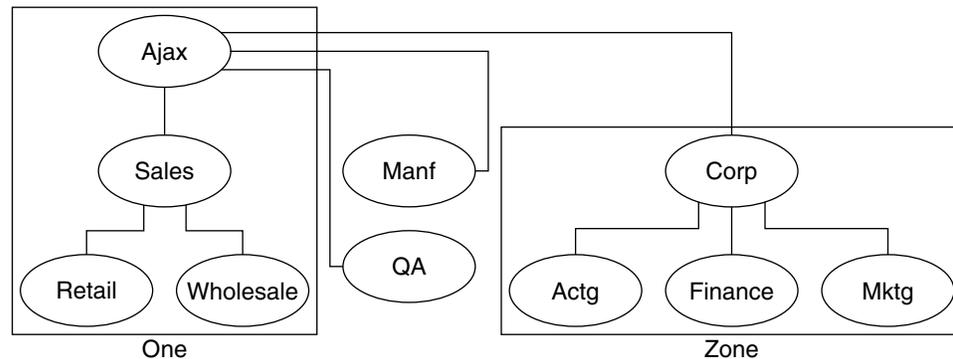


FIGURE 3-6 Domains and Zones

For example, the Ajax domain shown in the above contains a top domain (Ajax), four subdomains, and five sub-subdomains. It is divided into four zones. Thus, the Ajax name server administers a zone composed of the Ajax, Sales, Retail, and Wholesale domains. The Manf and QA domains are zones unto themselves served by their own name servers, and the Corp name server manages a zone composed of the Corp, Actg, Finance, and Mktg domains.

Reverse Mapping

The DNS database also includes zone files that use the IP address as a key to find the host name of the machine, enabling IP address to host name resolution. This process is called *reverse resolution* or more commonly, reverse mapping. Reverse mapping is used primarily to verify the identity of the machine that sent a message or to authorize remote operations on a local host.

The in-addr.arpa Domain

The in-addr.arpa domain is a conceptual part of the DNS namespace that uses IP addresses for its leaves, rather than domain names. It is the part of your zone that enables address-to-name mapping.

Just as DNS domain names are read with the lowest level subdomain occupying the furthest left position and the root at the far right, in-addr.arpa domain IP addresses are read from lowest level to the root. Thus, the IP addresses are read backward. For example, suppose a host has the IP address 192.168.21.165. In the in-addr.arpa zone files, its address is listed as 165.21.168.192.in-addr.arpa. with the dot at the end indicating the root of the in-addr.arpa domain.

Administering DNS (Tasks)

This chapter describes how to administer the Domain Name System (DNS).

The chapter covers the following topics.

- “Setting Up the `resolv.conf` File” on page 67
- “Configuring a Network For DNS” on page 68
- “How to Add DNS Compatibility and +/- Syntax” on page 74
- “Setting up DNS Servers” on page 75
- “Modifying DNS Data Files” on page 77
- “Adding and Deleting Clients” on page 79
- “Enabling a Client to Use IPv6” on page 80
- “Creating DNS Subdomains” on page 81
- “Solaris DNS BIND 8.2.4 Implementation” on page 84
- “DNS Forwarding” on page 85

Setting Up the `resolv.conf` File

A simple example `resolv.conf`(4) file for a server in the `doc.com` domain is shown below.

EXAMPLE 4-1 Sample `resolv.conf` File for DNS Server

```
;
; /etc/resolv.conf file for dnsmaster (sirius)
;
domain                doc.com
nameserver             192.168.0.0
nameserver             192.168.0.1
```

The first line of the `/etc/resolv.conf` file lists the domain name in the form:

domain *domainname*

Where *domainname* is the name registered with the Internet governing bodies (as of this writing, the InterNIC).

Note – No spaces or tabs are permitted at the end of the domain name. Make sure that you press return immediately after the last character of the domain name.

The second line identifies the server itself in the form:

```
nameserver 192.168.0.0
```

Succeeding lines list the IP addresses of one or two slave or cache-only name servers that the resolver should consult to resolve queries. Name server entries have the form:

```
nameserver IP_address
```

IP_address is the IP address of a slave or cache-only DNS name server. The resolver queries these name servers in the order they are listed until it obtains the information it needs.

Configuring a Network For DNS

To configure a network for DNS, you must set up a client and a server.

Setting Up a DNS Client

Set up the client(s) prior to setting up the DNS server.

▼ How to Set up a DNS Client

1. **Create the `/etc/resolv.conf` file.**

A simple example `resolv.conf` file for a client (non-server) machine in the `doc.com` domain is shown below.

EXAMPLE 4-2 Sample `resolv.conf` File

```
; Sample resolv.conf file for the machine polaris
domain doc.com
; try local name server
```

EXAMPLE 4-2 Sample `resolv.conf` File (Continued)

```
nameserver 10.0.0.1
; if local name server down, try these servers
nameserver 192.168.16.6
nameserver 192.168.16.7
; sort the addresses returned by gethostbyname(3c)
sortlist
130.155.160.0/255.255.240.0
130.155.0.0
```

The first line of the `/etc/resolv.conf` file lists the domain name in the following form.

```
domain domainname
```

Where *domainname* is the name registered with the Internet governing bodies (as of this writing, the InterNIC).

Note – No spaces or tabs are permitted at the end of the domain name. Make sure that you enter a hard carriage return immediately after the last character of the domain name.

The second line identifies the loopback name server in the form.

```
nameserver 10.0.0.1
```

Succeeding lines list the IP addresses of up to three DNS master, slave, or cache-only name servers that the resolver should consult to resolve queries. Do not list more than three master or slave servers. Name server entries have the following form.

```
nameserver IP_address
```

IP_address is the IP address of a master or slave DNS name server. The resolver queries these name servers in the order they are listed until it obtains the information it needs.

The fifth line of the `/etc/resolv.conf` file lists the address `sortlist` in the form:

```
sortlist
addresslist
```

addresslist specifies the sort order of the addresses returned by `gethostbyname(3c)`. In our example, `gethostbyname` returns the netmask pair `130.155.160.0/255.255.240.0` ahead of the IP address `130.155.0.0`.

2. Modify the `/etc/nsswitch.conf` file.

NIS. If your master enterprise-level naming service is NIS, with proper configuration, NIS is already DNS-enabled.

Files-based. If your master enterprise-level naming service is based on `/etc` files, or if your master enterprise-level naming service is NIS+, do the following.

- a. Become superuser.
- b. Open the `/etc/nsswitch.conf` file.
- c. DNS can be the *only* source or an *additional* source for the hosts information. Locate the `hosts` line and use DNS in one of the ways shown below.

```
hosts: files dns
```

or

```
hosts: nis dns [NOTFOUND=return] files
```

or

```
hosts: dns nis [NOTFOUND=return] files
```

Do *not* use the above syntax for NIS clients, or they will be forced to search for unresolved names twice in DNS.

- d. Specify DNS as a source of hosts information.
- e. Save the file and reboot.

Setting Up a DNS Server

▼ How to Set Up a DNS Server

1. Become superuser.
2. Set the server up as a DNS client (this includes setting up the server's `resolv.conf` file). See "Setting Up a DNS Client" on page 68.
3. Set up the boot file. See "Example Boot Files" on page 88.
4. Set up the data files. You need to set up four data files.
 - `named.ca`
 - `hosts`
 - `hosts.rev`
 - `named.local`
5. Initialize the server. See "Initializing the Server" on page 75.
6. Test the server. See "Testing Your Installation" on page 75.

Note – The most common use of DNS is to connect your network to the Internet. To connect to the Internet, your network IP address must be registered with whomever is administering your parent domain. Who that administrator is varies according to your geographic location and type of parent domain. This manual does not describe how to register networks with domain administrators.

How to Specify a Master Server

The two types of master server are as follows.

- *Zone master server.* Each zone has one server that is designated as the *master* master server for that zone. A zone's master master server is the *authoritative* server for that zone.
- *Zone slave server.* A zone can have one or more *slave* master servers. Slave master servers obtain their DNS data from the zone's master server.

To specify a server as the master server for a given zone, you create three master records in that server's `named.boot` file.

1. Create the master record for the zone.

This record designates the server as a master server for the zone and tells the server where to find the authoritative `hosts` file. A "master" record has three fields.

- The first field designates the server as master.
- The second field identifies the zone the master serves.
- The third field identifies the `hosts` file.

For example, the following line in a boot file specifies that the server is the master server for the `doc.com` zone, using authoritative data from the file `db.doc`.

```
master    doc.com    db.doc
```

2. Create a master record for the zone's reverse map.

This record designates the server as a master server for the zone's reverse address map, that is, the reverse address domain for `doc.com`. The record also tells the server where to find the authoritative `hosts` file. This record has three fields. The first field designates the server as master, the second field identifies the zone, and the third field identifies the `hosts.rev` file.

The reverse address domain for a zone contains the zone's IP address in reverse order followed by `in-addr.arpa`. For example, suppose that the `doc.com` zone's IP address is `10.0.0`. In that case, the reverse address domain would be `0.0.10.in-addr.arpa`.

Thus, the following line in a boot file specifies that the server is the master server for the reverse address domain of the `doc.com` zone, using authoritative data from the file `doc.rev`.

```
master 0.0.10.in-addr.arpa doc.rev
```

3. Create a master record for the reverse address of the local loopback interface or host.

This record designates the server as a master server for the loopback host, and tells the server where to find the authoritative `hosts` file. This record has three fields, the first field designates the server as master, the second field identifies the loopback host reverse address, and the third field identifies the `hosts` file.

Note – Loopback hosts are always identified as `0.0.10.in-addr.arpa`.

Thus, the following line in a boot file specifies that the server is the master server for the reverse address domain of the loopback host using authoritative data from the file named `local`.

```
master 0.0.10.in-addr.arpa named.local
```

How to Specify a Slave Server

A *slave* server maintains a copy of the data for the zone. The master server sends its data and delegates authority to the slave server. Clients can query a slave server for DNS information. By using slave servers, you can improve response time and reduce network overhead by spreading the load over multiple machines. Slave servers also provide redundancy in case the master server is not available.

When `in.named` starts, it requests all the data for the given zone from the master. The slave server then periodically checks with the master to see if it needs to update its database. The process of sending the most recent zone database from the master to the slave is called a *zone transfer*. Thus, you do not modify data files on a slave server, you modify the data files on the zone's master server and the slave servers update their files from the master.

To specify that a server is to be the slave server for a given zone, you create slave records in that server's `named.boot` file. Separate records can designate the server as a slave server for the zone, the zone's reverse address domain, and the loopback host.

A slave record has three required fields:

- The first field designates the server as slave.
- The second field identifies the zone it serves.
- The third field identifies the IP address of the master server for the zone from which the slave server obtains its authoritative data.

A "slave" record can have one or more optional fields after the required fields. The optional fields are:

- *slave servers*

After the IP address of the master server, you can add IP addresses of other slave servers. These provide additional sources from which the slave server can obtain data. Adding IP addresses of slave servers might, under some circumstances, reduce performance unless those IP addresses are additional network addresses of a multihomed master server.

- *Backup file*

After the IP address of the master (and optional slave) servers, you can add the name of a backup `hosts` file. If a backup file name is present, the slave server loads its data from that file, then checks with the master (and optional slave) servers to make sure that the data in the backup file is up to date. If the backup file is not up to date, it is brought up to date, based on the information received from the master server.

For example, the following lines in a boot file specify that the server is the slave server for the `doc.com` zone and its reverse address domain, that it obtains its authoritative data from the master server with an IP address of `172.16.0.1`, that it uses the server `172.16.0.2` as a slave source of zone data, and initially loads its data from the file `doc.com.bakup`:

```
slave doc.com 129.146.168.119 192.146.168.38 doc.com.bakup
slave 4.0.32.128.in-addr.arpa 129.146.168.119
```

In the context of the various example files presented in this chapter, the sample boot file lines above correspond to the boot file of the `dnsslave` server, which is an alias for the `sirius` machine whose IP address is `192.146.168.38`.

Note – A server can act as the master server for one or more zones, and as the slave server for one or more zones. The mixture of entries in the boot file determines whether a server is a master or slave server for a given zone.

How to Specify a Cache-Only or Stub Server

All servers are caching servers in the sense that they all maintain a cache of DNS data. A caching only or stub server is a server that is not a master server for any zone other than the `in-addr.arpa` domain.

A cache-only server does not maintain any authoritative data. It handles queries and asks the hosts listed in the `in.named` file for the information needed. In other words, a cache-only server handles the same kind of queries that authoritative name servers perform, but it does not maintain any authoritative data itself.

The following is a sample boot file for a cache only server.

EXAMPLE 4-3 Sample Master Boot File for Caching-only Server

```
;
; Sample named.boot file for caching-only name server
;
; type          domain          source file or host
;
directory /var/named
cache      .          named.ca
master     0.0.127.in-addr.arpa  named.local
```

You do not need a special line to designate a server as a cache-only server. What denotes a cache-only server is the absence of any `slave` or `master` authority lines in the boot file, except as noted below.

A cache-only server requires the following.

- A `directory` line in the boot file
- A `master 0.0.127.in-addr.arpa` line in the boot file
- A `cache . named.ca` line in the boot file

How to Add DNS Compatibility and +/- Syntax

This section describes how to add compatibility with the +/- syntax used in the `/etc/passwd`, `/etc/shadow`, and `/etc/group` files when you are using either NIS or NIS+ as your master naming service.

1. **Become superuser.**
2. **Open the `/etc/nsswitch.conf` file.**
3. **Change the `passwd` and `groups` sources to `compat`.**

- For use with NIS, enter:

```
passwd: compat
group: compat
```

- For NIS+, enter:

```
passwd: compat
passwd_compat: nisplus
group: compat
group_compat: nisplus
```

This provides the same syntax as in the Solaris 1.x release. It looks up `/etc` files and NIS maps as indicated by the +/- entries in the files.

4. Add `++` or `--` `netgroup` to the `/etc/passwd`, `/etc/shadow`, and `/etc/group` files.

Note – If you fail to add the `++` or `--` `netgroup` entries to `/etc/shadow` and `/etc/passwd`, you will not be able to log in.

5. **Save the file and reboot the system.**

Because some library routines do not periodically check the `nsswitch.conf` file to see whether it has been changed, you must reboot the machine to make sure those routines have the latest information in the file.

Setting up DNS Servers

Initializing the Server

To initialize a server, do the following.

▼ How to initialize the server

1. **Become superuser.**
2. **Install the `named.conf` configuration file and the required data files, as described in the previous sections.**
3. **Run `in.named`.**

```
#!/usr/sbin/in.named
```

Instead of running `in.named` from the command line, you can reboot.

Testing Your Installation

After your boot and data files are set up and `in.named` running, test your installation.

▼ How to Test Your Installation

1. **Become superuser.**
2. **Check your `syslog` file for error messages.**
See Chapter 6 for common DNS error messages and troubleshooting information.

3. Look up a host name in the local domain using the `nslookup` command.

```
dnsmaster% nslookup altair
Server:  dnsmaster.doc.com
Address: 192.146.168.5
Name:    altair.doc.com
Address: 192.146.168.10
```

- If your lookup is successful, your name server is probably functioning correctly.
- If you get a Can't find, or can't initialize address, type of message for your server, or a non-existent domain, type message, it might mean that your server is not correctly listed in the boot file or hosts files.
- If you get a can't find *name* or non-existent domain type of message, it might mean that the host you looked up is not in the server's `hosts` file, or the domain is incorrectly set in `resolv.conf`, or there is some other server problem.

4. Look up a remote domain name with `nslookup`.

If your network is connected to the Internet, look up the name of a remote domain. (If your network is not connected to the Internet, look up the name of a subdomain in another zone, if you have one.)

For example, to look up the name of the remote `internic.net` Internet domain, you would enter the following.

```
dnsmaster% nslookup internic.net

Server:  dnsmaster.doc.com
Address: 192.168.168.
Name:    internic.net
Addresses: 192.168.0.9, 192.168.0.6, 192.168.0.5, 192.168.0.8
```

- If you are successful, your name server is probably functioning correctly.
- If the above command does not find the remote domain name, one possible cause is that your network's connection to the Internet is not functioning properly.
- Another possible cause is that your `named.ca` file is not properly installed or set up.

The second time you use `nslookup` to find a domain, the answer will be returned as `non-authoritative`. This is normal because the answer is now coming from your cache, not the remote name server.

5. Look up a host name in your domain from a remote domain.

If your network is connected to the Internet, look up the name of a host in your domain from a remote domain. If your network is not connected to the Internet, look up the name of a host in your domain from another zone, if you have one.

For example, to look up the name of a host in your domain, from a remote Internet domain, you would enter two arguments after the `nslookup` command. The first argument is the name of the host for which you are searching, and the second argument is the name of the name server you are testing.

```
remotemachine9% nslookup altair remotemaster.foo.org.
```

```
Server: remotemaster.foo.org
Address: 192.168.0.1
Name: altair.doc.com
Addresses: 192.168.1.2
```

- If you are successful, your name server is probably functioning correctly.
- If the above command does not find the machine you are searching for, one possible cause is that your domain is not properly registered with whomever is administering the parent domain (.com in the above example).

Adding Additional Servers

You can add primary and secondary DNS servers to your network.

▼ How to Add Additional Servers

1. **Become superuser.**
2. **Set up the server as a DNS client. See “Adding a Client” on page 79.**
3. **Set up the following files.**

```
boot file
named.ca
hosts
hosts.rev
named.local
```

See “Setting up DNS Servers” on page 75 for details.

Modifying DNS Data Files

Whenever you add or delete a host or make some other change in one of the DNS data files in the master DNS server or otherwise modify DNS data files, you must also do the following.

- Change the serial number in the SOA resource record so the slave servers modify their data accordingly see “How to Change the SOA Serial Number” on page 78.
- Inform `in.named` on the master server that it should reread the data files and update its internal database. See “Forcing `in.named` to Reload DNS Data” on page 78.

How to Change the SOA Serial Number

Every DNS database file begins with a Start of Authority (SOA) resource record. Whenever you alter any data in a DNS database file, you must increment the SOA serial number by one integer.

For example, if the current SOA Serial Number in a data file is 101, and you make a change to the file's data, you must change 101 to 102. If you fail to change the SOA serial number, the domain's slave servers will not update their copy of the database files with the new information and the master and slave servers will become out of sync.

A typical SOA record of a sample `hosts` file looks like the following.

```
; sample hosts file
@      IN      SOA      nismaster.doc.com. root.nismaster.doc.com. (
        109 ; Serial
        10800 ; Refresh
                1800 ; Retry
        3600000 ; Expire
        86400 ) ; Minimum
```

Thus, if you made a change to this `hosts` file, you would change 109 to 110. The next time you change the file, you would change 110 to 111.

Forcing `in.named` to Reload DNS Data

When `in.named` successfully starts, the daemon writes its process ID to the file `/etc/named.pid`. To have `in.named` reread `named.conf` and reload the database do the following.

▼ How to force `in.named` to reload DNS data.

1. **Become superuser.**
2. `# kill -HUP `cat /etc/named.pid``

This will eliminate all previously cache, and the caching process will start over again.



Caution – Do not attempt to run `in.named` from `inetd`. This will continuously restart the name server and defeat the purpose of having a cache.

Adding and Deleting Clients

When you add or delete a client, always make your changes in the data files stored on your master DNS server. Do not make changes or edit the files on your slave servers because those will be automatically updated from the master server based on your changing the SOA serial number.

Adding a Client

To add a client to a DNS domain, you set the new machine up as a DNS client and then add records for the new machine to the appropriate `hosts` and `hosts.rev` files.

For example, to add the host `rigel` to the `doc.com` domain, do the following.

▼ How to Add a Client

1. **Become superuser.**
2. **Create a `/etc/resolv.conf` file on `rigel`.**
3. **Add `dns` to the `hosts` line of `rigel`'s `/etc/nsswitch.conf` file**
See "DNS and Internet Access" on page 44.
4. **Add an address (A) record for `rigel` to the master server's `hosts` file.**

```
rigel IN A 192.168.112
```
5. **Add any additional optional records for `rigel` to the master server's `hosts` file.**
Optional records could include the following.
 - Alias (CNAME)
 - Mail exchange (MX)
 - Well known services (WKS)
 - Host information (HINFO)
6. **Add a PTR record for `rigel` to the `hosts.rev` file.**
7. **Increment the SOA serial number in the master server's `hosts` and `hosts.rev` files.**
8. **Reload the server's data.**
Either reboot the server or type the following.

```
# kill -HUP `cat /etc/named.pid`
```

Removing a Client

To remove a client from a DNS domain do the following.

▼ How to Remove a Client

1. **Become superuser.**
2. **Remove `dns` from the `hosts` line of the machine's `nsswitch.conf` file.**
3. **Remove the machine's `/etc/resolv.conf` file.**
4. **Delete the records for that machine from the master server's `hosts` and `hosts.rev` files.**
5. **If the machine has `CNAME` records pointing to it, those `CNAME` records must also be deleted from the `hosts` file.**
6. **Set up replacements for services supported by the removed machine.**
If the machine is a master server, mail host, or host for any other necessary process or service, you must take whatever steps are necessary to set up some other machine to perform those services.

Enabling a Client to Use IPv6

TABLE 4-1 Enabling a Machine to Use IPv6

Task	Description	For Instructions, Go To
Enabling a Machine to Use IPv6	Modify the <code>/etc/nsswitch.conf</code> file and enable an NIS+ client to use IPv6	See below.

▼ How to Enable a Client to Use IPv6

1. **Become superuser.**
2. **Edit the `/etc/nsswitch.conf` file.**
3. **Add the new `ipnodes` source and specify the naming service, such as LDAP.**

```
ipnodes: ldap [NOTFOUND=return] files
```

`ipnodes` defaults to `files`. During the transition from IPv4 to IPv6, where all naming services are not aware of IPv6 addresses, you should accept the `files` default. Otherwise, unnecessary delays might result during the resolution of

addresses.

4. Save the file and reboot the machine.

Because the `nscd` daemon caches this information, which it reads at start up, you must reboot the machine now.

Creating DNS Subdomains

As your network grows you might find it convenient to divide it into one or more DNS subdomains. See “The DNS Namespace Hierarchy” on page 96 for a discussion of DNS domain hierarchy and structure.

When you divide your network into a parent domain and one or more subdomains, you reduce the load on individual DNS servers by distributing responsibility across multiple domains. In this way you can improve network performance. For example, suppose there are 900 machines on your network and all of them are in one domain. In this case, one set of DNS servers composed of a master and additional slave and caching-only servers have to support 900 machines. If you divide this network into a parent domain and two subdomain, each with 300 machines, then you have three sets of master and slave servers each responsible for only 300 machines.

By dividing your network into domains that match either your geographic or organizational structure (or both), the DNS domain names indicate where a given machine or email address fits into your structure. For example, `rigel@alameda.doc.com` implies that the machine `rigel` is located at your Alameda site, and the email address `barnum@sales.doc.com` implies that the user `barnum` is part of your Sales organization.

Dividing your network into multiple domains requires more set up work than keeping everything in one domain, and you have to maintain the delegation data that ties your domains together. On the other hand, when you have multiple domains, you can distribute domain maintenance tasks among different administrators or teams, one for each domain.

Planning Your Subdomains

The following are some points to consider before dividing your network into a parent and one or more subdomains.

- *Number of subdomains.* The more subdomains your create, the more initial setup work you have to do and the more ongoing coordination work for the administrators in the parent domain. The more subdomains, the more delegation work for the servers in the parent domain. On the other hand, fewer domains

mean larger domains, and the larger a domain is the more server speed and memory is required to support it.

- *Network divisions.* You can divide your network into multiple domains any way you want. The three most common methods are by organizational structure where you have separate subdomains for each department or division (sales, research, manufacturing, for example) by geography where you have separate subdomains for each site, or by network structure where you have separate subdomains for each major network component. The most important rule to remember is that administration and use will be easier if your domain structure follows a consistent, logical, and self-evident pattern.
- *Future considerations.* The most confusing domain structures are those that grow over time with subdomains added haphazardly as new sites and departments are created. To the degree possible, try to take future growth into account when designing your domain hierarchy. Also take into account stability. It is best to base your subdomains on what is most stable. For example, if your geographic sites are relatively stable but your departments and divisions are frequently reorganized, it is probably better to base your subdomains on geography rather than organizational function. On the other hand, if your structure is relatively stable but you frequently add or change sites, it is probably better to base your subdomains on your organizational hierarchy.
- *Wide area network links.* When a network spans multiple sites connected via modems or leased lines, performance will be better and reliability greater if your domains do not span such Wide Area Network (WAN) links. In most cases, WAN links are slower than contiguous network connections and more prone to failure. When servers have to support machines that can only be reached over a WAN link, you increase the network traffic funneling through the slower link, and if there is a power failure or other problem at one site, it could affect the machines at the other sites. (The same performance and reliability considerations apply to DNS zones. As a general rule of thumb, it is best if zones do not span WAN links.)
- *NIS+ compatibility.* If your enterprise-level naming service is NIS+, administration will be easier if your DNS and NIS+ domain and subdomain structures match.
- *Subdomain names.* To the degree possible, it is best to establish and follow a consistent policy for naming your subdomains. When domain names are consistent, it is much easier for users to remember and correctly specify them. Note that domain names are an important element in all of your DNS data files and that changing a subdomain name requires editing every file in which the old name appears. Thus, it is best to choose subdomain names that are stable and unlikely to need changing. You can use either full words, such as *manufacturing*, or abbreviations, such as *manf*, as subdomain names, but it will confuse users if some subdomains are named with abbreviations and others with full names. If you decide to use abbreviations, use enough letters to clearly identify the name because short cryptic names are hard to use and remember. Do not use reserved top-level Internet domain names as subdomain names. This means that names like *org*, *net*, *com*, *gov*, *edu*, and any of the two-letter country codes such as *jp*, *uk*, *ca*, and *it* should never be used as a subdomain name.

Setting Up a Subdomain

In most cases, new subdomains are usually created from the start with a new network and machines, or split off from an existing domain. The process is essentially similar in both cases.

Once you have planned your new subdomain, do the following.

▼ How to Set up a Subdomain

1. **Make sure all of the machines in the new subdomain are properly set up as DNS clients.**

If you are carving a new subdomain out of an existing domain, most of the machines are probably already set up of DNS clients. If you are building a new subdomain from scratch or adding new machines to an existing network, you must install properly configured `resolv.conf` and `nsswitch.conf` files on each machine.

2. **Install properly configured boot and DNS data files on the subdomain's master server.**

- `/etc/named.conf`.
- `/var/named/named.ca`.
- `/var/named/hosts`.
- `/var/named/hosts.rev`.
- `/var/named/named.local`.

Note that the server host files must have an Address (A) record, any necessary CNAME records for each machine in the subdomain and the server `hosts.rev` files must have a pointer (PTR) record for each machine in the subdomain. Optional HINFO and WKS records can also be added.

3. **If you are splitting an existing domain, remove the records for the machines in the new subdomain from the parent domain's master server `hosts` and `hosts.rev` files.**

You must delete the A records for the machines that are now in the new subdomain from the hosts files of the old domain's servers. You must also delete the PTR records for those machines from the old domain's `hosts.rev` files. Any optional HINFO and WKS records for the moved machines should also be deleted.

4. **If you are splitting an existing domain, add the new subdomain name to CNAME records in the parent domain's master server `hosts` and file.**

For example, suppose you are using the machine `aldebaran` as a fax server and it had the following CNAME record in the hosts file of the parent domain's servers.

```
faxserver IN CNAME aldebaran
```

In addition to creating a new `faxserver` CNAME record for `aldebaran` in the hosts file of the new subdomain's master server, you would also have to change this CNAME record in the parent domain's hosts file to include `aldebaran`'s subdomain as shown below:

```
faxserver    IN    CNAME    aldebaran.manf.doc.com
```

5. Add NS records for the new subdomain's servers to the parent domain's hosts file.

For example, suppose that your parent domain is `doc.com` and you are creating a new `manf.doc.com` subdomain with the machine `rigel` as `manf`'s master server and `aldebaran` as the slave server. You would add the following records to the `hosts` file of `doc.com`'s master server.

```
manf.doc.com 99999 IN NS rigel.manf.doc.com
              99999 IN NS aldebaran.manf.doc.com
```

6. Add A records for the new subdomain's servers to the parent domain's hosts file.

Continuing with the above example, you would add the following records to the `hosts` file of `doc.com`'s master server.

```
rigel.manf.doc.com      99999 IN A 1.22.333.121
aldebaran.manf.doc.com  99999 IN A 1.22.333.136
```

7. Start up `named` on the subdomain's servers.

```
# /usr/sbin/in.named
```

Instead of running `in.named` from the command line, reboot. See "in.named and DNS Name Servers" on page 53.

Solaris DNS BIND 8.2.4 Implementation

For your convenience, the Solaris operating environment supplies a compiled version of Berkeley Internet Name Domain (BIND) version 8.2.4. In compiling this software, options and choices were made to meet the needs of the greatest number of sites. If this pre-compiled version of BIND does not meet your requirements, you can recompile your own version of BIND from the publicly available source code.

In compiling the BIND version supplied with the Solaris operating environment the following choices were made.

- *RFC1535*. Not implemented because doing so would remove implicit search lists.
- *Inverse Queries*. Enabled because SunOS 4 `nslookup` will not work without them.
- *Default Domain Name*. If the DNS domain name is not set in `/etc/resolv.conf`, or via the `LOCALDOMAIN` environment variable, `libresolv` derives it from the NIS or NIS+ domain name.
- *Utility Scripts*. The BIND utility scripts are not included in this Solaris release.
- *Test Programs*. The BIND test programs `dig`, `dnsquery`, and `host` are not included in this Solaris release because their purpose is similar to that of `nslookup` and `nstest`.

▼ How to Migrate from BIND 4.9.x to BIND 8.2.4

1. Become superuser.
2. Run the Korn shell script, `/usr/sbin/named-bootconf`, to convert a BIND 4.9.x `named.boot` file to a BIND 8.2.4 `named.conf` file.

Note – In Solaris 9, the `named.boot` is ignored.

DNS Forwarding

The `nsswitch.conf` file controls DNS forwarding and Interent access for clients. NIS clients have implicit forwarding capabilities. NIS+ clients do not. See below.

▼ How to Enable DNS Forwarding Capabilities on an NIS+ Client

1. Become superuser.
2. Properly configure the `hosts` line in the `/etc/resolve.conf` file to read:
`hosts:nisplus dns files.`

In this implementation of NIS, if a `/etc/resolve.conf` file exists on the server, `ypstart` *automatically* starts the `ypserv` daemon with the `-d` option to forward requests to DNS. To stop forwarding to DNS, edit the `/usr/lib/netsvc/yp/ypstart` script to remove the `-d` option from the `ypserv` command. You must then reboot the machine.

▼ How to enable DNS forwarding capabilities on an older NIS client

1. Become superuser.
2. Set the `YP_INTERDOMAIN` key in the `hosts.byname` map and the `hosts.byaddr` map by modify the following lines in the `Makefile` (at the top of the file) from the following.

```
#B=-b
B=
```

to

B=-b

#B=

Now `makedbm` starts with the `-b` flag when making the maps, and inserts the `YP_INTERDOMAIN` into the `ndbm` files.

3. Rebuild the maps.

```
# /usr/ccs/bin/make hosts
```

4. Make sure that all NIS servers have an `/etc/resolv.conf` file that points to valid name server(s).

5. Stop each server with the `ypstop` command.

```
# /usr/lib/netsvc/yp/ypstop
```

6. Restart each server with the `ypstart` command.

```
# /usr/lib/netsvc/yp/ypstart
```

Note – If you have NIS servers that are not running Solaris 2 or higher, make sure that the `YP_INTERDOMAIN` key is present in the host maps. In addition, problems might arise if the master server and slave server are running *different* versions of Solaris. The following table summarizes the commands to issue to avoid such problems. The notation “4.0.3+” means “release 4.0.3 of SunOS or later.” The command `makedbm -b` is a reference to the “-B” variable in the `Makefile`.

TABLE 4-2 NIS/DNS in Heterogeneous NIS Domains

SLAVE	MASTER		
	4.0.3+		Solaris NIS
4.0.3+	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>makedbm -b</code> Slave: <code>ypxfr -b</code>	Master: <code>ypserv -d</code> Slave: <code>ypxfr -b</code>
Solaris NIS	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>ypserv -d</code> Slave: <code>ypxfr</code> with <code>resolve.conf</code> or <code>ypxfr -b</code>

The Solaris operating environment includes the dynamic library routines that make up the resolver.

DNS Administration (Reference)

This chapter covers the following topics.

- “Implementing DNS” on page 87
- “Setting Up the Data Files” on page 93
- “Setting Up Subdomains” on page 94
- “The DNS Namespace Hierarchy” on page 96
- “How DNS Affects Mail Delivery” on page 97
- “DNS Configuration and Data Files” on page 97
- “Data File Resource Record Format” on page 106

Implementing DNS

A Practical Example

This section shows the files you need to implement DNS for a sample Internet-connected network, based on the examples used in this chapter.



Caution – The IP addresses and network numbers used in examples and code samples in this manual are for illustration purposes only. Do *not* use them as shown because they might have been assigned to an actual network or host.

This example assumes the following.

- An environment connected to the Internet
- Two networks, each with its own domain (`doc.com` and `sales.doc.com`) and its own DNS zone

- The doc.com domain and zone is the top zone over the sales.doc.com subdomain and zone
- Each network has its own network number

TABLE 5-1 Example Network Domain and Zone Configuration

Name and Zone	Number
doc.com	123.45.6
sales.doc.com	111.22.3

- Each zone has a master and one slave server, and the slave server of sales.doc.com is also the master server of doc.com:

TABLE 5-2 Example Network DNS Servers

Zone	Host Name	Function	Address	CNAME
doc.com	sirius	master for doc.com	123.45.6.1	dnsmaster
doc.com	deneb	slave for doc.com	111.22.3.5	dnssecond
sales.doc.com	altair	master for sales.doc.com	111.22.3.4	dnssales
sales.doc.com	altair	slave for sales.doc.com	123.45.6.1	dnsmaster

Example Boot Files

The following code examples show boot files for the three servers in the two networks:

EXAMPLE 5-1 Example Boot File for dnsmaster Server

```
; named.boot file on the dnsmaster (sirius)
;
; files required by in.named are located here
directory /var/named
; here are the names of the master files
cache . named.ca
master doc.com db.doc
master 0.0.127.in-addr.arpa named.local
master 6.45.123.in-addr.arpa doc.rev
;This system is also the slave for the sales.doc.com domain
slave sales.doc.com 111.22.3.4 db.sales
slave 3.22.111.in-addr.arpa 111.22.3.4 sales.rev
```

EXAMPLE 5-2 Example Boot File for dnssales Server

```
; named.boot file on the dnssales (altair)
;
; in.named is located here
```

EXAMPLE 5-2 Example Boot File for dnssales Server (Continued)

```
directory /var/named
; here are the names of the master files
cache      .                named.ca
master     sales.doc.com    db.sales
master     0.0.127.in-addr.arpa db.127.0.0
master     3.22.111.in-addr.arpa db.192.168.8
```

EXAMPLE 5-3 Example Boot File for dnssecond Server

```
; named.boot file on the dnsecond (deneb)
directory /var/named
cache      .                named.ca
slave      doc.com          123.45.6.1 doc.com
slave      6.45.123.in-addr.arpa 123.45.6.1 doc.123.45.6
```

Example resolv.conf Files

The following code examples show `resolv.conf` files for the three servers in the two networks. (If the host in question is not running `in.named`, the local host address should not be used as a name server.)

EXAMPLE 5-4 Example `resolve.conf` File for dnsmaster Server

```
;
; /etc/resolv.conf file for dnsmaster (sirius)
;
domain      doc.com
nameserver  0.0.0.0
nameserver  111.22.3.5
```

EXAMPLE 5-5 Example `resolve.conf` File for dnssales Server

```
;
; /etc/resolv.conf file for dnssales (altair)
;
domain      sales.doc.com
nameserver  111.22.3.4
nameserver  123.45.6.1
```

EXAMPLE 5-6 Example `resolve.conf` File for dnssecond Server

```
;
; /etc/resolv.conf for dnssecond
;
domain      doc.com
nameserver  111.22.3.5
nameserver  123.45.6.1
```

Example named.local File

The following code example shows the named.local file used by the two master servers on the two networks. Both servers have the same file.

EXAMPLE 5-7 Example named.local File for Both master Servers

```
; SOA rec
0.0.127.in-addr.arpa. IN SOA siriusdoc.com. sysop.centauri.doc.com. (
                        19970331    ; serial number
                        10800        ; refresh every 3 hours
                        10800        ; retry every 3 hours
                        604800       ; expire after a week
                        86400 )      ; TTL of 1 day

; Name Servers
0.0.127.in-addr.arpa. IN NS  sirius.doc.com.
0.0.127.in-addr.arpa. IN NS  dnssecond.doc.com
1 IN PTR localhost.
```

Example hosts Files

The following code examples show db.doc and db.sales files for the two master servers on the two networks.

EXAMPLE 5-8 Example db.doc File for dnsmastr server

```
; SOA rec
doc.com. IN SOA sirius.doc.com. sysop.centauri.doc.com. (
                        19970332    ; serial number
                        10800        ; refresh every 3 hours
                        10800        ; retry every 3 hours
                        604800       ; expire after a week
                        86400 )      ; TTL of 1 day

; Name Servers
doc.com.          IN NS  sirius.doc.com.
sales.doc.com.   IN NS  altair.sales.doc.com.

; Addresses
localhost        IN A  127.0.0.1
sirius           IN A  123.45.6.1
rigel           IN A  123.45.6.112
antares         IN A  123.45.6.90
polaris         IN A  123.45.6.101
procyon         IN A  123.45.6.79
tauceti         IN A  123.45.6.69
altair.sales.doc.com.  N A  111.22.3.4

; aliases
dnsmastr        IN CNAME  sirius.doc.com.
dnssecond.doc.com  IN CNAME  deneb.doc.com
```

EXAMPLE 5-9 Example db.sales File for dnssales server

```
; SOA rec
sales.doc.com. IN SOA altair.sales.doc.com. sysop.polaris.doc.com. (
                        19970332    ; serial number
```

EXAMPLE 5-9 Example db.sales File for dnssales server (Continued)

```

                                10800          ; refresh every 3 hours
                                10800          ; retry every 3 hours
                                604800         ; expire after a week
                                86400 )        ; TTL of 1 day
; Name Servers
doc.com.                        IN NS    sirius.doc.com.
sales.doc.com.                 IN NS    altair.sales.doc.com.
; Addresses
altair                          IN A    111.22.3.4
localhost                       IN A    127.0.0.1
sirius.doc.com.                IN A    123.45.6.1
luna                            IN A    192.168.8.22
phoebus                        IN A    192.168.8.24
deimos                         IN A    192.168.8.25
ganymede                       IN A    192.168.8.27
europa                         IN A    192.168.8.28
callisto                       IN A    192.168.8.29
;
; aliases
dnssales.sales.doc.com IN CNAME    altair.sales.doc.com
```

Example hosts.rev Files

The following code examples show hosts.rev files for the two master servers on the two networks:

EXAMPLE 5-10 Example doc.rev File for dnsmast r server

```

; SOA rec
6.45.123.in-addr.arpa. IN SOA sirius.doc.com. sysop.centauri.doc.com. (
                                19970331     ; serial number
                                10800         ; refresh every 3 hours
                                10800         ; retry every 3 hours
                                604800         ; expire after a week
                                86400 )        ; TTL of 1 day
; Name Servers
6.45.123.in-addr.arpa. IN NS    sirius.doc.com.
;Pointer records for 123.45.6
1                               IN PTR  sirius.doc.com.
112                             IN PTR  rigel.doc.com.
90                              IN PTR  antares.doc.com.
101                             IN PTR  polaris.doc.com.
79                              IN PTR  procyon.doc.com.
69                              IN PTR  tauceti.doc.com.
```

EXAMPLE 5-11 Example hosts.rev File for dnssales Server

```

; SOA rec
3.22.111.in-addr.arpa. IN SOA altair.sales.doc.com. \
sysop.polaris.doc.com. (
```

EXAMPLE 5-11 Example hosts.rev File for dnssales Server (Continued)

```

                                19970331 ; serial number
                                10800   ; refresh every 3 hours
                                10800   ; retry every 3 hours
                                604800  ; expire after a week
                                86400 ) ; TTL of 1 day
; Name Servers
3.22.111.in-addr.arpa. IN NS altair.sales.doc.com.; \
Pointer records for 111.22.3
22          IN PTR luna
23          IN PTR deneb
24          IN PTR phoebus
25          IN PTR deimos
26          IN PTR altair
27          IN PTR ganymede
28          IN PTR europa
29          IN PTR callisto
```

Example name.ca File

The following code example shows the named.ca file that is stored on each of the two master servers on the two networks. Both servers use identical named.ca files.

EXAMPLE 5-12 Example named.ca File

```

;
; formerly NS1.ISI.EDU
.          3600000   NS      B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000   A      128.9.0.107
;
; formerly C.PSI.NET
.          3600000   NS      C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000   A      192.33.4.12
;
; formerly TERP.UMD.EDU
.          3600000   NS      D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET. 3600000   A      128.8.10.90
;
; formerly NS.NASA.GOV
;.         3600000   NS      E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET. 3600000   A      192.203.230.10
;
; formerly NS.ISC.ORG
.          3600000   NS      F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET. 3600000   A      192.5.5.241
;
; formerly NS.NIC.DDN.MIL
.          3600000   NS      G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET. 3600000   A      192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
```

EXAMPLE 5-12 Example named .ca File (Continued)

```
.                3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET. 3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
.                3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET. 3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
.                3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET. 3600000      A     198.41.0.10
;
; temporarily housed at NSI (InterNIC)
.                3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET. 3600000      A     198.41.0.11
;
; temporarily housed at ISI (IANA)
.                3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET. 3600000      A     198.32.64.12
;
; temporarily housed at ISI (IANA)
.                3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET. 3600000      A     198.32.65.12
; End of File
```

Setting Up the Data Files

All the data files used by the DNS daemon in `.named` are written in standard resource record format. Each line of a file is a record, called a resource record (RR). Each DNS data file must contain certain resource records.

Resource Record Types

The most commonly used types of resource records are listed in Table 5-7. They are usually entered in the order shown in Table 5-7, but that is not a requirement.

TABLE 5-3 Commonly Used Resource Record Types

Type	Description
SOA	Start of authority
NS	Name server

TABLE 5-3 Commonly Used Resource Record Types (Continued)

Type	Description
A	IPv4 Internet address (name to address)
AAAA	IPv6 Internet address (name to address)
PTR	Pointer (address to name)
CNAME	Canonical name (nickname)
TXT	Text information
MX	Mail exchanger

In the sample files included in the following sections, @ indicates the current zone or origin and lines that begin with a semicolon (;) are comments.

Setting Up Subdomains

Setting Up Subdomains—Same Zone

The simplest method is to include the subdomain in the parent domain's zone. In this way, one set of DNS servers and data files applies to all the machines regardless of their domain.

The advantage of the same-zone method is simplicity and ease of administration. The disadvantage is that one set of servers has to serve all machines in all of the zone's domains. If there are too many machines, the servers will be overloaded and network performance can decline.

Data files for multi-domain zones must include records for all machines and servers in each domain covered by the zone.

Setting up a multi-domain zone is the same as setting up a zone with a single domain, except that fully qualified domain names are used in the `hosts` file to identify machines in remote domains. In other words, in the `hosts` file, when you identify a machine in the server's local domain, you need to use only the machine's name. But when you identify a machine in some other domain, you must identify the machine with a fully qualified domain name in the format: *machine.domain*.

Server and machine names in `hosts.rev` and `named.local` files also need to be fully qualified with domain names. But that is true regardless of whether or not the zone has more than one domain.

Setting Up Subdomains—Different Zones

The advantage of the different-zone method is that you can assign different sets of servers to serve machines in different domains; in that way, you spread out server load so that no group of servers is overloaded. The disadvantage is that setup maintenance is more complicated.

Setting up subdomains that are in different zones is more complicated than including multiple domains in a single zone, because you have to specify how clients in different zones obtain DNS information from the other zones.

To divide a network into multiple domains, create a domain hierarchy. That is, one domain becomes the top domain. Beneath the top domain, you create one or more subdomains. If you want, you can create subdomains of subdomains. But every subdomain has a set place relative to the top domain in the hierarchy of domains. When read from left to right, domain names identify the domain's place in the hierarchy. For example, the `doc.com` domain is above the `sales.doc.com` domain, while the `west.sales.doc.com` domain is below the `sales.doc.com` domain.

DNS zones acquire a hierarchy from the domains that they contain. The zone containing a network's top domain is the top zone. A zone that contains one or more subdomains below the top domain is below the top zone in the zone hierarchy. When DNS information is passed from one zone to another, it is passed up and down the zone hierarchy. This means that each zone requires records in its data files that specify how to pass information up to the zone immediately above it, and down to any zones immediately below it.

To correctly transfer DNS information from one zone to another in a multi-zone network:

- `hosts.rev` file. There must be a PTR record in each `hosts.rev` file pointing to the name of one or more master servers in the zone immediately above it. This type of PTR record is exactly the same as any other PTR record in the file, except that it identifies a server in the zone above.
- `hosts` file NS records. There must be a zone NS record in each `hosts` file identifying each name server in each zone immediately below. This type of NS record requires the name of the zone below as the first field in the NS record. (The name of the zone is specified in the SOA record of the zone's host file.)
- `hosts` file A records. There must be an A record in each `hosts` file identifying the IP address of each name server in each zone immediately below. This type of A record has to have the name of the zone below as the first field in the A record. (The name of the zone is specified in the SOA record of the zone's host file.)

The example files in the next chapter illustrate a network with two zones.

The DNS Namespace Hierarchy

The entire collection of DNS administrative domains throughout the world are organized in a hierarchy called the *DNS namespace*. This section shows how the namespace organization affects both local domains and the Internet.

Domains and Subdomains

Like the UNIX™ file system, DNS domains are organized as a set of descending branches similar to the roots of a tree. Each branch is a domain, each subbranch is a *subdomain*. The terms *domain* and *subdomain* are relative. A given domain is a subdomain relative to those domains above it in the hierarchy, and a parent domain to the subdomains below it.

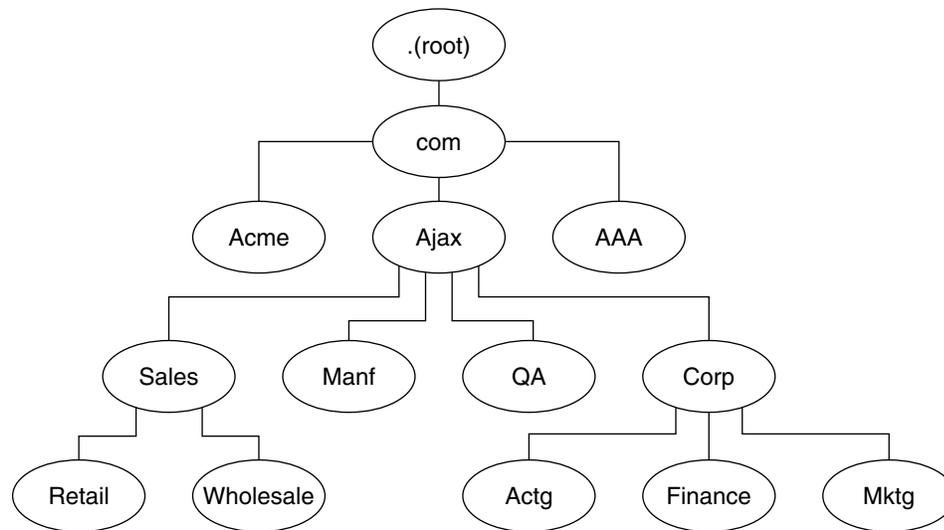


FIGURE 5-1 Domains and Subdomains

For example, in Figure 5-1, *com* is a parent domain to the *Acme*, *Ajax*, and *AAA* domains. Or you could just as easily say that those are subdomains relative to the *com* domain. In its turn, the *Ajax* domain is a parent to four subdomains (*Sales*, *Manf*, *QA*, and *Corp*).

A domain contains one parent (or top) domain plus the associated subdomains if any. Domains are named up the tree starting with the lowest (deepest) subdomain and ending with the root domain.

How DNS Affects Mail Delivery

In addition address mapping and maps addresses to host names, as discussed in “Name-to-Address Resolution” on page 50, DNS also helps mail delivery agents, such as `sendmail` and POP, deliver mail along the Internet.

To deliver mail across the Internet, DNS uses *mail exchange records* (MX records). Most organizations do not allow direct delivery of mail that comes across the Internet for hosts within the organization. Instead, they use a central mail host (or a set of mail hosts) to intercept incoming mail messages and route them to their recipients.

The mail exchange record identifies the mail host that services each machine in a domain. Therefore, a mail exchange record lists the DNS domain names of remote organizations and either the IP address or the host name of its corresponding mail host.

DNS Configuration and Data Files

In addition to the `in.named` daemon, DNS on a name server consists of a boot file called `named.conf`, a resolver file named `resolv.conf`, and four types of zone data files.

Names of DNS Data Files

So long as you are internally consistent, you can name the zone data files anything you want. This flexibility can lead to some confusion when working at different sites or referring to different DNS manuals and books.

For example, the file names used in Sun manuals and at most many Solaris sites vary from those used in the book *DNS and BIND* by Albitz and Liu, O’Reilly & Associates, 1992, and both of those nomenclatures have some differences from that used in the public-domain *Name Server Operations Guide for BIND*, University of California.

In addition, this manual and other DNS documentation uses generic names that identify a file’s main purpose, and specific example names for that file in code record samples. For example, Solaris Naming manuals use the generic name `hosts` when describing the function and role of that file, and the example names `db.doc` and `db.sales.doc` in code samples.

For reference purposes, the following table compares BIND file names from these three sources.

TABLE 5-4 BIND File Name Examples

Solaris Names	O'Reilly Names or other names	U.C. Berkeley Names	Content and Purpose of File
/etc/named.conf	/etc/named.conf	/etc/named.conf	The configuration file specifies the type of server it is running on and the zones that it serves as a 'Master', 'Slave', or 'Stub'. It also defines security, logging, and a finer granularity of options applied to zones.
/etc/resolv.conf	/etc/resolv.conf	/etc/resolv.conf	This file resides on every DNS client (including DNS servers) and designates the servers that the client queries for DNS information.
named.ca	db.cache db.root	root.cache	This file establishes the names of root servers and lists their addresses.
Generic: hosts Examples: db.doc db.sales	Generic: db.domain Examples: db.movie db.fx	Generic: hosts Example: ucbhosts	This file contains all the data about the machines in the local zone that the server serves.
Generic: hosts.rev Examples: doc.rev	Generic: db.ADDR Examples: db.192.249.249 db.192.249.253	hosts.rev	This file specifies a zone in the in-addr.arpa domain, a special domain that allows reverse (address-to-name) mapping.
named.local	Generic: db.cache Example: db.127.0.0	named.local	This file specifies the address for the local loopback interface, or localhost
\$INCLUDE files	\$INCLUDE files	\$INCLUDE files	Any file identified by an \$INCLUDE () statement in a data file.



Caution – The IP addresses and network numbers used in examples and code samples in this manual are for illustration purposes only. Do *not* use them as shown because they might have been assigned to an actual network or host.

The named.conf File

The BIND 8.2.4 configuration file, /etc/named.conf establishes the server as a master, slave, or cache-only name server. It also specifies the zones over which the server has authority and which data files it should read to get its initial data.

The /etc/named.conf file contains statements that implement the following.

- Security through an Access Control List (ACL) that defines a collection of IP addresses that an NIS+ host has read/write access
- Logging specifications
- Selectively applied options for a set of zones, rather than to all zones

The configuration file is read by `in.named` when the daemon is started by the server's start up script, `/etc/init.d/inetsvc`. The configuration file directs `in.named` either to other servers or to local data files for a specified domain.

named.conf Statements

The `named.conf` file contains statements and comments. Statements end with a semicolon. Some statements can contain a block of statements. Again, each statement in the block is terminated with a semicolon.

The `named.conf` file supports the following statements.

TABLE 5-5 `named.conf` Statements

<code>acl</code>	Defines a named IP address match list used for access control. The address match list designates one or more IP addresses (dotted-decimal notation) or IP prefixes (dotted-decimal notation followed with a slash and the number of bits in the netmask). The named IP address match list must be defined by an <code>acl</code> statement before it can be used elsewhere; no forward references allowed.
<code>include</code>	Inserts an include file at the point where the <code>include</code> statement is encountered. Use <code>include</code> to break up the configuration into more easily managed chunks.
<code>key</code>	Specifies a key ID used for authentication and authorization on a particular name server. See the <code>server</code> statement.
<code>logging</code>	Specifies the information the server logs and the destination of log messages.
<code>options</code>	Controls global server configuration options and sets default values for other statements.
<code>server</code>	Sets designated configuration options associated with a remote name server. Selectively applies options on a per-server basis, rather than to all servers.
<code>zone</code>	Defines a zone. Selectively applies options on a per-zone basis, rather than to all zones.

EXAMPLE 5-13 Example Master Configuration File for a master server

```
options {
    directory "/var/named";
    datasize 2098;
```

EXAMPLE 5-13 Example Master Configuration File for a master server (Continued)

```
        forward only;
        forwarders {
            99.11.33.44;
        };
        recursion no;
        transfers-in 10;
        transfers-per-ns 2;
        allow-transfer {
            127.0.1.1/24;
        };
};

logging {
    category queries { default_syslog; };
};

include "/var/named/abcZones.conf"

// here are the names of the master files
zone "cities.zn" {
    type master;
    file "db.cities.zn";
};

zone "0.0.127.in-addr.arpa." {
    type master;
    file "db.127.cities.zn";
};

zone "168.192.in-addr.arpa" {
    type master;
    file "db.cities.zn.rev";
};

zone "sales.doc.com" {
    type slave;
    file "slave/db.sales.doc";
    masters {
        192.168.1.151;
    };
};

zone "168.192.in-addr.arpa" {
    type slave;
    file "slave/db.sales.doc.rev";
    masters {
        192.168.1.151;
    };
};
```

The named . ca File

The named . ca file establishes the names of root servers and lists their addresses. If your network is connected to the Internet, named . ca lists the Internet name servers; otherwise, it lists the root domain name servers for your local network. The in . named daemon cycles through the list of servers until it contacts one of them. It then obtains from that server the current list of root servers, which it uses to update named . ca.

Setting Up the named . ca File

Root server names are indicated in the NS record and addresses in the A record. You need to add an NS record and an A record for each root server you want to include in the file.

How you obtain or create your named . ca file depends on whether or not your network is connected to the world Internet.

Internet named . ca File

If your network is connected to the Internet, at the present time you obtain your named . ca file from InterNIC registration services through the following.

- Anonymous FTP. The FTP site is: ftp . rs . internic . net. The file name is: /domain/named . root.
- Gopher. The Gopher site is: rs . internic . net. The file is: named . root, which can be found under the InterNIC Registration Services menu, InterNIC Registration Archives submenu.

If you are following the naming conventions used in this manual, you then move named . root to /var/named/named . ca.

EXAMPLE 5-14 Example Internet named . ca file

```
;
; formerly NS1.ISI.EDU
.           3600000    NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000    A    128.9.0.107
;
; formerly C.PSI.NET
.           3600000    NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000    A    192.33.4.12
;
; formerly TERP.UMD.EDU
.           3600000    NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET. 3600000    A    128.8.10.90
;
; formerly NS.NASA.GOV
;.          3600000    NS    E.ROOT-SERVERS.NET.
```

EXAMPLE 5-14 Example Internet named .ca file (Continued)

```
E.ROOT-SERVERS.NET.      3600000    A    192.203.230.10
;
; formerly NS.ISC.ORG
.                          3600000    NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000    A    192.5.5.241
;
; formerly NS.NIC.DDN.MIL
.                          3600000    NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000    A    192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
.                          3600000    NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000    A    128.63.2.53
;
; formerly NIC.NORDU.NET
.                          3600000    NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000    A    192.36.148.17
;
; temporarily housed at NSI (InterNIC)
.                          3600000    NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000    A    198.41.0.10
;
; temporarily housed at NSI (InterNIC)
.                          3600000    NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000    A    198.41.0.11
;
; temporarily housed at ISI (IANA)
.                          3600000    NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000    A    198.32.64.12
;
; temporarily housed at ISI (IANA)
.                          3600000    NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000    A    198.32.65.12
; End of File
```

Non-Internet named .ca File

If your network is not connected to the Internet, you create your own named .ca file. To do this, you designate one of your servers to be the root server, then create a named .ca file on every DNS server pointing to that root server.

For example, suppose your domain is named `private` and you designate the machine `ourroot` as your non-Internet root server. The `ourroot` machine has an IP address of `192.1.1.10`. Your named .ca files would then contain the line:

```
ourroot.private. 999999 IN A 192.1.1.10
```

Cache files also need an SOA record, NS records for each domain and subdomain, and A records for each server.

For example, suppose that in addition to ourroot you also had DNS name servers called ourmaster and ourslave. The named .ca files on all of your DNS servers would then look like the following.

EXAMPLE 5-15 Sample named .ca File (Non-Internet)

```
;
@    IN    SOA  ourroot.private.  hermit.ourroot.private (
        1997071401      ; serial number (YYYYMMDD##)
        10800           ; refresh after 3 hours
        3600            ; retry after 1 hour
        604800          ; expire after 1 week
        86400 )         ; minimum TTL of 1 day
;
ourroot.private.      999999  IN    A    192.1.1.10
;
private.              IN    NS   ourmaster.private.
1.1.192.in-addr.arpa IN    NS   ourmaster.private.

ourprivate.private.  IN    A    192.1.1.1
;
private.              IN    NS   ourslave.private.
1.1.192.in-addr.arpa IN    NS   ourslave.private.
ourslave.private.    IN    A    192.1.1.2
```

The hosts File

The hosts file contains all the data about the machines in the local zone. The name of this file is specified in the boot file. To avoid confusion with /etc/hosts, name the file something other than hosts, for example, you could name these files using the pattern db.domain. Using that nomenclature, the host files for the doc.com and sales.doc.com domains might be db.doc and db.sales.

Setting Up the hosts File

The hosts file contains all the data about every machine in your zone. If a zone covers more than one domain, all machines in all the domains covered by the zone are listed in the zone's host file. See "Setting Up the hosts File" on page 103.

Note – The name hosts is a generic name indicating the file's purpose and content. But to avoid confusion with /etc/hosts, you should name this file something other than hosts. If you have more than one zone, each zone must have its own hosts file and each of these zone hosts files must have a unique name. For example, if your DNS domain is divided into doc.com and sales.doc.com zones, you could name one hosts file db.doc and the other sales.db.doc.

There must be a separate, uniquely named, `hosts` file for each zone. If you have more than one zone, each zone's host file must include information about the master (master and slave) servers of the other zones, as described in Example 5–16.

EXAMPLE 5–16 Sample `hosts` File

```
;  
; SOA rec  
doc.com. IN SOA sirius.doc.com. sysop.centauri.doc.com. (  
          1997071401      ; serial number (YYYYMMDD##)  
          10800           ; refresh every 3 hours  
          10800           ; retry every 3 hours  
          604800          ; expire after a week  
          86400 )         ; TTL of 1 day  
  
; Name Servers  
doc.com.           IN NS  sirius.doc.com.  
sales.doc.com.    IN NS  altair.sales.doc.com.  
; Addresses  
localhost.        IN A   127.0.0.1  
  
sirius             IN A   192.168.6.1  
rigel              IN A   192.168.6.112  
antares            IN A   192.168.6.90  
polaris            IN A   192.168.6.101  
procyon            IN A   192.168.6.79  
tauceti            IN A   123.45.6.69  
altair.sales.doc.com. IN A   111.22.3.4  
; aliases  
durvasa            IN CNAME sirius.doc.com.  
dnsmastr           IN CNAME sirius.doc.com.  
dnssales           IN CNAME altair.sales.doc.com.
```

A `hosts` file usually contains these elements:

- A Start of Authority (SOA) record
- One or more Name Server (NS) records identifying master and slave DNS name servers
- Address (A) records for each host in the zone
- Canonical Name (CNAME) records for each host alias in the zone
- One or more Mail Exchange (MX) records

The `hosts.rev` File

The `hosts.rev` file specifies a zone in the `in-addr.arpa` domain, the special domain that allows reverse (address-to-name) mapping. The name of this file is specified in the boot file.

Setting Up the `hosts.rev` File

The `hosts.rev` file sets up inverse mapping.

Note – The name `hosts.rev` is a generic name indicating the file’s purpose and content. If you have more than one zone, each zone must have its own `hosts.rev` file and each of these zone `hosts.rev` files must have a unique name. For example, if your DNS domain is divided into `doc.com` and `sales.doc.com` zones, you could name one `hosts.rev` file `doc.rev` and the other `sales.rev`.

EXAMPLE 5-17 Sample `hosts.rev` File

```
; SOA rec
6.45.123.in-addr.arpa. IN SOA sirius.doc.com. sysop.centauri.doc.com. (
                        1997071401      ; serial number (YYYYMMDD##)
                        10800           ; refresh every 3 hours
                        10800           ; retry every 3 hours
                        604800          ; expire after a week
                        86400 )         ; TTL of 1 day
; Name Servers
6.45.123.in-addr.arpa. IN NS  sirius.doc.com.
1                       IN PTR sirius.doc.com.
```

A `hosts.rev` file contains the following elements.

- A Start of Authority (SOA) record
- One or more Name Server (NS) records identifying master and slave DNS name servers. Server names should be fully qualified.
- A PTR record for each host in the zone. Machine names should be fully qualified.

(See “Resource Record Types” on page 93 for detailed descriptions of these resource record types.)

The `named.local` File

The `named.local` file specifies the address for the local loopback interface, or `localhost`, with the network address `127.0.0.1`. The name of this file is specified in the boot file. Like other files, you can give it a name other than the name used in this manual.

Setting Up the `named.local` File

The `named.local` file sets up the local loopback interface for your name server.

EXAMPLE 5-18 Sample named.localFile

```
; SOA rec
0.0.127.in-addr.arpa. IN SOA sirius.doc.com sysop.centauri.doc.com (
    1997071401      ; serial number (YYYYMMDD##)
    10800          ; refresh every 3 hours
    10800          ; retry every 3 hours
    604800         ; expire after a week
    86400 )         ; TTL of 1 day

; Name Servers
0.0.127.in-addr.arpa. IN NS      sirius.doc.com
1                      IN PTR    localhost.
```

A named.local file contains these elements:

- A Start of Authority (SOA) record, which indicates the start of a zone and includes the name of the host on which the named.local data file reside.
- One or more Name Server (NS) records identifying master and slave DNS name servers. Server and domain names should be fully qualified.
- A PTR record for localhost

\$INCLUDE Files

An include file is any file named in an \$INCLUDE() statement in a DNS data file. \$INCLUDE files can be used to separate different types of data into multiple files for your convenience.

For example, suppose a data file contained following line:

```
$INCLUDE /etc/named/data/mailboxes
```

This line causes the /etc/named/data/mailboxes file to be loaded at that point. In this instance, /etc/named/data/mailboxes is an \$INCLUDE file. Use of \$INCLUDE files is optional. You can use as many as you wish, or none at all.

Data File Resource Record Format

All the data files used by the DNS daemon in.named are written in standard resource record format. Each DNS data file must contain certain resource records. This section describes the DNS data files and the resource records each file should contain.

Standard Resource Record Format

In the standard resource record format, each line of a data file is called a *resource record* (RR), which contains the following fields separated by white space:

```
namettlclassrecord-typerecord-specific-data
```

The order of the fields is always the same; however, the first two are optional (as indicated by the brackets), and the contents of the last vary according to the *record-type* field.

The *name* Field

The first field is the name of the domain that applies to the record. If this field is left blank in a given RR, it defaults to the name of the previous RR.

A domain name in a zone file can be either a fully qualified name, terminated with a dot, or a relative name, in which case the current domain is appended to it.

The *tll* Field

The second field is an optional time-to-live field. This specifies how long (in seconds) this data will be cached in the database before it is disregarded and new information is requested from a server. By leaving this field blank, the *tll* defaults to the minimum time specified in the Start-Of-Authority (SOA) resource record.

If the *tll* value is set too low, the server will incur a lot of repeat requests for data refreshment; if, on the other hand, the *tll* value is set too high, changes in the information will not be timely distributed.

Most *tll* values should be initially set to between a day (86400) and a week (604800). Then, depending on the frequency of actual change of the information, you can change the appropriate *tll* values to reflect that frequency. Also, if you have some *tll* values that have very high numbers because you know they relate to data that rarely changes. When you know that the data is now about to change, reset the *tll* to a low value (3600 to 86400) until the change takes place. Then change it back to the original high value.

All RR's with the same name, class, and type should have the same *tll* value.

The *class* Field

The third field is the record *class*. Only one *class* is currently in use: IN for the TCP/IP protocol family.

The *record-type* Field

The fourth field states the resource record *type*. There are many types of RR's; the most commonly used types are discussed in "Resource Record Types" on page 93.

The *record-specific-data* Field

The contents of the *record-specific-data* field depend on the type of the particular resource record.

Although case is preserved in names and data fields when loaded into the name server, all comparisons and lookups in the name server database are case insensitive. However, this situation might change in the future; thus, you should be consistent in your use of lower and uppercase.

Special Resource Record Characters

The following characters have special meanings:

TABLE 5-6 Special Resource Record Characters

Character	Definition
.	A free-standing dot in the name field refers to the current domain.
@	A free-standing @ in the name field denotes the current origin.
..	Two free-standing dots represent the null domain name of the root when used in the name field.
\X	Where X is any character other than a digit (0-9), quotes that character so that its special meaning does not apply. For example, you can use \. to place a dot character in a label.
\DDD	Where each D is a digit, this is the octet corresponding to the decimal number described by DDD. The resulting octet is assumed to be text and is not checked for special meaning.
()	Use parentheses to group data that crosses a line. In effect, line terminations are not recognized within parentheses.
;	A semicolon starts a comment; the remainder of the line is ignored.
*	An asterisk signifies a wildcard.

Most resource records have the current origin appended to names if they are not terminated by a dot (.). This is useful for appending the current domain name to the data, such as machine names, but might cause problems when you do not want this to happen. You should use a fully qualified name ending in a period if the name is not in the domain for which you are creating the data file.

Control Entries

The only lines that do not conform to the standard RR format in a data file are control-entry lines. There are two kinds of control entries: `$INCLUDE()` and `$ORIGIN()`.

`$INCLUDE`

An include line begins with `$INCLUDE` in column 1, and is followed by a file name (known as the `$INCLUDE` file). This feature is particularly useful for separating different types of data into multiple files as in this example:

```
$INCLUDE /etc/named/data/mailboxes
```

The line is interpreted as a request to load the `/etc/named/data/mailboxes` file at that point. The `$INCLUDE` command does not cause data to be loaded into a different zone or tree. The command allows for data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

Use of `$INCLUDE` statements and files is optional. You can use as many as you wish, or none at all.

`$ORIGIN()`

The `$ORIGIN` command is a way of changing the origin in a data file. The line starts in column 1, and is followed by a domain name. It resets the current origin for relative domain names (for example, not fully qualified names) to the stated name. This is useful for putting more than one domain in a data file.

Note – You cannot use `$ORIGIN()` for putting more than one zone in a data file.

Use of `$ORIGIN` commands in a data file is optional. If there is no `$ORIGIN()` statement the default origin for DNS data files is the domain named in the second field of the `master` or `slave` line of the `named.conf` file.

Resource Record Types

The most commonly used types of resource records are listed in Table 5–7. They are usually entered in the order shown in Table 5–7, but that is not a requirement.

TABLE 5-7 Commonly Used Resource Record Types

Type	Description
SOA	Start of authority
NS	Name server
A	Internet address (name to address)
PTR	Pointer (address to name)
CNAME	Canonical name (nickname)
TXT	Text information
WKS	Well-known services
HINFO	Host information
MX	Mail exchanger

Start-of-Authority record (SOA)

Example 5-19 shows the syntax of a start-of-authority (SOA) resource record.

EXAMPLE 5-19 SOA Record Format

```
name class SOA origin person-in-charge ( serial number  
    refresh  
    retry  
    expire  
    ttl)
```

The SOA record designates the start of a zone. The zone ends at the next SOA record. The SOA record fields are described below.

name

This field indicates the name of the zone. Note that the zone name must end with a trailing dot. For example: `doc.com.` is correct, while `doc.com` is wrong.

class

This field is the address class. For example, `IN` for Internet (the most commonly used class).

SOA

This field is the type of this resource record.

origin

This field is the name of the host where this data file resides. Note that this host name must end in a trailing dot. For example, `dnsmaster.doc.com.` is correct, but `dnsmaster.doc.com` is wrong.

person-in-charge

This field is the email address of the person responsible for the name server. For example, `kjd.nismaster.doc.com.` Again, this name must end with a trailing dot.

serial

This field is the version number of this data file. You must increment this number whenever you make a change to the data: slave servers use the `serial` field to detect whether the data file has been changed since the last time they copied the file from the master server.

refresh

This field indicates how often, in seconds, a slave name server should check with the master name server to see if an update is needed. For example, `7200` indicates a period of two hours.

retry

This field indicates how long, in seconds, a slave server is to retry after a failure to check for a refresh.

expire

This field is the upper limit, in seconds, that a slave name server is to use the data before it expires for lack of getting a refresh.

ttl

This field is the default number of seconds to be used for the time-to-live field on resource records that do not have a `tll` specified elsewhere.

There should only be one SOA record per zone. Example 5–20 is a sample SOA resource record.

EXAMPLE 5–20 Sample SOA Resource Record

```
;name class      SOA      origin      person-in-charge
doc.com. IN      SOA      dnsmaster.doc.com. root.nismaster.doc.com. (
                                101      ;Serial
```

EXAMPLE 5-20 Sample SOA Resource Record (Continued)

```
7200      ;Refresh
3600      ;Retry
432000    ;Expire
86400)    ;Minimum      )
```

Name Server (NS)

Example 5-21 shows the syntax of a name-server (NS) resource record:

EXAMPLE 5-21 NS Record Format

```
domainname [optional TTL] class NS name-server-name
```

The name-server record lists by name a server responsible for a given domain. The *name* field lists the domain that is serviced by the listed name server. If no *name* field is listed, then it defaults to the last name listed. One NS record should exist for each master and slave server for the domain. Example 5-22 is a sample NS resource record.

EXAMPLE 5-22 Sample NS Resource Record

```
;domainname [TTL] class NS nameserver
doc.com      90000 IN NS sirius.doc.com.
```

Address (A)

Example 5-23 shows the syntax of an address (A) resource record:

EXAMPLE 5-23 Address Record Format

```
machinename [optional TTL] class A address
```

The A record lists the address for a given machine. The *name* field is the host name, and the *address* is the IP address. One A record should exist for each address of the machine (in other words, routers, or gateways require at least two entries, a separate entry including the IP address assigned to each network interface).

EXAMPLE 5-24 Sample Address Record

```
;machinename [TTL] class A address
sirius       IN A 123.45.6.1
```

Host Information (HINFO)

Example 5-25 shows the syntax of a host-information (HINFO) resource record:

EXAMPLE 5-25 HINFO Record Format

```
[optional name] [optional TTL]      class   HINFO hardware  OS
```

The HINFO contains host-specific data. It lists the hardware and operating environment that are running at the listed host. If you want to include a space in the machine name or in the entry in the *hardware* field, you must surround the entry with quotes. The *name* field specifies the name of the host. If no name is specified, it defaults to the last *in.*-named host. One HINFO record should exist for each host. Example 5-26 is a sample HINFO resource record.

EXAMPLE 5-26 Sample HINFO Resource Record

```
; [name]      [TTL]  class HINFO  hardware  OS
                IN    HINFO  Sparc-10   UNIX
```



Caution – Because the HINFO field provides information about the machines on your network, many sites consider it a security risk and no longer use it.

Well-Known Services (WKS)

Example 5-27 shows the syntax of a well-known services (WKS) resource record:

EXAMPLE 5-27 WKS Record Format

```
[Optional name] [TTL] class WKS address protocol-list-of-services
```

The WKS record describes the well-known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in the *services* database. Only one WKS record should exist per protocol per address. Example 5-28 is an example of a WKS resource record.

EXAMPLE 5-28 Sample WKS Resource Record

```
; [name]      [TTL]  class   WKS    address      protocol-list-of-services
altair        IN    WKS    123.45.6.1  TCP ( smtp discard rpc
sftp uucp-path systat daytime
netstat qotd nntp doc.com )
```



Caution – The WKS record is optional. For security reasons, most sites no longer provide this information.

Canonical Name (CNAME)

Example 5–29 shows the syntax of a canonical-name (CNAME) resource record.

EXAMPLE 5–29 CNAME Record Format

```
nickname [optional TTL] class CNAME canonical-name
```

The CNAME specifies a nickname or alias for a canonical name. A nickname should be unique. All other resource records should be associated with the canonical name and not with the nickname. Do not create a nickname and then use it in other resource records. Nicknames are particularly useful during a transition period, when a machine's name has changed but you want to permit people using the old name to reach the machine. Nicknames can also be used to identify machines that serve some specific purpose such as a mail server. Example 5–30 is a sample CNAME resource record.

EXAMPLE 5–30 Sample CNAME Resource Record

```
;nickname      [TTL]      class   CNAME      canonical-name  
mailhost       IN          CNAME   antares.doc.com
```

Pointer Record (PTR)

Example 5–31 shows the syntax for a PTR resource record.

EXAMPLE 5–31 PTR Record Format

```
special-name [optional TTL] class PTR-real-name
```

A pointer record allows special names to point to some other location in the domain. In the example, PTRs are used mainly in the `in-addr.arpa.` records for the translation of an address (the special name) to a real name. When translating an address, if the domain is fully qualified only the machine identification number need be specified. PTR names should be unique to the zone. The PTR records Example 5–32 sets up reverse pointers for the special `in-addr.arpa` domain.

EXAMPLE 5–32 Sample PTR Resource Record

```
;special name  [TTL]      class   PTR-real-name  
1             IN          PTR     sirius.doc.com.
```

Mail Exchanger (MX)

Example 5–33 shows the syntax for a mail-exchanger (MX) resource record.

EXAMPLE 5–33 MX Record Format

```
name [optional TTL] class MX preference-value mailer-exchanger
```

The MX resource records are used to specify a machine that knows how to deliver mail to a domain or specific machines in a domain. There might be more than one MX resource record for a given name. In Example 5–34, `Seismo.CSS.GOV.` (note the fully qualified domain name) is a mail gateway that knows how to deliver mail to `Munnari.OZ.AU.` Other machines on the network cannot deliver mail directly to `Munnari.Seismo` and `Munnari` might have a private connection or use a different transport medium. The *preference-value* field indicates the order a mailer should follow when there is more than one way to deliver mail to a single machine. The value 0 (zero) indicates the highest preference. If there is more than one MX resource record for the same name, records might or might not have the same *preference* value.

You can use names with the wildcard asterisk (*) for mail routing with MX records. There are likely to be servers on the network that state that any mail to a domain is to be routed through a relay. In Example 5–34, all mail to hosts in domain `foo.com` is routed through `RELAY.CS.NET.` You do this by creating a wildcard resource record, which states that the mail exchanger for `*.foo.com` is `RELAY.CS.NET.` The asterisk will match any host or subdomain of `foo.com`, but it will not match `foo.com` itself.

EXAMPLE 5–34 Sample MX Resource Record

```

;name          [TTL]    class    MX    preference mailer-exchanger
Munnari.OZ.AU.      IN      MX      MX    0      Seismo.CSS.GOV.
foo.com.           IN      MX      MX    10     RELAY.CS.NET.
*.foo.com.         IN      MX      MX    20     RELAY.CS.NET.

```


DNS Troubleshooting (Reference)

This chapter described some common DNS problems and how to solve them.

Clients Can Find Machine by Name but Server Cannot

Symptoms

DNS clients can find machines by either IP address or by host name, but the server can only find machines by their IP addresses.

Probable cause and solution

This is most likely caused by omitting `dns` from the `hosts` line of the server's `nsswitch.conf` file. For example, a *bad* `hosts` line might look like this: `hosts: files`

When using DNS you must include `dns` in the `hosts` record of every machine's `nsswitch.conf` file. For example:

```
hosts: dns nisplus files
```

or

```
hosts: nisplus dns files
```

Changes Do Not Take Effect or Are Erratic

Symptom

You add or delete machines or servers but your changes are not recognized or do not take effect. Or in some instances the changes are recognized and at other times they are not yet in effect.

Probable cause

The most likely cause is that you forgot to increment the SOA serial number on the master server after you made your change. Since there is no new SOA number, your slave servers do not update their data to match that of the master so they are working with the old, unchanged data files.

Another possible cause is that the SOA serial number in one or more of the master data files was set to a value lower than the corresponding serial number on your slave servers. This could happen, for example, if you deleted a file on the master and then recreated it from scratch using an input file of some sort.

A third possible cause is that you forgot to send a HUP signal to the master server after making changes to the primary's data files.

Diagnosis and solution

First, check the SOA serial numbers in the data file that you changed and the corresponding file on the slave server.

- If the SOA serial number in the master file is equal to, or less than, the serial number in the slave file, increase the serial number on the primary's file so that it is greater than the number in the slave file. For example, if the SOA number in both files is 37, change the number in the primary's file to 38. The next time the slave checks with the primary, it will load the new data. (There are utilities that can force a master to immediately transfer data to the secondaries, if you have one of these utilities you can update the slave without waiting for it to check the primary.)
- Review the `syslog` output for the most recent named `named` restarted or named `named` reloading `named` entry. If the timestamp for that entry is before the time you finished making changes to the file, either reboot the server or force it to read the new data as explained in "Forcing `named` to Reload DNS Data" on page 78.

DNS Client Cannot Lookup “Short” Names

Symptoms

Client can lookup fully qualified names but not short names.

Possible cause and solution

Check the client's `/etc/resolv.conf` file for spaces at the end of the domain name. No spaces or tabs are allowed at the end of the domain name.

Reverse Domain Data Not Correctly Transferred to slave

While zone domain-named data is properly transferred from the zone master server to a zone slave server, the reverse domain data is not being transferred. In other words, the `host.rev` file on the slave is not being properly updated from the primary.

Possible causes

Syntax error in the slave server's boot file.

Diagnosis and Solution

Check the slave server's boot file. Make sure that the master server's IP address is listed for the reverse zone entries just as it is for the hosts data.

Server Failed and Zone Expired Problems

When a slave server cannot obtain updates from its master, it logs a `master unreachable` message. If the problem is not corrected, the slave expires the zone and stops answering requests from clients. When that happens, users start seeing `server failed` messages.

Symptoms

- Masters for slave zone *domain* unreachable messages in syslog.
- slave zone *domain* expired messages in syslog.
- *** *domain* Can't find *name*: server failed messages to users.

Note that if the problem is with a slave server, some users could still be successfully obtaining DNS information from the master and thus operating without experiencing any difficulty.

Possible causes

The two most likely causes for these problems are network failure and a wrong IP address for the master in the slave's boot file.

Diagnosis and solution

- Check that the slave's configuration file contains the correct IP address for the master. Check the line:

```
zone "someone" {  
                                type slave;  
    file "somefile":  
    master [IPaddress; ];  
};
```

Make sure that the IP address of the master matches the master's actual IP address and the address for the master specified in the hosts file. If the IP address is wrong, correct it, and then reboot the slave.

- If the master's IP address is correct, make sure the master is up and running correctly by pinging the master's IP address. For example, to ping the master at IP address 192.168.0.1, you would enter the following.
% **ping 192.168.0.1 -n 10**
- If the master does not respond to the ping, make sure it is up and running properly.
- If the master is running okay, use `ps` to make sure it is running `named`. If it is not running `named`, reboot it.
- If the master is correctly running `named`, you most likely have a network problem.

rlogin, rsh, and ftp Problems

Symptoms

- Users are asked for password when they try to `rlogin` to a machine in another domain over the Internet.

- Users are denied access when they try to ftp to a machine in another domain over the Internet.
- Users are denied access when they try to use rlogin or rsh to a machine on their own network.

Possible causes

- The user is working at a machine that does not have a PTR record in the master server's hosts.rev file.
- A missing or incorrect delegation of a subdomain in the hosts.rev file.

Diagnosis and solution

Check the appropriate hosts.rev file and make sure there is a PTR record for the user's machine. For example, if the user is working at the machine altair.doc.com with an IP address of 192.168.0.1, the doc.com master server's doc.rev file should have an entry like:

```
46      IN      PTR      altair.doc.com.
```

If the record is missing, add it to the hosts.rev file and then reboot the server or reload its data as explained in "Forcing in.named to Reload DNS Data" on page 78.

Check and correct the NS entries in the hosts.rev files and then reboot the server or reload its data as explained in "Forcing in.named to Reload DNS Data" on page 78.

Other DNS Syntax Errors

Symptoms

Error messages in console or syslog with operative phrases like the following are most often caused by syntax errors in DNS data and boot files.

- No such...
- Unknown field...
- Non-authoritative answer:
- Database format error...
- illegal or (illegal)
- error receiving zone transfer

Check the relevant files for spelling and syntax errors.

A common syntax error is misuse of the trailing dot in domain names (either using the dot when you should not, or not using it when you should). See "Setting up DNS Servers" on page 75.

PART III NIS Setup and Administration

This part provides an overview of the NIS naming service, as well as the setup, administration and troubleshooting of NIS within the Solaris operating environment.

Network Information Service (NIS) (Overview)

This chapter provides an overview of the Network Information Service (NIS).

NIS is a distributed naming service. It is a mechanism for identifying and locating network objects and resources. It provides a uniform storage and retrieval method for network-wide information in a transport-protocol and media-independent fashion.

This chapter covers the following topics.

- “NIS Introduction” on page 125
- “NIS Machine Types” on page 127
- “NIS Elements” on page 128
- “NIS Binding” on page 134
- “Differences in NIS Solaris 2.6 NIS and Earlier NIS Versions” on page 136

NIS Introduction

By running NIS, the system administrator can distribute administrative databases, called *maps*, among a variety of servers (*master* and *slaves*). The administrator can update those databases from a centralized location in an automatic and reliable fashion to ensure that all clients share the same naming service information in a consistent manner throughout the network.

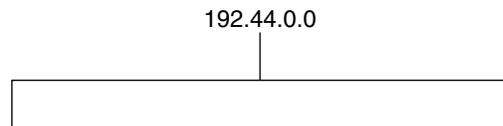
NIS was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information not only about machine names and addresses, but also about users, the network itself, and network services. This collection of network *information* is referred to as the NIS *namespace*.

Note – In some contexts *machine* names are referred to as *host* names or *machine* names. This discussion uses *machine*, but some screen messages or NIS map names might use *host* or *machine*.

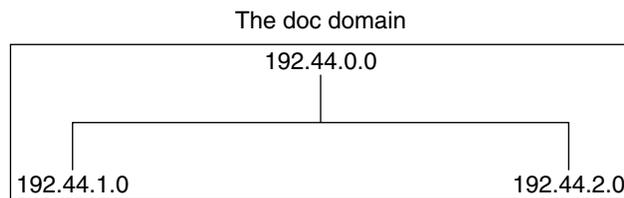
NIS Architecture

NIS uses a client-server arrangement. NIS servers provide services to NIS clients. The principal servers are called *master* servers, and for reliability, they have backup, or *slave* servers. Both master and slave servers use the NIS information retrieval software and both store NIS maps.

NIS uses domains to arrange the machines, users, and networks in its namespace. However, it does not use a domain hierarchy; an NIS namespace is flat.



Thus, this physical network would be arranged into one NIS domain.



An NIS domain cannot be connected directly to the Internet using just NIS. However, organizations that want to use NIS and also be connected to the Internet can combine NIS with DNS. You can use NIS to manage all local information and use DNS for Internet host lookup. NIS provides a forwarding service that forwards host lookups to DNS if the information cannot be found in an NIS map. The Solaris operating environment also allows you to set up the `nsswitch.conf` file so that hosts lookup requests go only to DNS, or to DNS and then NIS if not found by DNS, or to NIS and then DNS if not found by NIS. See Chapter 2 for details.

NIS Machine Types

There are three types of NIS machines.

- Master server
- Slave servers
- Clients of NIS servers

Any machine can be an NIS client, but only machines with disks should be NIS servers, either master or slave. Servers are also clients, typically of themselves.

NIS Servers

The NIS server does not have to be the same machine as the NFS file server.

NIS servers come in two varieties, master and slave. The machine designated as master server contains the set of maps that the system administrator creates and updates as necessary. Each NIS domain must have one, and only one, master server, which can propagate NIS updates with the least performance degradation.

You can designate additional NIS servers in the domain as slave servers. A slave server has a complete copy of the master set of NIS maps. Whenever the master server maps are updated, the updates are propagated among the slave servers. Slave servers can handle any overflow of requests from the master server, minimizing “server unavailable” errors.

Normally, the system administrator designates one master server for all NIS maps. However, because each individual NIS map has the machine name of the master server encoded within it, you could designate different servers to act as master and slave servers for different maps. To minimize confusion, designate a single server as the master for all the maps you create within a single domain. The examples in this chapter assume that one server is the master for all maps in the domain.

NIS Clients

NIS clients run processes that request data from maps on the servers. Clients do not make a distinction between master and slave servers, since all NIS servers should have the same information.

NIS Elements

The NIS naming service is composed of the following elements:

- Domains (see “The NIS Domain” on page 128).
- Maps (see “NIS Maps” on page 129).
- Daemons (see “NIS Daemons” on page 128).
- Utilities (see “NIS Utilities” on page 129).
- NIS Command Set (see “NIS-Related Commands” on page 133).

The NIS Domain

An NIS *domain* is a collection of machines which share a common set of NIS maps. Each domain has a domain name and each machine sharing the common set of maps belongs to that domain.

Any machine can belong to a given domain, as long as there is a server for that domain’s maps in the same network. An NIS client machine obtains its domain name and binds to an NIS server as part of its boot process.

NIS Daemons

NIS service is provided by five daemons as shown in Table 7–1.

TABLE 7–1 NIS Daemons

Daemon	Function
<code>ypserv</code>	Server process
<code>ypbind</code>	Binding process
<code>ypxfr</code>	High speed map transfer
<code>rpc.yppasswdd</code>	NIS password update daemon ** See NOTE below.**
<code>rpc.yupdated</code>	Modifies other maps such as <code>publickey</code>

Note – `rpc.yppasswdd` considers all shells that begin with an `r` to be restricted. This means that users who have a shell that begins with an `r`. For example, if you are in `/bin/rksh`, you are not allowed to change from that shell to another one. If you have a shell that begins with `r` but is not intended to be restricted as such, refer to Chapter 10 for the workaround.

NIS Utilities

NIS service is supported by nine utilities as shown in Table 7-2.

TABLE 7-2 NIS Utilities

Utility	Function
<code>makedbm</code>	Creates <code>dbm</code> file for an NIS map
<code>ypcat</code>	Lists data in a map
<code>ypinit</code>	Builds and installs an NIS database and initializes NIS client's <code>ybservers</code> list.
<code>ypmatch</code>	Finds a specific entry in a map
<code>ypoll</code>	Gets a map order number from a server
<code>yppush</code>	Propagates data from NIS master to NIS slave server
<code>ypset</code>	Sets binding to a particular server
<code>ypwhich</code>	Lists name of the NIS server and nickname translation table
<code>ypxfr</code>	Transfers data from master to slave NIS server

NIS Maps

The information in NIS maps is stored in `ndbm` format. `ypfiles(4)` and `ndbm(3C)` explain the format of the map file.

NIS maps were designed to replace UNIX `/etc` files, as well as other configuration files, so they store much more than names and addresses. On a network running NIS, the NIS master server for each NIS domain maintains a set of NIS maps for other machines in the domain to query. NIS slave servers also maintain duplicates of the master server's maps. NIS client machines can obtain namespace information from either master or slave servers.

NIS maps are essentially two-column tables. One column is the *key* and the other column is information related to the key. NIS finds information for a client by searching through the keys. Some information is stored in several maps because each

map uses a different key. For example, the names and addresses of machines are stored in two maps: `hosts.byname` and `hosts.byaddr`. When a server has a machine's name and needs to find its address, it looks in the `hosts.byname` map. When it has the address and needs to find the name, it looks in the `hosts.byaddr` map.

An NIS `Makefile` is stored in the `/var/yp` directory of machines designated as an NIS server at installation time. Running `make` in that directory causes `makedbm` to create or modify the default NIS maps from the input files.

Note – Always create maps on the master server, as maps created on a slave will not automatically be pushed to the master server.

Default NIS Maps

A default set of NIS maps are provided in the Solaris operating environment. You might want to use all these maps or only some of them. NIS can also use whatever maps you create or add when you install other software products.

Default maps for a NIS domain are located in each server's `/var/yp/domainname` directory. For example, the maps that belong to the domain `test.com` are located in each server's `/var/yp/test.com` directory.

Table 7-3 describes the default NIS maps, information they contain, and whether the software consults the corresponding administrative files when NIS is running.

TABLE 7-3 NIS Map Descriptions

Map Name	Corresponding NIS Admin File	Description
<code>bootparams</code>	<code>bootparams</code>	Contains path names of files clients need during boot: root, swap, possibly others.
<code>ethers.byaddr</code>	<code>ethers</code>	Contains machine names and Ethernet addresses. The Ethernet address is the key in the map.
<code>ethers.byname</code>	<code>ethers</code>	Same as <code>ethers.byaddr</code> , except the key is machine name instead of the Ethernet address.
<code>group.bygid</code>	<code>group</code>	Contains group security information with group ID as key.
<code>group.byname</code>	<code>group</code>	Contains group security information with group name as key.

TABLE 7-3 NIS Map Descriptions (Continued)

Map Name	Corresponding NIS Admin File	Description
hosts.byaddr	hosts	Contains machine name, and IP address, with IP address as key.
hosts.byname	hosts	Contains machine name and IP address, with machine (host) name as key.
mail.aliases	aliases	Contains aliases and mail addresses, with aliases as key.
mail.byaddr	aliases	Contains mail address and alias, with mail address as key.
netgroup.byhost	netgroup	Contains group name, user name and machine name.
netgroup.byuser	netgroup	Same as netgroup.byhost, except that key is user name.
netgroup	netgroup	Same as netgroup.byhost, except that key is group name.
netid.byname	passwd, hosts group	Used for UNIX-style authentication. Contains machine name and mail address (including domain name). If there is a netid file available it is consulted in addition to the data available through the other files.
netmasks.byaddr	netmasks	Contains network mask to be used with IP submitting, with the address as key.
networks.byaddr	networks	Contains names of networks known to your system and their IP addresses, with the address as key.
networks.byname	networks	Same as networks.byaddr, except key is name of network.
passwd.adjunct.byname	passwd and shadow	Contains auditing information and the hidden password information for C2 clients.
passwd.byname	passwd and shadow	Contains password information with user name as key.
passwd.byuid	passwd and shadow	Same as passwd.byname, except that key is user ID.
protocols.byname	protocols	Contains network protocols known to your network.

TABLE 7-3 NIS Map Descriptions (Continued)

Map Name	Corresponding NIS Admin File	Description
<code>protocols.bynumber</code>	<code>protocols</code>	Same as <code>protocols.byname</code> , except that key is protocol number.
<code>rpc.bynumber</code>	<code>rpc</code>	Contains program number and name of RPCs known to your system. Key is RPC program number.
<code>services.byname</code>	<code>services</code>	Lists Internet services known to your network. Key is port or protocol.
<code>services.byservice</code>	<code>services</code>	Lists Internet services known to your network. Key is service name.
<code>ypservers</code>	N/A	Lists NIS servers known to your network.

New `ipnodes` maps (`ipnodes.byaddr` and `ipnodes.byname`) are added to NIS. The maps store both IPv4 and IPv6 addresses. See `ipnodes(4)`. NIS clients and servers can communicate using either IPv4 or IPv6 RPC transports.

Using NIS Maps

NIS makes updating network databases much simpler than with the `/etc` files system. You no longer have to change the administrative `/etc` files on every machine each time you modify the network environment.

For example, when you add a new machine to a network running NIS, you only have to update the input file in the master server and run `make`. This automatically updates the `hosts.byname` and `hosts.byaddr` maps. These maps are then transferred to any slave servers and are made available to all of the domain's client machines and their programs. When a client machine or application requests a machine name or address, the NIS server refers to the `hosts.byname` or `hosts.byaddr` map as appropriate and sends the requested information to the client.

You can use the `ypcat` command to display the values in a map. The `ypcat` basic format is the following.

```
% ypcat mapname
```

where *mapname* is the name of the map you want to examine or its *nickname*. If a map is composed only of keys, as in the case of `ypservers`, use `ypcat -k`. Otherwise, `ypcat` prints blank lines. The `ypcat` man page describes more options for `ypcat`.

You can use the `ypwhich` command to determine which server is the master of a particular map. Type the following.

```
% ypwhich -m mapname
```

where *mapname* is the name or the nickname of the map whose master you want to find. `ypwhich` responds by displaying the name of the master server. For complete information, refer to the `ypwhich` man page.

NIS Map Nicknames

Nicknames are aliases for full map names. To obtain a list of available map nicknames, such as `passwd` for `passwd.byname`, type `ypcat -x` or `ypwhich -x`.

Nicknames are stored in the `/var/yp/nicknames` file, which contains a map nickname followed by the fully specified name for the map, separated by a space. This list might be added to or modified. Currently, there is a limit of 500 nicknames.

NIS-Related Commands

The NIS service includes specialized daemons, system programs, and commands, which are summarized in the following table.

TABLE 7-4 NIS Command Summary

Command	Description
<code>ypserv</code>	Serves NIS clients' requests for information from an NIS map. <code>ypserv</code> is a daemon that runs on NIS servers with a complete set of maps. At least one <code>ypserv</code> daemon must be present on the network for NIS service to function.
<code>ypbind</code>	Provides NIS server binding information to clients. It provides binding by finding a <code>ypserv</code> process that serves maps within the domain of the requesting client. <code>ypbind</code> must run on all servers and clients.
<code>ypinit</code>	Automatically creates maps for an NIS server from the input files. It is also used to construct the initial <code>/var/yp/binding/domain/ypservers</code> file on the clients. Use <code>ypinit</code> to set up the master NIS server and the slave NIS servers for the first time.
<code>make</code>	Updates NIS maps by reading the <code>Makefile</code> (when run in the <code>/var/yp</code> directory). You can use <code>make</code> to update all maps based on the input files or to update individual maps. The <code>ypmake (1M)</code> man page describes the functionality of <code>make</code> for NIS.
<code>makedbm</code>	<code>makedbm</code> takes an input file and converts it into <code>dbm.dir</code> and <code>dbm.pag</code> files—valid <code>dbm</code> files that NIS can use as maps. You can also use <code>makedbm -u</code> to disassemble a map, so that you can see the key-value pairs that comprise it.

TABLE 7-4 NIS Command Summary (Continued)

Command	Description
<code>ypxfr</code>	Pulls an NIS map from a remote server to the local <code>/var/yp/domain</code> directory, using NIS itself as the transport medium. You can run <code>ypxfr</code> interactively, or periodically from a <code>crontab</code> file. It is also called by <code>ypserv</code> to initiate a transfer.
<code>ypxfrd</code>	Provides map transfers service for <code>ypxfr</code> requests (generally slave servers). It is run only on the master server.
<code>yppush</code>	Copies a new version of an NIS map from the NIS master server to its slaves. You run it on the master NIS server.
<code>ypset</code>	Tells a <code>ypbind</code> process to bind to a named NIS server. This is not for casual use and its use is discouraged because of security implications. See the <code>ypset(1M)</code> and <code>ypbind(1M)</code> man pages for information about the <code>ypset</code> and <code>ypsetme</code> options to the <code>ypbind</code> process.
<code>ypoll</code>	Tells which version of an NIS map is running on a server that you specify. It also lists the master server for the map.
<code>ypcat</code>	Displays the contents of an NIS map.
<code>ypmatch</code>	Prints the value for one or more specified keys in an NIS map. You cannot specify which version of the NIS server map you are seeing.
<code>ypwhich</code>	Shows which NIS server a client is using at the moment for NIS services, or, if invoked with the <code>-m mapname</code> option, which NIS server is master of each of the maps. If only <code>-m</code> is used, it displays the names of all the maps available and their respective master servers.

NIS Binding

NIS clients get information from an NIS server through the binding process, which can work in one of two modes: server-list or broadcast.

- **Server-list.** In the server-list mode, the `ypbind` process queries the `/var/yp/binding/domain/ypservers` list for the names of all of the NIS servers in the domain. The `ypbind` process binds only to servers in this file. The file is created by running `ypinit -c`.
- **Broadcast.** The `ypbind` process can also use an RPC broadcast to initiate a binding. Since broadcasts are only local subnet events that are not routed further, there must be at least one server (master or slave) on the same subnet as the client. The servers themselves might exist throughout different subnets since map propagation works across subnet boundaries. In a subnet environment, one common method is to make the subnet router an NIS server. This allows the domain server to serve clients on either subnet interface.

Server-List Mode

The binding process in server-list mode works as follows:

1. Any program, running on the NIS client machine that needs information provided by an NIS map, asks `ypbind` for the name of a server.
2. `ypbind` looks in the `/var/yp/binding/domainname/ypservers` file for a list of NIS servers for the domain.
3. `ypbind` initiates binding to the first server in the list. If the server does not respond, `ypbind` tries the second, and so on, until it finds a server or exhausts the list.
4. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
5. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
6. `ypserv` sends the requested information back to the client.

Broadcast Mode

The broadcast mode binding process works as follows:

1. `ypbind` must be started with the broadcast option set (`broadcast`).
2. `ypbind` issues an RPC broadcast in search of an NIS server.

Note – In order to support such clients, it is necessary to have an NIS server on each subnet requiring NIS service.

3. `ypbind` initiates binding to the first server that responds to the broadcast.
4. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
5. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
6. `ypserv` sends the requested information back to the client.

Normally, once a client is bound to a server it stays bound to that server until something causes it to change. For example, if a server goes out of service, the clients it served will then bind to new servers.

To find out which NIS server is currently providing service to a specific client, use the following command.

```
%ypwhich machinename
```

Where *machinename* is the name of the client. If no machine name is mentioned, `ypwhich` defaults to the local machine (that is, the machine on which the command is run).

Differences in NIS Solaris 2.6 NIS and Earlier NIS Versions

The following NIS features are new or different in Solaris 2.6.

NSKit Discontinued

The most recent Solaris releases have not included NIS service. Up to now, NIS service had to be installed from the unbundled NSKit. NIS has now been included in the Solaris 2.6 and there is no 2.6 NSKit.

Because NIS service is now part of the Solaris 2.6, the `SUNWnsktu` and `SUNWnsktr` packages no longer exist. Instead, NIS is now installed via the NIS Server cluster (containing the `SUNWypu` and `SUNWyptr` packages).

NIS service is no longer started with the `/etc/init.d/yp` script which no longer exists. With the Solaris 2.6, you first configure your master server NIS maps with the `ypinit` script, and then start NIS with `ypstart`. NIS service is stopped with the `ypstop` command.

The `ypupdated` Daemon

The most recent versions of NSKit did not include the `ypupdated` daemon. The `ypupdated` daemon is now included in this Solaris release.

`/var/yp/securenets`

As with the previous NSKit release, the `/var/yp/securenets` file is now used to limit access to NIS services. If such a file exists on an NIS server, the server only answers queries or supplies maps to machines and networks whose IP addresses are listed in the file. For the file format, see `securenets(4)`.

The following is an example of a `securenets` file.

```
255.255.255.10      192.168.0.1
host    13.13.14.1
host    13.13.14.2
```

where 255 . 255 . 255 . 10 is the netmask and 13 . 13 . 13 . 255 is the network address. For the set up in line 1, `ypserv` responds to only those addresses in the subnet 13 . 13 . 13 . 255 range.

If you modify entries in the `/var/yp/securenets` file, you must kill and restart the `ypserv` and `ypxfrd` daemons.

Multihomed Machine Support

As with the previous NSKit release, the `ypserv` process provides support for machines which have more than one network address. When the machine maps are created, the `Makefile` creates a `YP_MULTI_HOSTNAME` entry in the map for any machine that has more than one address. This entry lists all the addresses for that machine. When the machine address is needed, an attempt is made to use the closest address on the list. See the `ypserv` man page for more details.

The determination of closest address is an arithmetic one and as such there is no check for address validity. For example, suppose that a multihomed machine has six IP addresses and only five of the interfaces on the machine are operating normally. Machines on a network that is not directly connected to this multihomed machine can receive the IP address for the down interface from `ypserv`. Thus, this hypothetical client can not reach the multihomed machine.

Note – All addresses for a multihomed machine should normally be active. If a particular address or machine is going to be out of service, remove it from the NIS maps.

SunOS 4 Compatibility Mode

NIS supports password configuration files in both the SunOS 4 (Solaris 1) password format and the Solaris 2 password and shadow file formats.

The mode of operation is determined by the existence of the file `$PWDIR/shadow`, where `$PWDIR` is the `Makefile` macro set in the `/var/yp/Makefile` file. If the shadow file exists, NIS operates in the Solaris 2 mode. If this file does not exist, NIS operates in the SunOS 4 mode.

In the SunOS 4 mode, all password information is kept in the `passwd` file. In the Solaris 2 mode, password information is kept in the `shadow` file and the user account information is kept in the `passwd` file.

If the make macro `PWDIR` is set to the `/etc` directory, NIS can operate only in the Solaris 2 mode because of the Solaris 2 `passwd` processing requirements. However, if `PWDIR` points to any directory other than `/etc`, the user has the option of keeping `passwd` configuration files in either the SunOS 4 format or in the Solaris 2 format. The `rpc.yppassd` daemon understands both password formats. The Solaris 2 format is recommended.

Setting Up and Configuring NIS Service

This chapter describes initial set up and configuration of the Network Information Service (NIS).

Note – In some contexts, *machine* names are referred to as *host* names or *machine* names. This discussion uses “machine,” but some screen messages or NIS map names might use *host* or *machine*.

This chapter covers the following topics.

- “Configuring NIS — Task Map” on page 139
- “Before You Begin Configuring NIS” on page 140
- “Planning Your NIS Domain” on page 140
- “Preparing the Master Server” on page 141
- “Starting NIS Service on the Master Server” on page 146
- “Setting Up NIS Slave Servers” on page 147
- “Setting Up NIS Clients” on page 149

Configuring NIS — Task Map

Task	For Instructions, Go To
Prepare source files for conversion.	“Preparing Source Files for Conversion to NIS Maps” on page 142

Task	For Instructions, Go To
Set up master server using <code>ypinit</code>	"How to Set Up the Master Server With <code>ypinit</code> " on page 144
Start NIS on master server.	"Starting NIS Service on the Master Server" on page 146
Set up slave servers.	"Setting Up a Slave Server" on page 147
Set up NIS client.	"Setting Up NIS Clients" on page 149

Before You Begin Configuring NIS

Before configuring your NIS namespace, you must do the following.

- Install properly configured `nsswitch.conf` files on all the machines that will be using NIS. See Chapter 2 for details.
- Plan your NIS domain. See the following section.

Planning Your NIS Domain

Before you configure machines as NIS servers or clients, you must plan the NIS domain.

Decide which machines will be in your NIS domain. An NIS domain does not have to be congruent with your network. A network can have more than one NIS domain, and there can be machines on your network that are outside of your NIS domain.

Choose an NIS domain name, which can be 256 characters long. A good practice is to limit domain names to no more than 32 characters. Domain names are case-sensitive. For convenience, you can use your Internet domain name as the basis for your NIS domain name. For example, if your Internet domain name is `doc.com`, you can name your NIS domain `doc.com`. If you wanted to divide `doc.com` into two NIS domains, one for the sales department and the other for the manufacturing department, you could name one `sales.doc.com` and the other `manf.doc.com`.

Before a machine can use NIS services, the correct NIS domain name and machine name must be set. A machine's name is set by the machine's `/etc/nodename` file and the machine's domain name is set by the machine's `/etc/defaultdomain` file. These files are read at boot time and the contents are used by the `uname -S` and `domainname` commands, respectively. Diskless machines read these files from their boot server.

Identify Your NIS Servers and Clients

Select one machine to be the master server. Decide which machines, if any, will be slave servers.

Decide which machines will be NIS clients. Typically all machines in your domain are set to be NIS clients, although this is not necessary.

Preparing the Master Server

Source Files Directory

The source files should be located in the `/etc` directory, on the master server or in some other directory. Having them in `/etc` is undesirable because the contents of the maps are then the same as the contents of the local files on the master server. This is a special problem for `passwd` and `shadow` files because all users have access to the master server maps and the root password would be passed to all YP clients through the `passwd` map. See “`Passwd Files and Namespace Security`” on page 141 for additional information.

However, if you put the source files in some other directory, you must modify the `Makefile` in `/var/yp` by changing the `DIR=/etc` line to `DIR=/your-choice`, where *your-choice* is the name of the directory you will be using to store the source files. This allows you to treat the local files on the server as if they were those of a client. (It is good practice to first save a copy of the original `Makefile`.)

In addition, if `audit_user`, `auth_attr`, `exec_attr` and `prof_attr` are to be taken from a directory other than the default, you must amend the `RBACDIR=/etc/security` to `RBACDIR=/your-choice`.

Passwd Files and Namespace Security

The `passwd` map is a special case. In addition to the old Solaris 1 `passwd` file format, this implementation of NIS accepts the Solaris 7 `/etc/passwd` and `/etc/shadow` file formats as input for building the NIS password maps.

For security reasons, the files used to build the NIS password maps should not contain an entry for `root`, to prevent unauthorized root access. Therefore, the password maps should not be built from the files located in the master server's `/etc` directory. The password files used to build the password maps should have the `root` entry removed from them and be located in a directory that can be protected from unauthorized access.

For example, the master server password input files should be stored in a directory such as `/var/yp`, or any directory of your choice, as long as the file itself is not a link to another file and its location is specified in the `Makefile`. The `/usr/lib/netsvc/yp/ypstart` script automatically sets the correct directory option according to the configuration specified in your `Makefile`.



Caution – Be sure that the `passwd` file in the directory specified by `PWDIR` does not contain an entry for `root`.

If your source files are in a directory other than `/etc`, you must alter the `PWDIR` password macro in the `Makefile` to refer to the directory where the `passwd` and shadow files reside, changing the line `PWDIR=/etc` to `PWDIR/your-choice`, where *your-choice* is the name of the directory you will be using to store the `passwd` map source files.

Preparing Source Files for Conversion to NIS Maps

Prepare the source files for conversion to NIS maps.

▼ How to Prepare Source Files for Conversion

1. **Become superuser.**
2. **Check the source files on the master server to make sure they reflect an up-to-date picture of your environment.**

Check the following files:

- `auto.home` or `auto_home`
- `auto.master` or `auto_master`
- `bootparams`
- `ethers`
- `group`
- `hosts`
- `ipnodes`
- `netgroup`
- `netmasks`
- `networks`
- `passwd`
- `protocols`
- `rpc`
- `service`
- `shadow`
- `user_attr`

3. Copy all of these source files, except `passwd`, to the `DIR` directory that you have selected.
4. Copy the `passwd` file to the `PWDIR` directory that you have selected.
5. Copy `audit_user`, `auth_attr`, `exec_attr`, and `prof_attr` to the selected `RBACDIR` directory.

6. Check the `/etc/mail/aliases` file.

Unlike other source files, the `/etc/mail/aliases` file cannot be moved to another directory. This file must reside in the `/etc/mail` directory. Make sure that the `/etc/mail/aliases` source file contains all the mail aliases that you want to have available throughout the domain. Refer to `aliases(4)` for more information.

7. Clean all comments and other extraneous lines and information from the source files.

These operations can be done through a `sed` or `awk` script or with a text editor. The `Makefile` performs some file cleaning automatically for you, but it is good practice to examine and clean these files by hand before running.

8. Make sure that the data in all the source files is correctly formatted.

Source file data needs to be in the correct format for that particular file. Check the man pages for the different files to make sure that each file is in the correct format.

Preparing the Makefile

After checking the source files and copying them into the source file directory, you now need to convert those source files into the `ndbm` format maps that the NIS service uses. This is done automatically for you by `ypinit` when called on the master server, as explained in the next section, “How to Set Up the Master Server With `ypinit`” on page 144.

The `ypinit` script calls the program `make`, which uses the `Makefile` located in the `/var/yp` directory. A default `Makefile` is provided for you in the `/var/yp` directory and contains the commands needed to transform the source files into the desired `ndbm` format maps.

You can use the default `Makefile` as it is, or modify it if you want. (If you do modify the default `Makefile`, be sure to first copy and store the original default `Makefile` in case you need it for future use.) You might need to make one or more of the following modifications to the `Makefile`:

- *Nondefault maps*

If you have created your own non-default source files and want to convert them to NIS maps, you must add those source files to the `Makefile`.

- *DIR value.* If you want the `Makefile` to use source files stored in some directory other than `/etc`, as explained in “Source Files Directory” on page 141, you must change the value of `DIR` in the `Makefile` to the directory that you want to use. When changing this value in the `Makefile`, do not indent the line.
- *PWDIR value*
If you want the `Makefile` to use `passwd`, `shadow`, and/or adjunct source files stored in some directory other than `/etc`, you must change the value of `PWDIR` in the `Makefile` to the directory that you want to use. When changing this value in the `Makefile`, do not indent the line.
- *Domain name resolver*
If you want the NIS server to use the domain name resolver for machines not in the current domain, comment out the `Makefile` line `B=`, and uncomment (activate) the line `B= -b`.

The function of the `Makefile` is to create the appropriate NIS maps for each of the databases listed under `all`. After passing through `makedbm` the data is collected in two files, `mapname.dir` and `mapname.pag`. Both files are in the `/var/yp/domainname` directory on the master server.

The `Makefile` builds `passwd` maps from the `/PWDIR/passwd`, `/PWDIR/shadow`, and `/PWDIR/security/passwd.adjunct` files, as appropriate.

How to Set Up the Master Server With `ypinit`

The `ypinit` script sets up master and slave servers and clients to use NIS. It also initially runs `make` to create the maps on the master server.

To use `ypinit` to build a fresh set of NIS maps on the master server, do the following.

▼ Setting up the master server using `ypinit`

1. **Become superuser on the master server.**
2. **Copy the contents of the `nsswitch.files` file to the `nsswitch.conf` file.**
`# cp /etc/nsswitch.files /etc/nsswitch.conf`
3. **Edit the `/etc/hosts` or `/etc/inet/ipnodes` file to add the name and IP address of each of the NIS servers.**
4. **Build new maps on the master server.**
`# /usr/sbin/ypinit -m`
5. **When `ypinit` prompts for a list of other machines to become NIS slave servers, type the name of the server you are working on, along with the names of your NIS slave servers.**

6. When `ypinit` asks whether you want the procedure to terminate at the first nonfatal error or continue despite nonfatal errors, type `y`.

When you choose `y`, `ypinit` exits upon encountering the first problem; you can then fix it and restart `ypinit`. This is recommended if you are running `ypinit` for the first time. If you prefer to continue, you can try to manually fix all problems that occur, and then restart `ypinit`.

Note – A nonfatal error can appear when some of the map files are not present. This is not an error that affects the functionality of NIS. You might need to add maps manually if they were not created automatically. Refer to “Default NIS Maps” on page 130 for a description of all default NIS maps.

7. `ypinit` asks whether the existing files in the `/var/yp/domainname` directory can be destroyed.

This message is displayed only if NIS has been previously installed.

8. After `ypinit` has constructed the list of servers, it invokes `make`.

This program uses the instructions contained in the `Makefile` (either the default one or the one you modified) located in `/var/yp`. The `make` command cleans any remaining comment lines from the files you designated. It also runs `makedbm` on the files, creating the appropriate maps and establishing the name of the master server for each map.

If the map or maps being pushed by the `Makefile` correspond to a domain other than the one returned by the command `domainname` on the master, you can make sure that they are pushed to the correct domain by starting `make` in the `ypinit` shell script with a proper identification of the variable `DOM`, as follows:

```
# make DOM=domainname password
```

This pushes the `password` map to the intended domain, instead of the domain to which the master belongs.

9. To enable NIS as the naming service, type the following.

```
# cp /etc/nsswitch.nis /etc/nsswitch.conf
```

This replaces the current switch file with the default NIS-oriented switch file. You can edit this file as necessary.

Master Supporting Multiple NIS Domains

Normally, an NIS master server supports only one NIS domain. However, if you are using a master server to support multiple domains, you must slightly modify the steps, as described in the previous section, when setting up the server to serve the additional domains.

Run the `domainname` command on the server. The domain name returned by the command is the server's default domain. The steps described in the previous section will work properly for setting up service for that domain. To configure service for any *other* domain, you must modify the `ypinit` shell script as follows.

```
# make DOM=correct-domain passwd
```

correct-domain is the name of the other domain that you are setting up service for, and `passwd` is the make target. This command pushes the `passwd` map to the intended domain, instead of the domain to which the master belongs.

Starting NIS Service on the Master Server

Now that the master maps are created, you can start the NIS daemons on the master server and begin service. To do this, you have to start `ypserv` on the server and run `ypbind`. When a client requests information from the server, `ypserv` is the daemon that answers information requests from clients after looking them up in the NIS maps.

There are two ways that NIS service can be started on a server:

- By automatically invoking the `/usr/lib/netsvc/yp/ypstart` script during the boot process
- Using `ypstart` from the command line

Starting NIS Service Automatically

After the NIS master server has been configured by running `ypinit`, `ypstart` is automatically invoked to start up `ypserve` when the machine is booted. See “How to Set Up the Master Server With `ypinit`” on page 144.

Starting and Stopping NIS From the Command Line

To begin NIS service from the command line, run the `ypstart` script.

```
# /usr/lib/netsvc/yp/ypstart
```

Note – Because there is a slight delay before `ypserv` is ready to respond to calls after startup, you should issue a three to five second sleep after `ypstart` when calling it from inside a program or script.

To stop NIS service, run the `ypstop` script.

```
# /usr/lib/netsvc/yp/ypstop
```

Setting Up NIS Slave Servers

Your network can have one or more slave servers. Having slave servers ensures the continuity of NIS services when the master server is not available.

Preparing a Slave Server

Before actually running `ypinit` to create the slave servers, you should run the `domainname` command on each NIS slave to make sure the domain name is consistent with the master server.

Note – Domain names are case-sensitive.

Make sure that the network is working properly before you configure an NIS slave server. In particular, check to be sure you can use `rcp` to send files from the master NIS server to NIS slaves.

Setting Up a Slave Server

The following procedure shows how to set up a slave server.

▼ How to Set up a Slave Server

1. **Become superuser.**
2. **Edit the `/etc/hosts` or `/etc/inet/ipnodes` file on the slave server to add the name and IP addresses of all the other NIS servers.**

3. Change directory to `/var/yp` on the slave server.

Note – You must first configure the new slave server as an NIS client so that it can get the NIS maps from the master for the first time. See “Setting Up NIS Clients” on page 149 for details.

4. To initialize the slave server as a client, type the following.

```
# /usr/sbin/ypinit -c
```

The `ypinit` command prompts you for a list of NIS servers. Enter the name of the local slave you are working on first, then the master server, followed by the other NIS slave servers in your domain in order from the physically closest to the furthest in network terms.

5. To determine if `ypbind` is running, type the following.

```
# ps -ef | grep ypbind
```

If a listing is displayed, `ypbind` is running.

6. If `ypbind` is running, stop it.

```
# /usr/lib/netsvc/yp/ypstop
```

7. Type the following to restart `ypbind`.

```
# /usr/lib/netsvc/yp/ypstart
```

8. To initialize this machine as a slave, type the following.

```
# /usr/sbin/ypinit -s master
```

Where *master* is the machine name of the existing NIS master server.

Repeat the procedures described in this section for each machine you want configured as an NIS slave server.

The following procedure shows how to start NIS on a slave server.

▼ How to Start NIS on a Slave Server

1. Become superuser.

2. Stop all existing `yp` processes.

```
# /usr/lib/netsvc/yp/ypstop
```

3. Start `ypserve` on the slave and run `ypbind`.

```
# /usr/lib/netsvc/yp/ypstart
```

Alternatively, you can reboot the slave server and the daemons start automatically.

Setting Up NIS Clients

Configuring a Machine to Use NIS

The two methods for configuring a machine to use NIS as its naming service are explained below.

- `ypinit`. The recommended method for configuring a client machine to use NIS is to login to the machine as `root` and run `ypinit -c`.

```
# ypinit -c
```

You will be asked to name NIS servers from which the client obtains naming service information. You can list as many master or slave servers as you want. The servers that you list can be located anywhere in the domain. It is a better practice to first list the servers closest (in network terms) to the machine, than those that are on more distant parts of the net.

- *Broadcast method*. An older method of configuring a client machine to use NIS to login to the machine as `root`, set the domain name with the `domainname` command, then run `ypbind`.

```
# domainname doc.com
```

```
# ypbind -broadcast
```

When you run `ypbind`, it searches the local subnet for an NIS server. If it finds a subnet, `ypbind` binds to it. This search is referred to as *broadcasting*. If there is no NIS server on the client's local subnet, `ypbind` fails to bind and the client machine is not able to obtain namespace data from the NIS service.

Administering NIS (Tasks)

This chapter describes how to administer NIS. The following topics are covered.

- “Password Files and Namespace Security” on page 151
- “Administering NIS Users” on page 152
- “Working With NIS Maps” on page 155
- “Updating and Modifying Existing Maps” on page 161
- “Adding a Slave Server” on page 166
- “Using NIS With C2 Security” on page 167
- “Changing a Machine’s NIS Domain” on page 168
- “Using NIS in Conjunction With DNS” on page 168
- “Turning Off NIS Services” on page 170

Password Files and Namespace Security

For security reasons, for these guidelines.

- It is best to limit access to the NIS maps on the master server.
- The files used to build the NIS password maps should not contain an entry for `root` to protect against unauthorized access. To accomplish this, the password files used to build the password maps should have the `root` entry removed from them and be located in a directory other than the master server’s `/etc` directory. This directory should be secured against unauthorized access.

For example, the master server password input files could be stored in a directory such as `/var/yp`, or any directory of your choice, as long as the file itself is not a link to another file and is specified in the Makefile. The `/usr/lib/netsvc/yp/ypstart` script automatically sets the correct directory option according to the configuration specified in your Makefile.

Note – In addition to the older Solaris 1 version `passwd` file format, this implementation of NIS accepts the Solaris 2 `passwd` and `shadow` file formats as input for building the NIS password maps.

Administering NIS Users

This section includes information about setting user passwords, adding new users to an NIS domain, and assigning users to `netgroups`.

Adding a New User to an NIS Domain

▼ How to Add a NIS User

1. **Become superuser on the master NIS server.**

2. **Create the new user's login ID with the `useradd` command.**

```
# useradd userID
```

userID is the login ID of the new user. This command creates entries in the `/etc/passwd` and `/etc/shadow` files on the master NIS server.

3. **Create the new user's initial password.**

To create an initial password that the new user can use to log in, run the `passwd` command.

```
# passwd userID
```

Where *userID* is the login ID of the new user. You will be prompted for the password to assign to this user.

This step is necessary because the password entry created by the `useradd` command is locked, which means that the new user cannot log in. By specifying an initial password, you unlock the entry.

4. **If necessary, copy the new entry into the server's `passwd` map input files.**

The map source files on your master server should be in a directory other than `/etc`. Copy and paste the new lines from the `/etc/passwd` and `/etc/shadow` files into the `passwd` map input files on the server. See "Password Files and Namespace Security" on page 151 for additional information.

For example, if you added the new user `brown`, the line from `/etc/passwd` that you would copy to your `passwd` input file would look like the following.

```
brown:x:123:10:User brown:/home/brown:/bin/csh:
```

The line for `brown` that you would copy from `/etc/shadow` would look like:

```
brown:W12345GkHic:6445:.....
```

5. **Make sure that the `Makefile` correctly specifies the directory where the password input file resides.**

6. **If appropriate, delete the new user's entries from `/etc/passwd` and `/etc/shadow` input files.**

For security reasons, do not keep user entries in the NIS master server `/etc/passwd` and `/etc/shadow` files. After copying the entries for the new user to the NIS map source files that are stored in some other directory, use the `userdel` command on the master server to delete the new user.

For example, to delete the new user `brown` from the master server's `/etc` files, you would enter the following.

```
# userdel brown
```

For more information about `userdel`, see the `userdel` man page.

7. **Update the NIS `passwd` maps.**

After you have updated the `passwd` input file on the master server, update the `passwd` maps by running `make` in the directory containing the source file.

```
#userdel brown # cd /var/yp # /usr/ccs/bin/make passwd
```

8. **Tell the new user the initial password you have assigned to his or her login ID.**

After logging in, the new user can run `passwd` at any time to establish a different password.

Setting User Passwords

Users run `passwd` to change their passwords.

```
% passwd username
```

Before users can change their passwords, you must start the `rpc.yppasswdd` daemon on the master server to update the password file. The commands for starting the daemon are already present in the `/usr/lib/netsvc/yp/ypstart` file.

The `rpc.yppasswdd` daemon is started automatically by `ypstart` on the master server. Notice that when the `-m` option is given to `rpc.yppasswdd`, a `make` is forced in `/var/yp` immediately following a modification of the file. If you want to avoid having this `make` take place each time the `passwd` file is changed, remove the `-m` option from the `rpc.yppasswdd` command in the `ypstart` script and control the pushing of the `passwd` maps through the `crontab` file.

Note – No arguments should follow the `rpc.yppasswd -m` command. Although you can edit the `ypstart` script file to achieve a different action, it is not recommended that you modify this file other than optionally removing the `-m` option. All commands and daemons invoked by this file with the proper set of command line parameters. If you choose to edit this file, be especially careful when editing the `rpc.yppasswdd` command. If you add an explicit call to the `passwd.adjunct` file, the exact `$PWDIR/security/passwd.adjunct` path must be used; otherwise, incorrect processing results.

Netgroups

NIS netgroups are groups (sets) of users or machines that you define for your administrative purposes. For example, you can create netgroups that do the following

- Define a set of users who can access a specific machine
- Define a set of NFS client machines to be given some specific file system access
- Define a set of users who are to have administrator privileges on all the machines in a particular NIS domain

Each netgroup is given a netgroup name. Netgroups do not directly set permissions or access rights. Instead, the netgroup names are used by other NIS maps in places where a user name or machine name would normally be used. For example, suppose you created a netgroup of network administrators called `netadmins`. To grant all members of the `netadmins` group access to a given machine, you need only add a `netadmin` entry to that machine's `/etc/passwd` file. Netgroup names can also be added to the `/etc/netgroup` file and propagated to the NIS netgroup map. See `netgroup(4)` for more detailed information on using netgroups.

On a network using NIS, the netgroup input file on the master NIS server is used for generating three maps: `netgroup`, `netgroup.byuser`, and `netgroup.byhost`. The `netgroup` map contains the basic information in the `netgroup` input file. The two other NIS maps contain information in a format that speeds lookups of netgroup information, given the machine or user.

Entries in the `netgroup` input file are in the format: *name ID*, where *name* is the name you give to a netgroup, and *ID* identifies a machine or user who belongs to the netgroup. You can specify as many IDs (members) to a netgroup as you want, separated by commas. For example, to create a netgroup with three members, the `netgroup` input file entry would be in the format: *name ID, ID, ID*. The member IDs in a `netgroup` input file entry are in the following format.

```
( [- |machine] , [- |user] , [domain] )
```

Where *machine* is a machine name, *user* is a user ID, and *domain* is the machine or user's NIS domain. The *domain* element is optional and should only be used to identify machines or users in some other NIS domain. The *machine* and *user* element of each member's entry are required, but a dash (-) is used to denote a null. There is no necessary relationship between the machine and user elements in an entry.

The following are two sample `netgroup` input file entries, each of which create a netgroup named `admins` composed of the users `hauri` and `juanita` who is in the remote domain `sales` and the machines `altair` and `sirius`.

```
admins (altair, hauri), (sirius,juanita,sales)
admins (altair,-), (sirius,-), (-,hauri), (-,juanita,sales)
```

Various programs use the netgroup NIS maps for permission checking during login, remote mount, remote login, and remote shell creation. These programs include `mountd`, `login`, `rlogin`, and `rsh`. The `login` command consults the netgroup maps for user classifications if it encounters netgroup names in the `passwd` database. The `mountd` daemon consults the netgroup maps for machine classifications if it encounters netgroup names in the `/etc/dfs/dfstab` file. `rlogin` and `rsh` In fact, any program that uses the `ruserok` interface consults the netgroup maps for both machine and user classifications if they encounter netgroup names in the `/etc/hosts.equiv` or `.rhosts` files.

If you add a new NIS user or machine to your network, be sure to add them to appropriate netgroups in the `netgroup` input file. Then use the `make` and `yppush` commands to create the netgroup maps and push them to all of your NIS servers. See `netgroup(4)` for detailed information on using netgroups and netgroup input file syntax.

Working With NIS Maps

Obtaining Map Information

Users can obtain information from and about the maps at any time by using the `ypcat`, `ypwhich`, and `ypmatch` commands. In the examples that follow, *mapname* refers both to the official name of a map and to its nickname, if any.

To list all the values in a map, type the following.

```
% ypcat mapname
```

To list both the keys and the values (if any) in a map, type the following.

```
% ypcat -k mapname
```

To list all the map nicknames, type any of the following commands.

```
%ypcat -x
```

```
% ypwhich -x
```

```
% ypmatch -x
```

To list all the available maps and their master(s), type the following.

```
% ypwhich -m
```

To list the master server for a particular map, type the following.

```
% ypwhich -m mapname
```

To match a key with an entry in a map, type the following.

```
% ypmatch key mapname
```

If the item you are looking for is not a key in a map, type the following.

```
% ypcat mapname | grep item
```

where *item* is the information for which you are searching. To obtain information about other domains, use the `-d domainname` options of these commands.

If the machine requesting information for a domain other than its default does not have a binding for the requested domain, `ypbind` consults the `/var/yp/binding/domainname/ypservers` file for a list of servers for that domain. If this file does not exist it issues an RPC broadcast for a server. In this case, there must be a server for the requested domain on the same subnet as the requesting machine.

Changing a Map's Master Server

To change the master server for a selected map, you first have to build the map on the new NIS master. Since the old master server name occurs as a key-value pair in the existing map (this pair is inserted automatically by `makedbm`), copying the map to the new master or transferring a copy to the new master with `ypxfr` is insufficient. You have to reassociate the key with the new master server name. If the map has an ASCII source file, you should copy this file to the new master.

▼ How to Change a Map's Master Server

1. **Become superuser.**
2. **Log in to the new master as superuser and type the following.**

```
newmaster# cd /var/yp
```

3. The **Makefile** must have an entry for the new map before you specify the map to make. If this is not the case, edit the **Makefile** now, using a map called **sites.byname**.

4. To update or remake the map, type the following.

```
newmaster# make sites.byname
```

5. If the old master remains an NIS server, remote log in (**rlogin**) to the old master and edit **Makefile**. Make sure you comment out the section of the **Makefile** that made **sites.byname** so that it is no longer made there.

6. If **sites.byname** only exists as an **ndbm** file, remake it on the new master by disassembling a copy from any NIS server, then running the disassembled version through **makedbm**.

```
newmaster# cd /var/yp
```

```
newmaster# ypcat sites.byname | makedbm -domain/sites.byname
```

After making the map on the new master, you must send a copy of the new map to the other slave servers. Do not use **yppush**, because the other slaves will try to get new copies from the old master, rather than the new one. A typical method for circumventing this is to transfer a copy of the map from the new master back to the old master. To do this, become **superuser** on the old master server and type the following.

```
oldmaster# /usr/lib/netsvc/yp/yxfr -h newmaster sites.byname
```

Now it is safe to run **yppush**. Any remaining slave servers still believe that the old master is the current master and will attempt to get the current version of the map from the old master. When clients do so, they will get the new map, which names the new master as the current master.

If this method fails, you can log in as **root** on each NIS server and execute the **yxfr** command shown above.

Modifying Configuration Files

NIS intelligently parses the setup files. Although this makes NIS administration easier, it does make the behavior of NIS more sensitive to changes in the setup and configuration files.

Use the procedures in this section when modifying any of the following.

- `/var/yp/Makefile` to add or delete supported maps
- Adding or deleting `/etc/resolv.conf` to allow or deny DNS forwarding
- Adding or deleting `$(PWDIR)/security/passwd.adjunct` to allow or deny C2 security. (`$(PWDIR)` is defined in `/var/yp/Makefile`.)

▼ How to Modify Configuration Files.

1. **Become superuser.**
2. **Stop the NIS server.**
`# /etc/init.d/ypstop`
3. **Make the necessary changes to your files.**
4. **Restart the NIS server.**
`# /etc/init.d/yp start`

You do not have to stop and start NIS when changing NIS maps or the map source files.

Keep in mind the following.

- Deleting a map or source file from an NIS master server does not automatically result in corresponding deletions from slave servers. You must delete maps and source files from slave servers by hand.
- New maps do not automatically get pushed to existing slave servers. You must run `ypxfr` from the slaves.

Modifying and Using the Makefile

You can modify the `Makefile` provided by default in `/var/yp` to suit your needs. (Be sure to keep an unmodified copy of the original `Makefile` for future reference.) You can add or delete maps, and you can change the names of some of the directories.

To add a new NIS map, you must get copies of the `ndbm` files for the map into the `/var/yp/domainname` directory on each of the NIS servers in the domain. This is normally done for you by the `Makefile`. After deciding which NIS server is the master of the map, modify the `Makefile` on the master server so that you can conveniently rebuild the map. Different servers can be masters of different maps, but in most cases this leads to administrative confusion. Try to set only one server as the master of all maps.

Typically a human-readable text file is filtered through `awk`, `sed`, or `grep` to make it suitable for input to `makedbm`. Refer to the default `Makefile` for examples. See the `make(1S)` for general information about the `make` command.

Use the mechanisms already in place in the `Makefile` when deciding how to create dependencies that `make` will recognize. Be aware that `make` is very sensitive to the presence or absence of tabs at the beginning of lines within the dependency rules. A missing tab can invalidate an entry that is otherwise well formed.

Adding an entry to the `Makefile` involves the following.

- Adding the name of the database to the `all` rule

- Writing the `time` rule
- Adding the rule for the database

For example, in order for the `Makefile` to work on automounter input files, you would have to add the `auto_direct.time` and `auto_home.time` maps to the NIS database.

To add these maps to the NIS database:

▼ How to Modify the `Makefile`

1. **Log in as a superuser.**
2. **Modify the line that starts with the word `all` by adding the name(s) of the database you want to add:**

```
all: passwd group hosts ethers networks rpc services protocols \
    netgroup bootparams aliases netid netmasks \
    auto_direct auto_home auto_direct.time auto_home.time
```

The order of the entries is not relevant, but the blank space at the beginning of the continuation lines must be a Tab, not spaces.

3. **Add the following lines at the end of the `Makefile`:**

```
auto_direct: auto_direct.time
auto_home: auto_home.time
```

4. **Add an entry for `auto_direct.time` in the middle of the file.**

```
auto_direct.time: $(DIR)/auto_direct
@(while read L; do echo $$L; done < $(DIR)/auto_direct
$(CHKPIPE) | \ (sed -e "/^#/d" -e "s/#.*$$//" -e "/^ *$$/d"
$(CHKPIPE) | \ $(MAKEDBM) - $(YPBDDIR)/$(DOM)/auto_direct;
@touch auto_direct.time;
@echo "updated auto_direct";
@if [ ! $(NOPUSH) ]; then $(YPPUSH) auto_direct; fi
@if [ ! $(NOPUSH) ]; then echo "pushed auto_direct"; fi
```

where

- `CHKPIPE` makes certain that the operations to the left of the pipe (`|`) are successfully completed before piping the results to next commands. If the operations to the left of the pipe do not successfully complete, the process is terminated with a NIS make terminated message.
- `NOPUSH` prevents the makefile from calling `yppush` to transfer the new map to the slave servers. If `NOPUSH` is not set, the push is done automatically.

The `while` loop at the beginning is designed to eliminate any backslash-extended lines in the input file. The `sed` script eliminates comment and empty lines.

The same procedure should be followed for all other automounter maps, such as `auto_home`, or any other nondefault maps.

5. Run make.

```
# make mapname
```

Where *mapname* is the name of the map you want to make.

If you do not want the Makefile to produce maps for a specific database, edit the Makefile as follows.

1. Delete the name of the database from the all rule.

2. Delete or comment out the database rule for the database you want to delete.

For example, to delete the `hosts` database, the `hosts.time` entry should be removed.

3. Remove the time rule.

For example, to delete the `hosts` database, the `hosts: hosts.time` entry should be removed.

4. Remove the map from the master and slave servers.

Changing Makefile Macros/Variables

You can change the settings of the variables defined at the top of the Makefile by changing the value to the right of the equal sign (=). For instance, if you do not want to use the files located in `/etc` as input for the maps, but you would rather use files located in another directory, such as `/var/etc/domainname`, you should change `DIR` from `DIR=/etc` to `DIR=/var/etc/domainname`. You should also change `PWDIR` from `PWDIR=/etc` to `PWDIR=/var/etc/domainname`.

The variables are the following.

- *DIR*= The directory containing all of the NIS input files except `passwd` and `shadow`. The default value is `/etc`. Since it is not good practice to use the files in the master server's `/etc` directory as NIS input files, you should change this value.
- *PWDIR*= The directory containing the `passwd` and `shadow` NIS input files. Since it is not good practice to use the files in the master server's `/etc` directory as NIS input files, you should change this value.
- *DOM*= The NIS domain name. The default value of `DOM` is set using the `domainname` command. However, most NIS commands use the current machine's domain which is set in the machine's `/etc/defaultdomain` file.

Updating and Modifying Existing Maps

After you have installed NIS, you might discover that some maps require frequent updating while others never need to change. For example, the `passwd.byname` map can change frequently on a large company's network, while the `auto_master` map changes little, if at all.

As mentioned in "Default NIS Maps" on page 130, the default location of the default NIS maps is on the master server in `/var/yp/domainname`, where *domainname* is the name of the NIS domain. When you need to update a map, you can use one of two updating procedures, depending on whether or not it is a default map.

- A default map is a map in the default set created by `ypinit` from the network databases.
- Nondefault maps can be any of the following.
 - Maps included with an application purchased from a vendor
 - Maps created specifically for your site
 - Maps created from a nontext file

The following sections explain how to use various updating tools. In practice, you might decide to only use them if you add nondefault maps or change the set of NIS servers after the system is already up and running.

Updating Maps Supplied with the Default Set

Use the following procedure for updating maps supplied with the default set.

▼ How to update maps supplied with the default set

1. **Become a superuser on the master server.**
Always modify NIS maps only on the master server.
2. **Edit the source file for the map you want to change, whether that file resides in `/etc` or in some other directory of your choice.**

3. **Type the following.**

```
# cd /var/yp
# make mapname
```

The `make` command then updates your map according to the changes you made in its corresponding file. It also propagates the changes among the other servers.

Propagating an NIS Map

After a map is changed, the `Makefile` uses `yppush` to propagate a new map to the slave servers (unless `NOPUSH` is set in the `Makefile`). It does this by informing the `ypserv` daemon and sending a map transfer request. The `ypserv` daemon on the slave then starts a `ypxfr` process, which in turn contacts the `ypxfrd` daemon on the master server. Some basic checks are made (for example did the map really change?) and then the map is transferred. `ypxfr` on the slave then sends a response to the `yppush` process indicating whether the transfer succeeded.

Note – The above procedure will *not* work for newly created maps that do not yet exist on the slave servers. New maps must be sent to the slave servers by running `ypxfr` on the slaves.

Occasionally, maps fail to propagate and you must to use `ypxfr` manually to send new map information. You can choose to use `ypxfr` in two different ways: periodically through the root `crontab` file, or interactively on the command line. These approaches are discussed in the following sections.

Using cron for Map Transfers

Maps have different rates of change. For instance, some might not change for months at a time, such as `protocols.byname` among the default maps and `auto_master` among the nondefault maps; but `passwd.byname` can change several times a day. Scheduling map transfer using the `crontab` command allows you to set specific propagation times for individual maps.

To periodically run `ypxfr` at a rate appropriate for the map, the root `crontab` file on each slave server should contain the appropriate `ypxfr` entries. `ypxfr` contacts the master server and transfers the map only if the copy on the master server is more recent than the local copy.

Note – If your master server runs `rpc.yppasswd` with the default `-m` option, then each time someone changes their `yp` password, the `passwd` daemon runs `make`, which rebuilds the `passwd` maps.

Using Shell Scripts With cron and ypxfr

As an alternative to creating separate `crontab` entries for each map, you might prefer to have the root `crontab` command run a shell script that periodically updates all maps. Sample map-updating shell scripts are in the `/usr/lib/netsvc/yp` directory.

The script names are `ypxfr_1perday`, `ypxfr_1perhour`, and `ypxfr_2perday`. You can modify or replace these shell scripts to fit your site requirements. Example 9-1 shows the default `ypxfr_1perday` shell script.

EXAMPLE 9-1 `ypxfr_1perday` Shell Script

```
#!/bin/sh
#
# ypxfr_1perday.sh - Do daily yp map check/updates
PATH=/bin:/usr/bin:/usr/lib/netsvc/yp:$PATH
export PATH
# set -xv
ypxfr group.byname
ypxfr group.bygid
ypxfr protocols.byname
ypxfr protocols.bynumber
ypxfr networks.byname
ypxfr networks.byaddr
ypxfr services.byname
ypxfr ypservers
```

This shell script updates the maps once per day, if the root `crontab` is executed daily. You can also have scripts that update maps once a week, once a month, once every hour, and so forth, but be aware of the performance degradation implied in frequently propagating the maps.

Run the same shell scripts as root on each slave server configured for the NIS domain. Alter the exact time of execution from one server to another to avoid bogging down the master.

If you want to transfer the map from a particular slave server, use the `-h machine` option of `ypxfr` within the shell script. Here is the syntax of the commands you put in the script.

```
# /usr/lib/netsvc/yp/ypxfr -h machine [ -c ] mapname
```

Where *machine* is the name of the server with the maps you want to transfer, and *mapname* is the name of the requested map. If you use the `-h` option without specifying a machine, `ypxfr` tries to get the map from the master server. If `ypserv` is not running locally at the time `ypxfr` is executed, you must use the `-c` flag so that `ypxfr` does not send a clear current map request to the local `ypserver`.

You can use the `-s domain` option to transfer maps from another domain to your local domain. These maps should be the same across domains. For example, two domains might share the same `services.byname` and `services.byaddr` maps. Alternatively, you can use `rcp`, or `rdist` for more control, to transfer files across domains.

Directly Invoking `ypxfr`

The second method of invoking `ypxfr` is to run it as a command. Typically, you do this only in exceptional situations—for example, when setting up a temporary NIS server to create a test environment or when trying to quickly get an NIS server that has been out of service consistent with the other servers.

Logging `ypxfr` Activity

The transfer attempts and results of `ypxfr` can be captured in a log file. If a file called `/var/yp/ypxfr.log` exists, results are appended to it. No attempt to limit the size of the log file is made. To prevent it from growing indefinitely, empty it from time to time by typing the following.

```
# cd /var/yp# cp ypxfr.log ypxfr.log.old
# cat /dev/null > /var/yp/ypxfr.log
```

You can have `crontab` execute these commands once a week. To turn off logging, remove the log file.

Modifying Default Maps

To update a nondefault map, you must do the following.

1. Create or edit its corresponding text file.
2. Build (or rebuild) the new or updated map. There are two ways to build a map.
 - Use the `Makefile`. Using the `Makefile` is the preferred method of building a non-default map. If the map has an entry in the `Makefile`, run `make name` where `name` is the name of map you want to build. If the map does not have a `Makefile` entry, try to create one following the instructions in “Modifying and Using the `Makefile`” on page 158.
 - Use the `/usr/sbin/makedbm` program. `makedbm(1M)` fully describes this command.

Using `makedbm` to Modify a Non-Default Map

There are two different methods for using `makedbm` to modify maps if you do not have an input file:

- Redirect the `makedbm -u` output to a temporary file, modify the file, then use the modified file as input to `makedbm`.
- Have the output of `makedbm -u` operated on within a pipeline that feeds into `makedbm`. This is appropriate if you can update the disassembled map with either `awk`, `sed`, or a `cat` append.

Creating New Maps from Text Files

Assume that a text file `/var/yp/mymap.asc` was created with an editor or a shell script on the master. You want to create an NIS map from this file and locate it in the *homedomain* subdirectory. To do this, type the following on the master server.

```
# cd /var/yp # makedbm mymap.asc homedomain/mymap
```

The *mymap* map now exists on the master server in the directory *homedomain*. To distribute the new map to slave servers run `ypxfr`.

Adding Entries to a File-Based Map

Adding entries to *mymap* is simple. First, you must modify the text file `/var/yp/mymap.asc`. If you modify the actual `dbm` files without modifying the corresponding text file, the modifications are lost. Then run `makedbm` as shown above.

Creating Maps From Standard Input

When no original text file exists, create the NIS map from the keyboard by typing input to `makedbm`, as shown below (end with Control-D).

```
ypmaster# cd /var/yp
```

```
ypmaster# ypmaster# makedbm - homedomain/mymapkey1 value1 key2 value2 key3  
value3
```

Modifying Maps Made From Standard Input

If you later need to modify the map, you can use `makedbm` to disassemble the map and create a temporary text intermediate file. To disassemble the map and create a temporary file, type the following.

```
% cd /var/yp
```

```
% makedbm -u homedomain/mymap > mymap.temp
```

The resulting temporary file `mymap.temp` has one entry per line. You can edit this file as needed, using any text editor.

To update the map, give the name of the modified temporary file to `makedbm` by typing the following.

```
% makedbm mymap.temp homedomain/mymap
```

```
% rm mymap.temp
```

Then propagate the map to the slave servers, by becoming root and typing the following.

```
# yppush mymap
```

The preceding paragraphs explained how to use `makedbm` to create maps; however, almost everything you actually have to do can be done by `ypinit` and `Makefile` unless you add nondefault maps to the database or change the set of NIS servers after the system is already up and running.

Whether you use the `Makefile` in `/var/yp` or some other procedure the goal is the same. A new pair of well-formed `dbm` files must end up in the `maps` directory on the master server.

Adding a Slave Server

After NIS is running, you might need to create an NIS slave server that you did not include in the initial list given to `ypinit`.

To add a NIS slave server:

▼ How to Add a Slave Server

1. **Log in to the master server as a superuser.**

2. **Change to the NIS domain directory.**

```
# cd /var/yp/domainname
```

3. **Disassemble the `ypservers` file.**

```
# makedbm -u ypservers >/tmp/temp_file
```

The `makedbm` command converts `ypservers` from `ndbm` format to a temporary ASCII file `/tmp/temp_file`.

4. **Edit the `/tmp/temp_file` file using a text editor. Add the name of the new slave server to the list of servers. Then save and close the file.**

5. **Run the `makedbm` command with `temp_file` as the input file and `ypservers` as the output file.**

```
# makedbm /tmp/temp_file ypservers
```

`makedbm` then converts `ypservers` back into `ndbm` format.

6. Verify that the `ypservers` map is correct (since there is no ASCII file for `ypservers`) by typing the following on the slave.

```
slave3# makedbm -u ypservers
```

The `makedbm` command displays each entry in `ypservers` on your screen.

Note – If a machine name is not in `ypservers`, it will not receive updates to the map files because `yppush` consults this map for the list of slave servers.

7. Set up the new slave server's NIS domain directory by copying the NIS map set from the master server.

To do this, become superuser on the new NIS slave and run the `ypinit` and `ypbind` commands.

```
slave3# cd /var/yp
```

```
slave3# ypinit -c list of servers
```

```
slave3# /usr/lib/netsvc/yp/ypbind
```

8. Initialize this machine as a slave.

```
slave3# /usr/sbin/ypinit -s ypmaster
```

where `ypmaster` is the machine name of the existing NIS master server.

9. Run `ypstop` to stop the machine running as an NIS client.

```
# /usr/lib/netsvc/yp/ypstop
```

10. Run `ypstart` to start NIS slave service.

```
# /usr/lib/netsvc/yp/ypstart
```

Using NIS With C2 Security

If the `$PWDIR/security/passwd.adjunct` file is present, C2 security is started automatically. (`$PWDIR` is defined in `/var/yp/Makefile`.) The C2 security mode uses the `passwd.adjunct` file to create the `passwd.adjunct` NIS map. In this implementation, NIS allows you to use both the `passwd.adjunct` file and `shadow` file to manage security. The `passwd.adjunct` file is processed only when you type the following.

```
# make passwd.adjunct
```

The `make passwd` command processes the `passwd` map only, not the `passwd.adjunct` map when you run `make` manually in the C2 security mode.

Changing a Machine's NIS Domain

To change the NIS domain name of a machine, do the following.

▼ How to Change a Machine's NIS Domain Name

1. **Become superuser.**
2. **Edit the machine's `/etc/defaultdomain` file, exchanging its present contents with the new domain name for the machine.**
For example, if the current domain name is `sales.doc.com`, you might change it to `research.doc.com`.
3. **Run `domainname 'cat /etc/defaultdomain'`**
4. **Set the machine up as an NIS client, slave, or master server.**
See for Chapter 8 for details.

Using NIS in Conjunction With DNS

Typically, NIS clients are configured with the `nsswitch.conf` file to use only NIS for machine name and address lookups. If this type of lookup fails, an NIS server can forward these lookups to DNS.

▼ Configuring Machine Name and Address Lookup Through NIS and DNS

1. **Log into the machine and become a superuser.**
2. **The two map files, `hosts.byname` and `hosts.byaddr` must include the `YP_INTERDOMAIN` key. To test this key, edit the `Makefile` and modify the following lines.**

```
#B=-b
B=
to
B=-b
#B=
```

makedbm will now start with the `-b` flag when it makes the maps, and the `YP_INTERDOMAIN` key will be inserted into the `ndbm` files.

3. Run make to rebuild maps.

```
# /usr/ccs/bin/make hosts
```

4. Check that all the NIS server's `/etc/resolv.conf` files point to valid nameservers.

Note – If you have NIS servers that are not running Solaris, Release 2, make sure `YP_INTERDOMAIN` exists in the hosts maps.

5. To enable DNS forwarding, stop each server.

```
# /usr/lib/netsvc/yp/ypstop
```

6. Restart each server.

```
# /usr/lib/netsvc/yp/ypstart
```

In this implementation of NIS, `ypstart` will automatically start the `ypserv` daemon with the `-d` option to forward requests to DNS.

Dealing with Mixed NIS Domains

If the master and slave servers are not both running Solaris 2, refer to the following table for how to avoid potential problems. The notation “4.0.3+” refers to the that and later releases of SunOS. `makedm -b` is a reference to the “-B” variable in the `Makefile`.

TABLE 9-1 NIS/DNS in Heterogeneous NIS Domains

Slave	Master		
	4.0.3+		Solaris
4.0.3+	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>ypserv -d</code> <code>ypxrf -b</code>
Solaris NIS	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>ypserv -d</code> Slave: <code>ypxfr</code> with <code>resolv.conf</code> or <code>ypxfr -b</code>

Turning Off NIS Services

If `ypserv` on the master is disabled, you can no longer update any of the NIS maps. If you choose to turn off NIS on a network currently running it, you can disable NIS after the next reboot by renaming the `ypbind` file to `ypbind.orig`, as follows.

```
% mv /usr/lib/netsvc/yp/ypbind /usr/lib/netsvc/yp/ypbind.orig
```

To disable NIS after the next reboot on a particular NIS slave or master, type the following on the server in question.

```
% mv /usr/lib/netsvc/yp/ypserv /usr/lib/netsvc/yp/ypserv.orig
```

To stop NIS immediately, type the following.

```
% /usr/lib/netsvc/yp/ypstop
```

The NIS service is automatically restarted after the next reboot unless the `ypbind` and `ypserv` files are renamed as described above.

NIS Troubleshooting

This chapter explains how to resolve problems encountered on networks running NIS. It covers problems seen on an NIS client and those seen on an NIS server.

Before trying to debug an NIS server or client, review Chapter 7 which explains the NIS environment. Then look for the subheading in this section that best describes your problem.

NIS Binding Problems

Symptoms

Common symptoms of NIS binding problems include the following.

- Messages saying that `ybind` can't find or communicate with a server.
- Messages saying that server not responding.
- Messages saying that NIS is unavailable
- Commands on a client limp along in background mode or function much slower than normal
- Commands on a client hang. Sometimes commands hang even though the system as a whole seems fine and you can run new commands
- Commands on a client crash with obscure messages, or no message at all

NIS Problems Affecting One Client

If only one or two clients are experiencing symptoms that indicate NIS binding difficulty, the problems probably are on those clients. If many NIS clients are failing to bind properly, the problem probably exists on one or more of the NIS servers. See “NIS Problems Affecting Many Clients” on page 175.

ypbind Not Running on Client

One client has problems, but other clients on the same subnet are operating normally. On the problem client, run `ls -l` on a directory, such as `/usr`, that contains files owned by many users, including some not in the client `/etc/passwd` file. If the resulting display lists file owners who are not in the local `/etc/passwd` as numbers, rather than names, this indicates that NIS service is not working on the client.

These symptoms usually mean that the client `ypbind` process is not running. Run `ps -e` and check for `ypbind`. If you do not find it, log in as superuser and start `ypbind` by typing:

```
client# /usr/lib/netsvc/yp/ypstart
```

Missing or Incorrect Domain Name

One client has problems, the other clients are operating normally, but `ypbind` is running on the problem client. The client might have an incorrectly set domain.

On the client, run the `domainname` command to see which domain name is set.

```
client7# domainname neverland.com
```

Compare the output with the actual domain name in `/var/yp` on the NIS master server. The actual NIS domain is shown as a subdirectory in the `/var/yp` directory.

```
Client7# ls /var/yp...
```

```
-rwxr-xr-x 1 root Makefile drwxr-xr-x 2 root  
binding drwx----- 2 root doc.com ...
```

If the domain name returned by running `domainname` on a machine is not the same as the server domain name listed as a directory in `/var/yp`, the domain name specified in the machine's `/etc/defaultdomain` file is incorrect. Log in as superuser and correct the client's domain name in the machine's `/etc/defaultdomain` file. This assures that the domain name is correct every time the machine boots. Now reboot the machine.

Note – The domain name is case-sensitive.

Client Not Bound to Server

If your domain name is set correctly, `ybind` is running, and commands still hang, then make sure that the client is bound to a server by running the `ypwhich` command. If you have just started `ybind`, then run `ypwhich` several times (typically, the first one reports that the domain is not bound and the second succeeds normally).

No Server Available

If your domain name is set correctly, `ybind` is running, and you get messages indicating that the client cannot communicate with a server, this might indicate a number of different problems:

- Does the client have a `/var/yp/binding/domainname/ypservers` file containing a list of servers to bind to? If not, run `ypinit -c` and specify in order of preference the servers that this client should bind to.
- If the client does have a `/var/yp/binding/domainname/ypservers` file, are there enough servers listed in it if one or two become unavailable? If not, add additional servers to the list by running `ypinit -c`.
- If none of the servers listed in the client's `ypservers` file are available, the client searches for an operating server using broadcast mode. If there is a functioning server on the client's subnet, the client will find it (though performance might be slowed during the search). If there are no functioning servers on the client's subnet can solve the problem in several ways:
 - If the client has no server on the subnet and no route to one, you can install a new slave server on that subnet.
 - You can make sure your routers are configured to pass broadcast packets so that the client can use broadcast to find a server on another subnet. You can use the `netstat -r` command to verify the route.
 - If there should be a route, but it is not working, make sure that the route daemon `in.routed/in.rdisc` is running. If it is not running, start it.

Note – For reasons of security and administrative control it is preferable to specify the servers a client is to bind to in the client's `ypservers` file rather than have the client search for servers through broadcasting. Broadcasting slows down the network, slows the client, and prevents you from balancing server load by listing different servers for different clients.

- Do the servers listed in a clients `ypservers` file have entries in the `/etc/hosts` file? If not, add the servers to the NIS maps `hosts` input file and rebuild your maps by running `ypinit -c` or `ypinit -s` as described “Working With NIS Maps” on page 155.
- Is the `/etc/nsswitch.conf` file set up to consult the machine’s local `hosts` file in addition to NIS? See Chapter 2 for more information on the switch.
- Is the `/etc/nsswitch.conf` file set up to consult `files` first for `services` and `rpc`?

ypwhich Displays Are Inconsistent

When you use `ypwhich` several times on the same client, the resulting display varies because the NIS server changes. This is normal. The binding of the NIS client to the NIS server changes over time when the network or the NIS servers are busy. Whenever possible, the network becomes stable at a point where all clients get acceptable response time from the NIS servers. As long as your client machine gets NIS service, it does not matter where the service comes from. For example, an NIS server machine can get its own NIS services from another NIS server on the network.

When Server Binding is Not Possible

In extreme cases where local server binding is not possible, use of the `ypset` command can temporarily allow binding to another server, if available, on another network or subnet. However, in order to use the `-ypset` option, `ypbind` must be started with either the `-ypset` or `-ypsetme` options.

Note – For security reasons, the use of the `-ypset` and `-ypsetme` options should be limited to debugging purposes under controlled circumstances. Use of the `-ypset` and `-ypsetme` options can result in serious security breaches because while the daemons are running, anyone can alter server bindings causing trouble for others and permitting unauthorized access to sensitive data. If you must start `ypbind` with these options, once you have fixed the problem you should kill `ypbind` and restart it again without those options.

ypbind Crashes

If `ypbind` crashes almost immediately each time it is started, look for a problem in some other part of the system. Check for the presence of the `rpcbind` daemon by typing the following.

```
% ps -ef | grep rpcbind
```

If `rpcbind` is not present or does not stay up or behaves strangely, consult your RPC documentation.

You might be able to communicate with `rpcbind` on the problematic client from a machine operating normally. From the functioning machine, type the following.

```
% rpcinfo client
```

If `rpcbind` on the problematic machine is fine, `rpcinfo` produces the following output.

```
program    version  netid    address    service    owner
...
100007     2       udp     0.0.0.0.2.219  ypbind     superuser
100007     1       udp     0.0.0.0.2.219  ypbind     superuser
100007     1       tcp     0.0.0.0.2.220  ypbind     superuser
100007     2       tcp     0.0.0.0.128.4  ypbind     superuser
100007     2       ticotsord  \000\000\020H  ypbind     superuser
100007     2       ticots   \000\000\020K  ypbind     superuser
...
```

Your machine will have different addresses. If the addresses are not displayed, `ypbind` has been unable to register its services. Reboot the machine and run `rpcinfo` again. If the `ypbind` processes are there and they change each time you try to restart `/usr/lib/netsvc/yp/ypbind`, reboot the system, even if the `rpcbind` daemon is running.

NIS Problems Affecting Many Clients

If only one or two clients are experiencing symptoms that indicate NIS binding difficulty, the problems probably are on those clients. See “NIS Problems Affecting One Client” on page 172. If many NIS clients are failing to bind properly, the problem probably exists on one or more of the NIS servers.

`rpc.yppasswdd` Considers a Non-restricted Shell Which Begins with `r` to be Restricted

1. create `/etc/default/yppasswdd` that contains a special string:
"check_restricted_shell_name=1"
2. If the "check_restricted_shell_name=1" string is commented out, the 'r' check will no occur.

Network or Servers are Overloaded

NIS can hang if the network or NIS servers are so overloaded that `ypserv` cannot get a response back to the client `ypbind` process within the time-out period.

Under these circumstances, every client on the network experiences the same or similar problems. In most cases, the condition is temporary. The messages usually go away when the NIS server reboots and restarts `yppserv`, or when the load on the NIS servers or network itself decreases.

Server Malfunction

Make sure the servers are up and running. If you are not physically near the servers, use the `ping` command.

NIS Daemons Not Running

If the servers are up and running, try to find a client machine behaving normally, and run the `ypwhich` command. If `ypwhich` does not respond, kill it. Then log in as `root` on the NIS server and check if the NIS `ypbind` process is running by entering the following.

```
# ps -e | grep yp
```

Note – Do not use the `-f` option with `ps` because this option attempts to translate user IDs to names which causes more naming service lookups that might not succeed.

If either the `ypbind` or `yppserv` daemons are not running, kill them and then restart them by entering the following.

```
# /usr/lib/netsvc/yp/ypstop
```

```
# /usr/lib/netsvc/yp/ypstart
```

If both the `yppserv` and `ypbind` processes are running on the NIS server, use `ypwhich`.

If `ypwhich` does not respond, `yppserv` has probably hung and should be restarted. While logged in as `root` on the server, kill `yppserv` and restart it by typing the following.

```
# /usr/lib/netsvc/yp/ypstop # /usr/lib/netsvc/yp/ypstart
```

Servers Have Different Versions of an NIS Map

Because NIS propagates maps among servers, occasionally you might find different versions of the same map on various NIS servers on the network. This version discrepancy is normal and acceptable if the differences do not last for more than a short time.

The most common cause of map discrepancy is that something is preventing normal map propagation. For example, an NIS server or router between NIS servers is down. When all NIS servers and the routers between them are running, `ypxfr` should succeed.

If the servers and routers are functioning properly, check the following:

- Log `ypxfr` output (see “Logging `ypxfr` Output” on page 177).
- Check the control files (see “Check the `crontab` File and `ypxfr` Shell Script” on page 177).
- Check the `ypservers` map on the master. See “Check the `ypservers` Map” on page 178.

Logging `ypxfr` Output

If a particular slave server has problems updating maps, log in to that server and run `ypxfr` interactively. If `ypxfr` fails, it tells you why it failed, and you can fix the problem. If `ypxfr` succeeds, but you suspect it has occasionally failed, create a log file to enable logging of messages. To create a log file, enter the following on the slave..

```
ypslave# cd /var/yp
ypslave# touch ypxfr.log
```

This creates a `ypxfr.log` file that saves all output from `ypxfr`.

The output resembles the output `ypxfr` displays when run interactively, but each line in the log file is time stamped. (You might see unusual ordering in the time-stamps. That is okay—the time-stamp tells you when `ypxfr` started to run. If copies of `ypxfr` ran simultaneously but their work took differing amounts of time, they might actually write their summary status line to the log files in an order different from that which they were invoked.) Any pattern of intermittent failure shows up in the log.

Note – When you have fixed the problem, turn off logging by removing the log file. If you forget to remove it, it continues to grow without limit.

Check the `crontab` File and `ypxfr` Shell Script

Inspect the root `crontab` file, and check the `ypxfr` shell script it invokes. Typographical errors in these files can cause propagation problems. Failures to refer to a shell script within the `/var/spool/cron/crontabs/root` file, or failures to refer to a map within any shell script can also cause errors.

Check the *ypservers* Map

Also, make sure that the NIS slave server is listed in the *ypservers* map on the master server for the domain. If it is not, the slave server still operates perfectly as a server, but *yppush* does not propagate map changes to the slave server.

Work Around

If the NIS slave server problem is not obvious, you can work around it while you debug using *rcp* or *ftp* to copy a recent version of the inconsistent map from any healthy NIS server. The following shows how to transfer the problem map.

```
ypslave# rcp ypmaster:/var/yp/mydomain/map.* /var/yp/mydomain
```

The *** character has been escaped in the command line, so that it will be expanded on *ypmaster*, instead of locally on *ypslave*.

ypserv Crashes

When the *ypserv* process crashes almost immediately, and does not stay up even with repeated activations, the debug process is virtually identical to that described in “*ypbind* Crashes” on page 174. Check for the existence of the *rpcbind* daemon as follows.

```
ypserver% ps -e | grep rpcbind
```

Reboot the server if you do not find the daemon. Otherwise, if the daemon is running, type the following and look for similar output.

```
% rpcinfo -p ypserv
```

%	program	vers	proto	port	service
100000	4	tcp	111	portmapper	
100000	3	tcp	111	portmapper	
100068	2	udp	32813	cmsd	
...					
100007	1	tcp	34900	ypbind	
100004	2	udp	731	ypserv	
100004	1	udp	731	ypserv	
100004	1	tcp	732	ypserv	
100004	2	tcp	32772	ypserv	

Your machine might have different port numbers. The four entries representing the *ypserv* process are the following.

100004	2	udp	731	ypserv
100004	1	udp	731	ypserv
100004	1	tcp	732	ypserv
100004	2	tcp	32772	ypserv

If there are no entries, and `ypserv` is unable to register its services with `rpcbind`, reboot the machine. If there are entries, de-register the service from `rpcbind` before restarting `ypserv`. To de-register the service from `rpcbind`, on the server type the following.

```
# rpcinfo -d number 1
```

```
# rpcinfo -d number 2
```

where *number* is the ID number reported by `rpcinfo` (100004, in the example above).

PART **IV**

Configuring Sun ONE Directory Server 5.1

The following chapter discusses how to configure Sun™ ONE Directory Server 5.1 (formerly iPlanet™ Directory Server 5.1).

Configuring Sun™ ONE Directory Server 5.1 (Tasks)

This chapter discusses how to configure Sun ONE Directory Server 5.1. You must complete the procedures contained in this chapter before you can go on to configure Sun ONE Directory Server 5.1 for use with Solaris LDAP naming services clients.

Note – If you are using a directory server other than Sun ONE Directory Server 5.1, skip this chapter. See “Generic Directory Server Requirements” on page 286 for list of basic requirements for other directory servers when used in conjunction with Solaris LDAP naming services clients.

Refer to the following Sun ONE manuals for in-depth information regarding Sun ONE Directory Server 5.1.

- *Sun ONE Directory Server 5.1 Schema Reference Guide*
- *Sun ONE Directory Server 5.1 Deployment Manual*
- *Sun ONE Directory Server 5.1 Configuration, Command, and File Reference*
- *Sun ONE Directory Server 5.1 Administrator’s Guide*

This chapter covers the following topics.

- “Preparing for Configuration” on page 184
- “Configuration Components” on page 184
- “Configuration Choices” on page 185
- “Choosing Unique Port Numbers” on page 185
- “Choosing User and Group” on page 186
- “Defining Authentication Entities” on page 186
- “Choosing Your Directory Suffix” on page 187
- “Choosing the Location of the Configuration Directory” on page 188
- “Choosing the Location of the User Directory” on page 189
- “Choosing the Administration Domain” on page 189
- “Configuration Process Overview” on page 190
- “Using Express Configuration” on page 191
- “Using Typical Configuration” on page 192

Preparing for Configuration

Before you begin configuring Sun ONE Directory Server 5.1, you should have an understanding of the various components and the design and configuration decisions you need to make.

To help you configure Sun ONE Directory Server 5.1, you should be familiar with the concepts contained in the following sections.

- Components
- Configuration Decisions
- Configuration Process Overview
- Configuration Privileges

The *Sun ONE Directory Server 5.1 Deployment Guide* contains basic directory concepts as well as guidelines to help you design and successfully deploy your directory service.

Configuration Components

Sun ONE Directory Server 5.1 contains the following software components, which are installed by default when you install the entire Solaris disk suite.

- Sun ONE Console

The Sun ONE Console provides the common user interface for all Sun ONE server products. From it you can perform common server administration functions such as stopping and starting servers and installing new server instances. Sun ONE Console can be installed as a stand-alone application on any machine. You can also install it on your network and use it to manage remote servers.
- Administration Server

The Administration Server is a common front-end to all Sun ONE servers. It receives communications from Sun ONE Console and passes those communications on to the appropriate Sun ONE server.
- Sun ONE Directory Server 5.1

Sun ONE Directory Server 5.1 is a high-performance, scalable LDAP server with an on-disk database. The Sun ONE Directory Server 5.1 runs as the `ns-slapd` process on Solaris. This is the server that manages the directory databases and responds to client requests. Sun ONE Directory Server 5.1 is a required component.

Configuration Choices

During Directory Server configuration, you are prompted for basic information. Decide how you are going to configure these basic parameters before you begin the configuration process. You are prompted for some or all of following information, depending on the type of configuration that you decide to perform the following.

- Port number
- Users and groups to run the server as
- Your directory suffix
- Several different authentication user IDs
- The administration domain

Choosing Unique Port Numbers

Port numbers can be any number from 1 to 65535. Keep the following in mind when choosing a port number for your Sun ONE Directory Server 5.1.

- The standard Sun ONE Directory Server 5.1 (LDAP) port number is 389.
- Port 636 is reserved for LDAP over SSL. Therefore, do not use port number 636 for your standard LDAP configuration, even if 636 is not already in use. You can also use LDAP over TLS on the standard LDAP port.
- Port numbers between 1 and 1024 have been assigned to various services by the Internet Assigned Numbers Authority. Do not use port numbers below 1024 other than 389 or 636 for directory services as they will conflict with other services. Additionally, port numbers below 1024 are accessible by root only.
- Sun ONE Directory Server 5.1 must run as root using either port 389 or 636.
- Make sure the ports you choose are not already in use. Additionally, if you are using both LDAP and LDAPS communications, make sure the port numbers chosen for these two types of access are not identical.

Note – If the LDAP naming service clients are using SSL encryption, you *must* use the default port numbers 389 and 636, so that the server runs as `root`. See “Transport Layer Security (TLS)” on page 214 for information on Transport Layer Security.

For information on how to set up LDAP over SSL (LDAPS) for Sun ONE Directory Server 5.1, see the *Sun ONE Directory Server 5.1 Administrator's Guide*.

Choosing User and Group

For security reasons, it is always best to run UNIX-based production servers with normal user privileges. That is, you do not want to run Directory Server with root privileges. However, you will have to run Directory Server with root privileges if you are using the default Directory Server ports. If Directory Server is to be started by Administration Server, Administration Server must run either as root or as the same user as Sun ONE Directory Server 5.1.

You must therefore decide what user accounts you will use for the following purposes.

- The user and group under which you will run Sun ONE Directory Server 5.1.
If you will not be running Sun ONE Directory Server 5.1 as root, it is strongly recommended that you create a user account for all Sun ONE servers. You should not use any existing operating system account, and must not use the `nobody` account. Also you should create a common group for the Sun ONE Directory Server 5.1 files; again, you must not use the `nobody` group
- The user and group under which you will run Administration Server.
For configurations that use the default port numbers, this must be root. However, if you use ports over 1024, then you should create a user account for all Sun ONE servers, and run Administration Server as this account.

As a security precaution, when Administration Server is being run as root, it should be shut it down when it is not in use.

You should use a common group for all Sun ONE servers, such as `gid Sun ONE`, to ensure that files can be shared between servers when necessary.

Before you can install Sun ONE Directory Server 5.1 and Administration Server, you must make sure that the user and group accounts you will use exist on your system.

Defining Authentication Entities

As you configure Sun ONE Directory Server 5.1 and Administration Server, you will be asked for various user names, distinguished names (DN), and passwords. This list of login and bind entities will differ depending on the type of configuration that you are performing.

- Directory Manager DN and password
The Directory Manager DN is the special directory entry to which access control does not apply. Think of the directory manager as your directory's superuser. (In previous releases of Sun ONE Directory Server, the Directory Manager DN was known as the root DN).

The default Directory Manager DN is `cn=Directory Manager`. Because the Directory Manager DN is a special entry, the Directory Manager DN does not have to conform to any suffix configured for your Sun ONE Directory Server 5.1. Therefore, you must not manually create an actual Sun ONE Directory Server 5.1

entry that has the same DN as the directory manager DN.

The Directory Manager password must be at least 8 characters long, and is limited to ASCII letters, digits, and symbols.

Note – It is wise to use the same Directory Manager DN and password for *all* of your LDAP servers, especially if you have set the replicas to follow referrals to the master server during client add and modify operations.

- Configuration Directory Administrator ID and password

The configuration directory administrator is the person responsible for managing all Sun ONE servers accessible through Sun ONE Console. If you log in with this user ID, then you can administer any Sun ONE server that you can see in the server topology area of Sun ONE Console.

For security, the configuration directory administrator should not be the same as the directory manager. The default configuration directory administrator ID is `admin`.

- The Administration Server User and password

You are prompted for this only during custom configurations. The Administration Server user is the special user that has all privileges for the local Administration Server. Authentication as this person allows you to administer all Sun ONE servers stored on this server.

Administration Server user ID and password is used only when Sun ONE Directory Server 5.1 is down and you are unable to log in as the configuration directory administrator. The existence of this user ID means that you can access Administration Server and perform disaster recovery activities such as starting Sun ONE Directory Server 5.1, reading log files, and so forth.

Normally, Administration Server user and password should be identical to the configuration directory administrator ID and password.

Choosing Your Directory Suffix

A directory suffix is the directory entry that represents the first entry in a directory tree. You will need at least one directory suffix for the tree that will contain your enterprise's data. It is common practice to select a directory suffix that corresponds to the DNS host name used by your enterprise. For example, if your organization uses the DNS name `example.com`, then select a suffix of `dc=example,dc=com`.

For more information on planning the suffixes for your directory service, see the *Sun ONE Directory Server 5.1 Deployment Guide*.

Choosing the Location of the Configuration Directory

Many Sun ONE servers including Directory Server 5.1 use an instance of Sun ONE Directory Server 5.1 to store configuration information. This information is stored in the `o=NetScapeRoot` directory tree. It does not need to be held on the same Sun ONE Directory Server 5.1 as your directory data. Your configuration directory is Sun ONE Directory Server 5.1 that contains the `o=NetScapeRoot`.

If you are installing Sun ONE Directory Server 5.1 only to support other Sun ONE servers, then that Sun ONE Directory Server 5.1 is your configuration directory. If you are installing Sun ONE Directory Server 5.1 to use as part of a general directory service, then you will have multiple Sun ONE Directory Server 5.1s installed in your enterprise and you must decide which one will host the configuration directory tree, `o=NetScapeRoot`. You must make this decision before you install any Sun ONE servers (including Sun ONE Directory Server 5.1).

For ease of upgrades, you should use a Sun ONE Directory Server 5.1 instance that is dedicated to supporting the `o=NetScapeRoot` tree; this server instance should perform no other function with regard to managing your enterprise's directory data. Also, do not use port 389 for this server instance because doing so could prevent you from installing a Sun ONE Directory Server 5.1 on that host that can be used for management of your enterprise's directory data.

Because the configuration directory normally experiences very little traffic, you can allow its server instance to coexist on a machine with another more heavily loaded Sun ONE Directory Server 5.1 instance. However, for very large sites that are installing a large number of Sun ONE servers, you may want to dedicate a low-end machine to the configuration directory so as to not hurt the performance of your other production servers. Sun ONE server configurations result in write activities to the configuration directory. For large enough sites, this write activity could result in a short-term performance hit to your other directory activities.

Also, as with any directory configuration, consider replicating the configuration directory to increase availability and reliability. See the *Sun ONE Directory Server 5.1 Deployment Guide* for information on using replication and DNS round robins to increase directory availability.



Caution – If the configuration directory tree is corrupted, you might need to reinstall all other Sun ONE servers that are registered in that configuration directory. Remember the following guidelines when dealing with the configuration directory.

- Always back up your configuration directory after you install a new Sun ONE server
 - Never change the host name or port number used by the configuration directory
 - Never directly modify the configuration directory tree. Only the setup program for the various Sun ONE servers should ever modify the configuration
-

Choosing the Location of the User Directory

Just as the configuration directory is Sun ONE Directory Server 5.1 that is used for Sun ONE server administration, the user directory is Sun ONE Directory Server 5.1 that contains the entries for users and groups in your enterprise.

For most directory configurations, the user directory and the configuration directory should be two separate server instances. These server instances can be installed on the same machine, but for best results you should consider placing the configuration directory on a separate machine.

Between your user directory and your configuration directory, it is your user directory that will receive the overwhelming percentage of the directory traffic. For this reason, you should give the user directory the greatest computing resources. Because the configuration directory should receive very little traffic, it can be installed on a machine with very low-end resources.

Also, you should use the default directory ports (389 and 636) for the user directory. If your configuration directory is managed by a server instance dedicated to that purpose, you should use some non-standard port for the configuration directory.

You cannot install a user directory until you have installed a configuration directory somewhere on your network.

Choosing the Administration Domain

The administration domain allows you to logically group Sun ONE servers together so that you can more easily distribute server administrative tasks. A common scenario is for two divisions in a company to each want control of their individual Sun ONE servers. However, you may still want some centralized control of all the servers in your enterprise. Administration domains allow you to meet these conflicting goals.

Administration domains have the following qualities.

- All servers share the same configuration directory, regardless of the domain to which they belong
- Servers in two different domains may use two different user directories for authentication and user management
- The configuration directory administrator has complete access to all installed Sun ONE servers, regardless of the domain to which they belong
- Each administration domain can be configured with an administration domain owner. This owner has complete access to all the servers in the domain but does not have access to the servers in any other administration domain
- The administration domain owner can grant individual users administrative access on a server by server basis within the domain

For many configurations, you can have just one administration domain. In this case, choose a name that is representative of your organization. For other configurations, you may want different domains because of the demands at your site. In the latter case, try to name your administration domains after the organizations that will control the servers in that domain.

For example, if you are an ISP and you have three customers for whom you are installing and managing Sun ONE servers, create three administration domains each named after a different customer.

Configuration Process Overview

You can use one of several configuration processes to install Sun ONE Directory Server 5.1. Each one guides you through the configuration process and ensures that you configure the various components in the correct order.

The following sections outline the configuration processes available.

Selecting an Configuration Process

You can configure Sun ONE Directory Server 5.1 software using one of the four different configuration methods provided in the setup program.

- *Express configuration*
Use this if you are installing for the purposes of evaluating or testing Sun ONE Directory Server 5.1. See "Using Express Configuration" on page 191.
- *Typical configuration*
Use this if you are performing a normal install of Sun ONE Directory Server 5.1. See "Using Typical Configuration" on page 192.

- *Custom configuration*

In Sun ONE Directory Server 5.1, the custom configuration process is very similar to the typical configuration process. The main difference is that the custom configuration process will allow you to import an LDIF file to initialize the user directory database that is created by default.

Beyond determining which type of configuration process you will use, the process for configuring Sun ONE Directory Server 5.1 is as follows:

1. Plan your directory service. By planning your directory tree in advance, you can design a service that is easy to manage and easy to scale as your organization grows. For guidance on planning your directory service, refer to the *Sun ONE Directory Server 5.1 Deployment Guide*.
2. Configure your Sun ONE Directory Server 5.1 as described in this chapter.
3. Create the directory suffixes and databases. You do not have to populate your directory now; however, you should create the basic structure for your tree, including all major roots and branch points. For information about the different methods of creating a directory entry, refer to the *Sun ONE Directory Server 5.1 Administrator's Guide*.
4. Create additional Sun ONE Directory Server 5.1 instances and set up replication agreements between your Sun ONE Directory Server 5.1 instances to ensure availability of your data.

Using Express and Typical Configuration

Using Express Configuration

Use express configuration if you are installing Sun ONE Directory Server 5.1 to evaluate or test the product. Because express configuration does not offer you the choice of selecting your server port number or your directory suffix, you should not use it for production configurations. To perform an express configuration, do the following.

▼ How to configure Sun ONE Directory Server 5.1 using express configuration

1. **Become superuser.**
2. **Run the Sun ONE Directory Server 5.1 program by typing the following.**

```
# /usr/sbin/directoryserver setup
```

3. When you are prompted for what you want to install, hit enter for [the default] Sun ONE servers.
4. When you are prompted for the type of configuration, choose Express.
5. For the user and group to run the servers as, enter the identity that you want this server to run as.
6. For Configuration Directory Administrator ID and password, enter the name and password that you will log in as when you want to authenticate to the console with full privileges. Think of this as the root or superuser identity for the Sun ONE Console.

The server is then minimally configured, and started. You are told what host and port number on which the Administration Server is listening.

Note the following about your new Sun ONE Directory Server 5.1 configuration.

- Sun ONE Directory Server 5.1 is listening on port 389
- The server is configured to use the following suffixes

`dc=your_machine s_DNS_domain_name`

That is, if your machine is named `test.example.com`, then you have the suffix `dc=example`, `dc=com` configured for this server.

`o=NetscapeRoot`

Do not modify the contents of the directory under the `o=NetscapeRoot` suffix. Either create data under the first suffix, or create a new suffix to be used for this purpose. For details on how to create new suffixes for your Sun ONE Directory Server 5.1, see the *Sun ONE Directory Server 5.1 Administrator's Guide*.

Using Typical Configuration

Most first time configurations of Sun ONE Directory Server 5.1 can be performed using the Typical option of the setup program.

▼ How to configure Sun ONE Directory Server 5.1 using typical configuration

1. Become superuser.
2. Run the Sun ONE Directory Server 5.1 program.
`# /usr/sbin/directoryserver setup`
3. When you are prompted for what you want to install, hit enter for [the default] Sun ONE Servers.

4. When you are prompted for **Directory Suite and Administration Services**, hit enter to select all [the default].
5. Hit enter to select all **Directory Suite** components.
6. Hit enter to select all **Administration** components.
7. When prompted for the **hostname**, select the default [the host] or enter an alternative fully qualified domain name.



Caution – Note that the default hostname may be incorrect if the installer cannot locate a DNS name for your system. For example, you might not have a DNS name if your system uses NIS. The hostname must be a fully qualified host and domain name. If the default hostname is not a fully qualified host and domain name, configuration will fail.

8. The setup program then asks you for the **System User** and the **System Group** names. Enter the identity under which you want the servers to run.
9. For the **configuration directory**, select the default if this directory will host your **o=NetscapeRoot** tree. Otherwise, enter **Yes**. You will then be asked for the contact information for the configuration directory.
If the server you are currently installing is not the configuration directory, then the configuration directory must exist before you can continue this configuration.
10. The setup program then asks if the server you are currently installing will be the one for your user data. For most cases, you can select the default. However, if you intend this server instance to be used as a configuration directory only, then you should enter **Yes**.
11. For the **Sun ONE Directory Server 5.1 port**, select the default (389) unless you already have another application using that port.
12. For the **Sun ONE Directory Server 5.1 Identifier**, enter a unique value (normally the default is sufficient).

This value is used as part of the name of the directory in which the Sun ONE Directory Server 5.1 instance is installed. For example, if your machine's host name is `phonebook`, then this name is the default and selecting it will cause the Sun ONE Directory Server 5.1 instance to be installed into a directory labeled `slapd-phonebook`.



Caution – The Sun ONE Directory Server 5.1 identifier must not contain a period. For example, `example.server.com` is not a valid server identifier name.

13. **For Configuration Directory Administrator ID and password, enter the name and password that you will log in as when you want to authenticate to the console with full privileges.**
14. **For a directory suffix, enter a distinguished name meaningful to your enterprise.**
This string is used to form the name of all your organization's directory entries. Therefore, pick a name that is representative of your organization. It is recommended that you pick a suffix that corresponds to your internet DNS name.
For example, if your organization uses the DNS name example.com, then enter `dc=example, dc=com` here.
15. **For Directory Manager DN, enter the distinguished name that you will use when managing the contents of your directory with unlimited privileges.**

Note – Any Distinguished Names must be entered in the UTF-8 character set encoding. Older encodings such as ISO-8859-1 are not supported.

In former releases of Sun ONE Directory Server 5.1, the Directory Manager was known as the root DN. This is the entry that you bind to the directory as when you want access control to be ignored. This distinguished name can be short and does not have to conform to any suffix configured for your directory. However, it should not correspond to an actual entry stored in your directory.

16. **For the Directory Manager password, enter a value that is at least 8 characters long.**
17. **For Administration Domain, enter the domain that you want this server to belong to.**
The name you enter should be a unique string that is descriptive of the organization responsible for administering the domain.
18. **For the administration port number, enter a value that is not in use (for example, you might want to use the value 5100 to indicate a 5.1 Sun ONE Directory Server 5.1). Be sure to record this value somewhere you can remember.**
19. **For the user you want to run Administration Server as, enter root, the default.**
The server is then minimally configured, and started. You are told what host and port number Administration Server is listening on. The server is configured to use the following suffixes.
 - The suffix that you configured
 - `o=NetscapeRoot`Do not modify the contents of the directory under the `o=NetscapeRoot` suffix. Either create data under the first suffix, or create a new suffix to be used for this purpose. For details on how to create new suffixes for your Sun ONE Directory Server 5.1, see the *Sun ONE Directory Server 5.1 Administrator's Guide*.

PART **V**

LDAP Naming Services Setup and Administration

This part provides an overview of the LDAP naming services. Additionally, it covers the setup, configuration, administration, and troubleshooting of LDAP naming services in the Solaris operating environment, with a focus on the use of Sun™ ONE Directory Server 5.1 (formerly iPlanet™ Directory Server 5.1).

Introduction to LDAP Naming Services (Overview/Reference)

The LDAP chapters describe how to set up a Solaris LDAP naming services client to work with Sun™ ONE Directory Server 5.1 (formerly iPlanet™ Directory Server 5.1). A brief description of generic directory server requirements appears in Chapter 18.

Note – A directory server is not necessarily an LDAP server. However, in the context of these chapters, the term “directory server” is synonymous with “LDAP server.”

Audience Assumptions

The LDAP naming services chapters are written for system administrators who already have a working knowledge of LDAP. Following is a partial list of concepts with which you must be very familiar. Otherwise, you might have difficulty using this guide to deploy LDAP naming services in the Solaris environment.

- LDAP Information Model (entries, object classes, attributes, types, values)
- LDAP Naming Model (Directory Information Tree (DIT) structure)
- LDAP Functional Model (search parameters: base object (DN), scope, size limit, time limit, filters (browsing indexes for the Sun ONE Directory Server), attribute list)
- LDAP Security Model (authentication methods, access control models)
- Overall planning and design of an LDAP directory service, including how to plan the data and how to design the DIT, topology, replication, and security

Suggested Background Reading

To learn more about any of the aforementioned concepts or to study LDAP and the deployment of directory services in general, refer to the following sources:

- *Understanding and Deploying LDAP Directory Services* by Timothy A. Howes, Ph.D. and Mark C. Smith

In addition to providing a thorough treatment of LDAP directory services, this book includes useful case studies on deploying LDAP. Examples of deployments include a large university, a large multinational enterprise, and an enterprise with an extranet.

- *Sun ONE Directory Server 5.1 Deployment Guide*, which is included on the documentation CD

This guide provides a foundation for planning your directory, including directory design, schema design, the directory tree, topology, replication, and security. The last chapter provides sample deployment scenarios to help you plan both simple, smaller-scale deployments and complex worldwide deployments.

- *Sun ONE Directory Server 5.1 Administrator's Guide*, which is included on the documentation CD

Additional Prerequisites

If you are transitioning from using NIS+ to using LDAP, refer to Chapter 19. Complete the transition before proceeding with these chapters.

If you need to install Sun ONE Directory Server 5.1, refer to the *Sun ONE Directory Server 5.1 Installation Guide*.

LDAP Naming Services Compared to Other Naming Services

The following table shows a comparison between the FNS, DNS, NIS, NIS+, and LDAP naming services.

	DNS	NIS	NIS+	FNS	LDAP
Namespace	Hierarchical	Flat	Hierarchical	Hierarchical	Hierarchical

	DNS	NIS	NIS+	FNS	LDAP
Data Storage	Files/resource records	2 column maps	Multi—columned tables	Maps	Directories (varied) Indexed database
Servers	Master/slave	Master/slave	Root master/ non-root master; primary/ secondary; cache/stub	N/A	Master/replica Multi master replica
Security	None	None (root or nothing)	DES-Authentication	None (root or nothing)	SSL, varied
Transport	TCP/IP	RPC	RPC	RPC	TCP/IP
Scale	Global	LAN	LAN	Global (with DNS) /LAN	Global

Using Fully Qualified Domain Names

Unlike NIS or NIS+ clients, an LDAP client always returns a fully qualified domain name (FQDN) for a host name. The LDAP FQDN is similar to the FQDN returned by DNS. For example, suppose your domain name is the following:

```
west.example.net
```

Both `gethostbyname()` and `getipnodebyname()` return the FQDN version when looking up the host name *server*:

```
server.west.example.net
```

Also, if you use interface-specific aliases such as `asserver-#`, a long list of fully qualified host names are returned. If you are using host names to share file systems or have other such checks, you must account for the checks. For example, if you assume non-FQDNs for local hosts and FQDNs only for remote DNS-resolved hosts, you must account for the difference. If you set up LDAP with a different domain name from DNS, the same host might end up with two different FQDNs, depending on the lookup source.

Advantages of LDAP Naming Services

- LDAP enables you to consolidate information by replacing application-specific databases, which reduces the number of distinct databases to be managed.
- LDAP allows for more frequent data synchronization between masters and replicas.
- LDAP is multi-platform and multi-vendor compatible.

Restrictions of LDAP Naming Services

Following are some restrictions associated with LDAP naming services:

- Clients prior to Solaris 8 are not supported..
- An LDAP server cannot be its own client.
- Setting up and managing an LDAP naming services is more complex and requires careful planning.

Note – A directory server (an LDAP server) *cannot* be its own client. That is, you cannot configure the machine that is running the directory server software to become an LDAP naming services client.

New LDAP Naming Services Features for Solaris 9

- The configuration of LDAP directory server setup has been simplified with the use of `idsconfig`.
- A more robust security model that supports strong authentication and Transport Layer Security (TLS) encrypted sessions. A client's proxy credentials are *no longer* stored in a client's profile on the directory server.
- The `ldapaddent` command allows you to populate and dump data onto the server.
- Service search descriptors and attribute mapping.
- New profile schema.

Transitioning from NIS+ to LDAP

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment.

For more information, go

to <http://www.sun.com/directory/nisplus/transition.html>.

For information about transitioning from NIS+ to LDAP, refer to Chapter 19.

LDAP Naming Services Setup (Task Map)

Task	For Instructions
Plan the network model	“Planning the Network Model” on page 224
Plan the DIT	“Planning the Directory Information Tree (DIT)” on page 224
Set up replica servers	“Replica Servers” on page 226
Plan the security model	“Planning the Security Model” on page 227
Choose client profiles and default attribute values	“Planning Client Profiles and Default Attribute Values” on page 227
Plan the data population	“Planning the Data Population” on page 228
Configure Sun ONE Directory Server 5.1 prior to using it with LDAP naming services	“Using Express and Typical Configuration” on page 191
Set up Sun ONE Directory Server 5.1 for use with LDAP naming clients	Chapter 15
Manage printer entries	“Managing Printer Entries” on page 241
Initialize an LDAP client	“Initializing a Client” on page 246

Initialize a client using profiles	<i>"Using Profiles to Initialize a Client"</i> on page 246
Initialize a client manually	<i>"Initializing a Client Manually"</i> on page 247
Uninitialize a client	<i>"Uninitializing a Client"</i> on page 248
Use service search descriptors to modify client profiles	<i>"Using Service Search Descriptors to Modify Client Access to Various Services"</i> on page 235
Retrieve naming service information	<i>"Retrieving LDAP Naming Services Information"</i> on page 251
Customize a client environment	<i>"Customizing the Client Environment"</i> on page 252

Basic Components and Concepts (Overview)

This chapter covers the following topics.

- “LDAP Data Interchange Format (LDIF)” on page 203
- “Default Directory Information Tree (DIT)” on page 206
- “Default Schema” on page 207
- “Service Search Descriptors (SSDs) and Schema Mapping” on page 208
- “Client Profiles” on page 210
- “ldap_cachemgr Daemon” on page 213
- “LDAP Naming Services Security Model” on page 214

LDAP Data Interchange Format (LDIF)

LDIF is a text-based format for describing directory service information. Using LDIF you can move information from one directory to another. The following are examples of LDIF format for each service. Use `ldaplist(1)` with the `-l` option to display the following information.

```
% ldaplist -l hosts myhost
```

```
hosts
```

```
dn: cn=myhost+ipHostNumber=7.7.7.115,ou=Hosts,dc=mydc,dc=mycom,dc=com
cn: myhost
iphostnumber: 7.7.7.115
objectclass: top
objectclass: device
objectclass: ipHost
description: host 1 - floor 1 - Lab a - building b
```

```
% % ldaplist -l passwd user1
```

```
passwd
```

```
dn: uid=user1,ou=People,dc=mydc,dc=mycom,dc=com
uid: user1
cn: user1
userpassword: {crypt}duTx91g7PoNzE
uidnumber: 199995
gidnumber: 20
gecos: Joe Smith [New York]
homedirectory: /home/user1
loginshell: /bin/csh
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
```

```
% ldaplist -l services name
```

```
services
```

```
dn: cn=name+ipServiceProtocol=udp,ou=Services,dc=mydc,dc=mycom,dc=com
cn: name
cn: nameserver
ipServiceProtocol: udp
ipServicePort: 42
objectclass: top
objectclass: ipService
```

```
% ldaplist -l group mygroup
```

```
group
```

```
dn: cn=mygroup,ou=Group,dc=mydc,dc=mycom,dc=com
cn: mygroup
gidnumber: 4441
memberuid: user1
memberuid: user2
memberuid: user3
userpassword: {crypt}duTx91g7PoNzE
objectclass: top
objectclass: posixGroup
```

```
% ldaplist -l netgroup mynetgroup
```

```
netgroup
```

```
cn=mynetgroup,ou=netgroup,dc=central,dc=sun,dc=com
objectclass=nisNetgroup
objectclass=top
cn=mynetgroup
nisnetgrouptriple=(user1..mydc.mycom.com,-,)
nisnetgrouptriple=(user1.,-,)
membernisnetgroup=mylab
```

```
% ldaplist -l networks 200.20.20.0
```

```
networks
```

```
dn: ipNetworkNumber=200.20.20.0,ou=Networks,dc=mydc,dc=mycom,dc=com
cn: mynet-200-20-20
ipnetworknumber: 200.20.20.0
objectclass: top
objectclass: ipNetwork
description: my Lab Network
ipnetmasknumber: 255.255.255.0
```

```
% ldaplist -l netmasks 201.20.20.0
```

```
netmasks
```

```
dn: ipNetworkNumber=201.20.20.0,ou=Networks,dc=mydc,dc=mycom,dc=com
cn: net-201
ipnetworknumber: 201.20.20.0
objectclass: top
objectclass: ipNetwork
description: my net 201
ipnetmasknumber: 255.255.255.0
```

```
% ldaplist -l rpc ypserv
```

```
rpc
```

```
dn: cn=ypserv,ou=Rpc,dc=mydc,dc=mycom,dc=com
cn: ypserv
cn: ypprog
oncrpcnumber: 100004
objectclass: top
objectclass: oncRpc
```

```
% ldaplist -l protocols tcp
```

```
protocols
```

```
dn: cn=tcp,ou=Protocols,dc=mydc,dc=mycom,dc=com
cn: tcp
ipprotocolnumber: 6
description: transmission control protocol
objectclass: top
objectclass: ipProtocol
```

```
% ldaplist -l bootparams myhost
```

```
bootparams
```

```
dn: cn=myhost,ou=Ethers,dc=mydc,dc=mycom,dc=com
bootparameter: root=boothost:/export/a/b/c/d/e
objectclass: top
objectclass: device
objectclass: bootableDevice
cn: myhost
```

```
% ldaplist -l ethers myhost
```

```

ethers

dn: cn=myhost,ou=Ethers,dc=mydc,dc=mycom,dc=com
macaddress: 8:1:21:71:31:c1
objectclass: top
objectclass: device
objectclass: ieee802Device
cn: myhost

% ldaplist -l publickey myhost

publickey

dn: cn=myhost+ipHostNumber=200.20.20.99,ou=Hosts,dc=mydc,dc=mycom,dc=com
cn: myhost
iphostnumber: 200.20.20.99
description: Joe Smith
nispublickey: 9cc01614d929848849add28d090acdaa1c78270aeec969c9
nissecretkey: 999999998769c999c39e7a6ed4e7afd687d4b99908b4de99
objectclass: top
objectclass: NisKeyObject
objectclass: device
objectclass: ipHost

% ldaplist -l aliases myname

aliases

dn: mail=myname,ou=aliases,dc=mydc,dc=mycom,dc=com
cn: myname
mail: myname
objectclass: top
objectclass: mailgroup
mgrprfc822mailmember: my.name

```

Default Directory Information Tree (DIT)

By default, Solaris LDAP clients access the information assuming that the DIT has a given structure. For each domain supported by the LDAP server, there is an assumed subtree with an assumed structure. This default structure, however, can be overridden by specifying Service Search Descriptors (SSDs). For a given domain, the default DIT will have a base container that holds a number of subtrees containing entries for a specific information type. See the following table for the names of these subtrees.

TABLE 13-1 DIT Default Locations

Default Container	Information Type
ou=Ethers	bootparams(4), ethers(4)
ou=Group	group(4)
ou=Hosts	hosts(4), ipnodes(4), publickey for hosts
ou=Aliases	aliases(4)
ou=Netgroup	netgroup(4)
ou=Networks	networks(4), netmasks(4)
ou=People	passwd(1), shadow(4), user_attr(4), audit_user(4), publickey for users
ou=printers	printers(4)
ou=Protocols	protocols(4)
ou=Rpc	rpc(4)
ou=Services	services(4)
ou=SolarisAuthAttr	auth_attr(4)
ou=SolarisProfAttr	prof_attr(4), exec_attr(4)
ou=projects	project
automountMap=auto_*	auto_*

Default Schema

Schemas are definitions describing what types of information can be stored as entries in an LDAP directory. To support LDAP naming clients, the directory server's schema might need to be extended. Detailed information about IETF and Solaris specific schemas is included in Chapter 18. The various RFCs can also be accessed on the IETF Web site <http://www.ietf.org>.

Service Search Descriptors (SSDs) and Schema Mapping

Note – If you use schema mapping, you must do so in a very careful and consistent manner.

As previously discussed, LDAP naming services expect, by default, the DIT to be structured in a certain way. If you want, you can instruct the Solaris LDAP naming service to search in other locations than the default locations in the DIT. Additionally, you can specify that different attributes and object classes be used in place of those specified by the default schema. For a list of default filters, see “Default Filters Used by LDAP Naming Services” on page 287.

SSDs

The `serviceSearchDescriptor` attribute defines how and where an LDAP naming service client should search for information for a particular service. The `serviceSearchDescriptor` contains a service name, followed by one or more semicolon-separated base-scope-filter triples. These base-scope-filter triples are used to define searches only for the specific service and are searched in order. If multiple base-scope-filters are specified for a given service, then when that service looks for a particular entry, it will search in each base with the specified scope and filter.

Note – The default location is not searched for a service (database) with an SSD unless it is included in the SSD. Unpredictable behavior will result if multiple SSDs are given to a service.

In the following example, the Solaris LDAP naming service client performs a one-level search in `ou=west,dc=example,dc=com` followed by a one-level search in `ou=east,dc=example,dc=com` for the `passwd` service. To look up the `passwd` data for a user’s username, the default LDAP filter `(&(objectClass=posixAccount)(uid=username))` is used for each BaseDN.

```
serviceSearchDescriptor: passwd:ou=west,dc=example,dc=com;ou=east,dc=example,dc=com
```

In the following example, the Solaris LDAP naming service client would perform a subtree search in `ou=west,dc=example,dc=com` for the `passwd` service. To look up the `passwd` data for user `username`, the subtree `ou=west,dc=example,dc=com` would be searched with the LDAP filter `(&(fulltimeEmployee=TRUE)(uid=username))`.

```
serviceSearchDescriptor: passwd:ou=west,dc=example,  
dc=com?sub?fulltimeEmployee=TRUE
```

It is also possible to associate multiple containers with a particular service type.

For example, the following service search descriptor specifies that the three containers, `ou=myuser,dc=example,dc=com`, `ou=newuser,dc=example,dc=com`, and `ou=extuser,dc=example,dc=com` are searched for the password entries. Note that a trailing `'/'` implies that the `defaultSearchBase` is appended to the relative base in the SSD.

```
defaultSearchBase: dc=example,dc=com  
serviceSearchDescriptor: \  
passwd:ou=myuser;ou=newuser,ou=extuser,dc=example,dc=com
```

Attribute Map

The Solaris LDAP naming service allows one or more attribute names to be remapped for any of its services. (The Solaris LDAP client uses the well-known attributes documented in Chapter 18.) If you map an attribute, you must be sure that the attribute has the same meaning and syntax as the original attribute. Note that mapping the `userPassword` attribute might cause problems.

There are a couple of reasons you might want to use schema mappings.

- You want to map attributes in an existing directory server
- If you have user names that differ only in case, you must map the `uid` attribute, which ignores case, to an attribute that does not ignore case

The format is `service:attribute-name=mapped-attribute-name`.

If you want to map more than one attribute for a given service, you can define multiple `attributeMap` attributes.

In the following example, the `employeeName` and `home` attributes would be used whenever the `uid` and `homeDirectory` attributes would be used for the `passwd` service.

```
attributeMap: passwd:uid=employeeName  
attributeMap: passwd:homeDirectory=home
```

There exists one special case where you can map the `passwd` service's `gecos` attribute to several attributes. The following is an example.

```
attributemap: gecos=cn sn title
```

This maps the `gecos` values to a space-separated list of the `cn`, `sn`, and `title` attribute values.

objectClass Map

The Solaris LDAP naming service allows object classes to be remapped for any of its services. If you want to map more than one object class for a given service, you can define multiple `objectclassMap` attributes. In the following example, the `myUnixAccount` object class is used whenever the `posixAccount` object class is used.

```
objectclassMap: passwd:posixAccount=myUnixAccount
```

Client Profiles

To simplify Solaris client setup, and avoid having to reenter the same information for each and every client, create a single client profile on the directory server. This way, a single profile defines the configuration for all clients configured to use it. Any subsequent change to the profile attributes is propagated to the clients at a rate defined by the refresh interval.

These client profiles should be stored in a well-known location on the LDAP server. The root DN for the given domain must have an object class of `nisDomainObject` and a `nisDomain` attribute containing the client's domain. All profiles are located in the `ou=profile` container relative to this container. These profiles should be readable anonymously.

Client Profile Attributes

The following lists the Solaris LDAP client's profile attributes, which can be set automatically when you run `idsconfig(1M)`. See "Initializing a Client Manually" on page 247 for information on how to set a client profile manually.

TABLE 13-2 Client Profile Attributes

Attribute	Description
<code>cn</code>	The profile name. No default value, must be specified.

TABLE 13-2 Client Profile Attributes (Continued)

Attribute	Description
preferredServerList	The host addresses of the preferred servers is a space separated list of server addresses. (Do not use host names.) The servers in this list are tried in order BEFORE those in the defaultServerList until a successful connection is made. This has no default value. At least one server must be specified in either preferredServerList or defaultServerList.
defaultServerList	The host addresses of the default servers is a space separated list of server addresses. (Do not use host names.) After the servers in the preferredServerList are tried, those default servers on the client's subnet are tried, followed by the remaining default servers, until a connection is made. At least one server must be specified in either the preferredServerList or defaultServerList. The servers in this list are tried only after those on the preferred server list. This attribute has no default value.
defaultSearchBase	The DN relative to which to locate the well-known containers. There is no default for this value. However, this can be overridden for a given service by the serviceSearchDescriptor attribute.
defaultSearchScope	Defines the scope of a database search by a client. It can be overridden by the serviceSearchDescriptor attribute. The possible values are one or sub. The default value is a one level search.
authenticationMethod	Identifies the method of authentication used by the client. The default is none (anonymous). See "Choosing Authentication Methods" on page 217 for more information.
credentialLevel	Identifies the type of credentials a client should use to authenticate. The choices are anonymous or proxy. The default is anonymous.
serviceSearchDescriptor	Defines how and where a client should search for a naming database, for example, if the client should look in one or more points in the DIT. By default no SSDs are defined.

TABLE 13-2 Client Profile Attributes (Continued)

Attribute	Description
<code>serviceAuthenticationMethod</code>	Authentication method used by a client for the specified service. By default, no service Authentication Methods are defined. If a service does not have <code>serviceAuthenticationMethod</code> defined, it will default to the value of <code>authenticationMethod</code> .
<code>attributeMap</code>	Attribute mappings used by client. By default no <code>attributeMap</code> is defined.
<code>objectclassMap</code>	Object class mappings used by client. By default no <code>objectclassMap</code> is defined.
<code>searchTimeLimit</code>	Maximum time [in seconds] a client should allow for a search to complete before timing out. This does not affect the time the LDAP server will allow for a search to complete. Default value is 30 seconds.
<code>bindTimeLimit</code>	Maximum time in seconds a client should allow to bind with a server before timing out. Default value is 30 seconds.
<code>followReferrals</code>	Specifies whether a client should follow an LDAP referral. Possible values TRUE or FALSE. The default value is TRUE.
<code>profileTTL</code>	Time between refreshes of the client profile from the LDAP server by the <code>ldap_cachemgr(1M)</code> . Default is 43200 seconds or 12 hours. If given a value of 0, the profile will never be refreshed.

Local Client Attributes

The following table lists the client attributes that can be set locally using `ldapclient(1M)`.

TABLE 13-3 Local Client Attributes

Attribute	Description
<code>domainName</code>	Specifies the client's domain name (which becomes the default domain for the client machine). This has no default value and must be specified.

TABLE 13-3 Local Client Attributes (Continued)

Attribute	Description
proxyDN	The proxy's distinguished name. If the client machine is configured with <code>credentialLevel</code> of <code>proxy</code> , the <code>proxyDN</code> must be specified.
proxyPassword	The proxy's password. If the client machine is configured with <code>credentialLevel</code> of <code>proxy</code> , the <code>proxyPassword</code> must be defined.
certificatePath	The directory on the local file system containing the certificate databases. If a client machine is configured with <code>authenticationMethod</code> or <code>serviceAuthenticationMethod</code> using TLS, then this attribute is used. The default value is <code>/var/ldap</code> .

Note – If the `BaseDN` in an SSD contains a trailing comma, it is treated as a relative value of the `defaultSearchBase`. The values of the `defaultSearchBase` are appended to the `BaseDN` before a search is performed.

ldap_cachemgr Daemon

`ldap_cachemgr(1M)` is a daemon that runs on LDAP client machines. It performs the following key functions.

- It gains access to the configuration files, running as root
- It refreshes the client configuration information stored in the profiles on the server and pulls this data from the clients
- It maintains a sorted list of active LDAP servers to use
- It improves lookup efficiency by caching some common lookup requests submitted by various clients
- It improves the efficiency of host lookups

Note – The `ldap_cachemgr` must be running at all times for LDAP naming services to work.

Refer to `ldap_cachemgr(1M)` for detailed information.

LDAP Naming Services Security Model

Introduction

Solaris LDAP naming services use the LDAP repository as a source of both a naming service and as an authentication service. This section discusses the concepts of client identity, authentication methods, `pam_ldap(5)` and `pam_unix(5)` modules, and password management.

To access the information stored in the LDAP repository, clients can first establish identity with the directory server. This identity can be either anonymous or as an object recognized by the LDAP server. Based on the client's identity and the server's Access Control Information (ACI), the LDAP server will allow the client to read or write directory information. For more information on ACIs, consult the *Sun ONE Directory Server 5.1 Administrator's Guide*.

If the client is connecting as anything other than anonymous for any given request, the client must prove its identity to the server using an authentication method supported by both the client and server. Once the client has established its identity, it can then make the various LDAP requests.

Keep in mind that there is a distinction between how the naming service and the authentication service (`pam_ldap`) authenticate to the directory. The naming service will read various entries and their attributes from the directory based on predefined identity. The authentication service (`pam_ldap(5)`) establishes whether the user has entered the correct password by using that user's name and password to authenticate to the LDAP server.

Transport Layer Security (TLS)

TLS can be used to secure communication between an LDAP client and the directory server, providing both privacy and data integrity. The TLS protocol is a super set of the Secure Sockets Layer (SSL) protocol. Solaris LDAP naming services support TLS connections. Be aware that using SSL will add load to the directory server and the client.

You will need to set up your directory server for SSL. See the *Sun ONE Directory Server 5.1 Administrator's Guide* for more information on setting up Sun ONE Directory Server 5.1 for SSL. You will also need to set up your LDAP client for SSL.

Note – In order to use TLS for Solaris LDAP naming services, the directory server must use the default ports, 389 and 636, for LDAP and SSL, respectively. If your directory server does not use these ports, you cannot use TLS at this time.

See “Setting Up TLS Security” on page 249 for more information.

Assigning Client Credential Levels

LDAP naming services clients authenticate to the LDAP server according to a credential level. LDAP clients can be assigned three possible credential levels with which to authenticate to a directory server.

- anonymous
- proxy
- proxy anonymous

Anonymous

If you use `anonymous` access, you only have access to data that is available to everyone. Also, you should consider the security implications. Allowing anonymous access for certain parts of the directory implies that anyone with access to the directory will be able to perform those operations. If you are using an `anonymous` credential level, you will need to allow read access to all the LDAP naming entries and attributes.



Caution – Allowing `anonymous write` to a directory should never be done, as anyone could change information in the DIT to which they have write access, including another user’s password, or their own identity.

Note – Sun ONE Directory Server 5.1 allows you to restrict access based on IP addresses, DNS name, authentication method, and time-of-day. You might want to limit access with further restrictions. See “Managing Access Control” in the *Sun ONE Directory Server 5.1 Administrator’s Guide* for more information.

Proxy

The client authenticates or binds to the directory using a proxy account. This proxy account can be any entry that is allowed to bind to the directory. This proxy account needs sufficient access to perform the naming service functions on the LDAP server. You will need to configure the `proxyDN` and `proxyPassword` on every client using

the proxy credential level. The encrypted proxyPassword will be stored locally on the client. You can set up different proxies for different groups of clients. For example, you can configure a proxy for all the sales clients to access both the company-wide-accessible and sales directories, while preventing sales clients from accessing human resource directories with payroll information. Or, in the most extreme cases, you can either assign different proxies to each client or assign just one proxy to all clients. A typical LDAP deployment would probably lie between the two extremes. Consider the choices carefully. Too few proxy agents might limit your ability to control user access to resources. However, having too many proxies complicates the setup and maintenance of the system. You need to grant the appropriate rights to the proxy user. This will vary depending on your environment. See “Credential Storage” on page 216 for information on how to determine which authentication method makes the most sense for your configuration.

If the password changes for a proxy user, you need to update it on every client that uses that proxy user. If you use password aging on LDAP accounts, be sure to turn it off for proxy users.

Note – Be aware that the proxy credential level applies to all users and processes on any given machine. If two users need to use different naming policies, they must use different machines.

In addition, if clients are using a proxy credential to authenticate, the proxyDN must have the same proxyPassword on all of the servers.

proxy anonymous

proxy anonymous is a multi-valued entry, in that more than one credential level is defined. A client assigned the proxy anonymous level will first attempt to authenticate with its proxy identity. If the client is unable to authenticate as the proxy user for whatever reason (user lockout, password expired, for example), then the client will use anonymous access. This might lead to a different level of service, depending on how the directory is configured.

Credential Storage

If you configure a client to use a proxy identity, the client saves its proxyDN and proxyPassword in /var/ldap/ldap_client_cred. For the sake of increased security, this file is restricted to root access only, and the value of proxyPassword is encrypted. While past LDAP implementations have stored proxy credentials in a client’s profile, Solaris 9 LDAP naming services do not. Any proxy credentials set using ldapclient during initialization are stored locally. This results in improved security surrounding a proxy’s DN and password information. See Chapter 16 for more information on setting up client profiles.

Choosing Authentication Methods

When you assign the `proxy` or `proxy-anonymous` credential level to a client, you also need to select a method by which the proxy authenticates to the directory server. By default, the authentication method is `none`, which implies anonymous access. The authentication method may also have a transport security option associated with it.

The authentication method, like the credential level, may be multi-valued. For example, in the client profile you could specify that the client first tries to bind using the `simple` method secured by TLS. If unsuccessful, the client would try to bind with the `sasl/digest-MD5` method. The `authenticationMethod` would then be `tls:simple;sasl/digest-MD5`.

LDAP naming services support some Simple Authentication and Security Layer (SASL) mechanisms. These mechanisms allow for a secure password exchange without requiring TLS. However, these mechanisms do not provide data integrity or privacy. See RFC 2222 for information on SASL.

The following authentication mechanisms are supported.

- `none`

The client does not authenticate to the directory. This is equivalent to the `anonymous` credential level.

- `simple`

If the client machine uses the `simple` authentication method, it binds to the server by sending the user's password in the clear. The password is thus subject to snooping. The primary advantages of using the `simple` authentication method are that all directory servers support it and that it is easy to set up.

- `sasl/digest-MD5`

The client's password is protected during authentication, but the session is not encrypted. Some directory servers, including Sun ONE Directory Server 5.1, also support the `sasl/digest-MD5` authentication method. The primary advantage of `digest-MD5` is that the password does not go over the wire in the clear during authentication and therefore is more secure than the `simple` authentication method. See RFC 2831 for information on `digest-MD5`. `digest-MD5` is considered an improvement over `cram-MD5` for its improved security.

When using `sasl/digest-MD5`, the authentication is secure, but the session is not protected.

- `sasl/cram-MD5`

In this case, the LDAP session is not encrypted, but the client's password is protected during authentication, as authentication is performed using `sasl/cram-MD5`.

See RFC 2195 for information on the `cram-MD5` authentication method, which is supported by some, but not all directory servers. For instance, Sun ONE Directory Server 5.1 does not support `cram-MD5`.

- `tls:simple`
The client binds using the `simple` method and the session is encrypted. The password is protected.
- `tls: sasl/cram-MD5`
The LDAP session is encrypted and the client authenticates to the directory server using `sasl/cram-MD5`.
- `tls:sasl/digest-MD5`
The LDAP session is encrypted and the client authenticates to the directory server using `sasl/digest-MD5`.



Caution – Sun ONE Directory Server 5.1 requires passwords to be stored in the clear in order to use `digest-MD5`. If the authentication method is set to `sasl/digest-MD5` or `tls:sasl/digest-MD5`, then the passwords for the proxy user will need to be stored in the clear. Be careful that the `userPassword` attribute has the proper ACIs if it is stored in the clear, so that it is not readable.

Authentication and Services

The authentication method can be specified for a given service in the `serviceAuthenticationMethod` attribute. The following services currently support this.

- `passwd-cmd`
This service is used by `bypasswd(1)` to change the login password and password attributes.
- `keyserv`
This service is used by `theckey(1)` and `newkey(1M)` utilities to create and change a user's Diffie-Hellman key pair.
- `pam_ldap`
This service is used for authenticating users with `pam_ldap(5)`.

Note – If the service does not have a `serviceAuthenticationMethod` set, it will default to the value of the `authenticationMethod` attribute.

The following example shows a section of a client profile in which the users will use `sasl/digest-MD5` to authenticate to the directory server, but will use an SSL session to change their password.

```
serviceAuthenticationMethod=pam_ldap:sasl/digest-MD5
serviceAuthenticationMethod=passwd-cmd:tls:simple
```

Pluggable Authentication Methods

By using the PAM framework, you can choose among several authentication services. You can use either `pam_unix(5)` or `pam_ldap(5)` in conjunction with LDAP.

Because of its increased flexibility and support of stronger authentication methods, the use of `pam_ldap` is recommended.

`pam_unix(5)`

If you have not changed the `pam.conf(4)` file, `pam_unix(5)` is enabled by default. `pam_unix(5)` follows the traditional model of UNIX authentication, which means that

1. The client retrieves the user's encrypted password from the name service.
2. The user is prompted for his password.
3. The user's password is encrypted.
4. The client compares the two encrypted passwords to determine if the user should be authenticated or not.

Additionally, there are two restrictions when using `pam_unix(5)`.

- The password must be stored in UNIX `crypt` format and not in any other encryption methods, including clear.
- The `userPassword` attribute must be readable by the name service.
For example, if you set the credential level to `anonymous`, then anyone must be able to read the `userPassword` attribute. Similarly, if you set the credential level to `proxy`, then the proxy user must be able to read the `userPassword` attribute.

Note – `pam_unix(5)` is not compatible with `sasl` authentication method `digest-MD5`, since Sun ONE Directory Server 5.1 requires passwords to be stored in the clear in order to use `digest-MD5`, but `pam_unix` requires the password be stored in `crypt` format.

`pam_ldap(5)`

When using `pam_ldap(5)`, the user binds to the LDAP server. The authentication method is defined in `pam_ldap`'s `serviceAuthenticationMethod` parameter if one exists. Otherwise, the `authenticationMethod` is used by default.

If `pam_ldap(5)` is able to bind to the server with the user's identity and supplied password, it authenticates the user.

`pam_ldap(5)` does not read the `userPassword` attribute. Therefore, there is no need to grant access to read the `userPassword` attribute unless there are other clients using `pam_unix(5)`. `pam_ldap(5)` does not support the `none` authentication method. Thus, you must define the `serviceAuthenticationMethod` or the `authenticationMethod` attributes in order for clients to use `pam_ldap(5)`.



Caution – If the `simple` authentication method is used, the `userPassword` attribute can be read on the wire by third parties.

See “Example `pam.conf` File for `pam_ldap`” on page 264.

PAM and Changing Passwords

Use `passwd(1)` to change a password. In order to change the password, the `userPassword` attribute must be writeable by the user. Remember that the `serviceAuthenticationMethod` for `passwd-cmd` will override the `authenticationMethod` for this operation. Depending on the authentication used, the current password might be un-encrypted on the wire.

In the case of `pam_unix(5)` the new `userPassword` attribute is encrypted using UNIX `crypt` and tagged before being written to LDAP. Therefore, the new password is encrypted on the wire, regardless of the authentication method used to bind to the server.

For `pam_ldap`, when a password is changed, the new password is un-encrypted. Therefore, to insure privacy, you need to use TLS. If TLS is not used, the new `userPassword` will be subject to snooping.

When setting the password with `pam_ldap(5)` with Sun ONE Directory Server 5.1, the password is encrypted using the `serverStorageScheme` (as it is untagged). See “User Account Management” in the *Sun ONE Directory Server 5.1 Administrator’s Guide* for additional information about the `passwordStorageScheme` attribute.

Note – You need to consider the following when setting the `passwordStorageScheme` attribute. If a NIS, NIS+, or another client using `pam_unix` is using LDAP as a repository, then `passwordStorageScheme` needs to be `crypt`. Also, if using `pam_ldap` with `sasl/digest-MD5` with Sun ONE Directory Server 5.1, `passwordStorageScheme` must be set to `clear`.

Password Management

LDAP naming services take advantage of the password and account lockout policy support in Sun ONE Directory Server 5.1. You can configure `pam_ldap(5)` to support user account management. `passwd(1)` enforces password syntax rules set by the Sun ONE Directory Server password policy, when used with the proper PAM configuration.

The following password management features are supported through `pam_ldap(5)`. These features depend on Sun ONE Directory Server 5.1's password and account lockout policy configuration. You can enable as many or as few of the features as you want.

- Password aging and expiration notification
Users must change their passwords according to a schedule. A password expires if it is not changed within the time configured. An expired password causes user authentication to fail.
Users see a warning message whenever they log in within the expiration warning period. The message specifies the number of hours or days until the password expires.
- Password syntax checking
New passwords must meet the minimum password length requirements. In addition, a password cannot match the value of the `uid`, `cn`, `sn`, or `mail` attributes in the user's directory entry.
- Password in history checking
Users cannot reuse passwords. If a user attempts to change the password to one that was previously used, `passwd(1)` fails. LDAP administrators can configure the number of passwords kept in the server's history list.
- User account lockout
A user account can be locked out after a given number of repeated authentication failures. A user can also be locked out if his account is inactivated by an administrator. Authentication will continue to fail until the account lockout time is passed or the administrator reactivates the account.

Note – The preceding password management features only work with Sun ONE Directory Server 5.1 bundled with Solaris 9. For information about configuring the password and account lockout policy on the server, see the “User Account Management” chapter in the *Sun ONE Directory Server 5.1 Administrator's Guide*. Also see “Example `pam_conf` file for `pam_ldap` Configured for Password Management” on page 266.

Before configuring the password and account lockout policy on Sun ONE Directory Server 5.1, make sure all hosts use the “newest” LDAP client with `pam_ldap` password management. The “newest” version of the LDAP client is part of Solaris 9, update 2.

In addition, make sure the clients have a properly configured `pam.conf(4)` file. Otherwise, LDAP naming services will not work when proxy or user passwords expire.

Planning for the LDAP Naming Service (Tasks)

This chapter discusses the high-level planning you should do before beginning the server and client setup and installation process.

This chapter covers the following topics.

- “Overview” on page 223
- “Planning the Network Model” on page 224
- “Planning the Directory Information Tree (DIT)” on page 224
- “Replica Servers” on page 226
- “Planning the Security Model” on page 227
- “Planning Client Profiles and Default Attribute Values” on page 227
- “Planning the Data Population” on page 228

Overview

The LDAP client profile is a collection of configuration information an LDAP client uses to access the LDAP naming service information on the supporting LDAP server to provide LDAP naming services. In this chapter, we will use this center piece of the LDAP configuration to discuss the planning of the various aspects of the LDAP naming services. These include the network model, the Directory Information Tree, the security model, the default values of the various profile attributes, and finally, the preparation for data population.

Planning the Network Model

For availability and performance considerations, it would be best if each subnet of the company-wide network has its own LDAP server to service all the LDAP clients in the subnet. Only one of them needs to be a master LDAP server. The rest could all be replicas of the master server.

To plan for the network configuration, consider how many servers are available, how a client would be able to get to the servers, and in what order the servers should be accessed. If there is one per subnet, we could use the `defaultServerList` attribute to list all the servers and have the LDAP client sort and manipulate the access order. If the servers need to be accessed in certain order due to speed or data management reasons, then we should use the `preferredServerList` attribute to define the fixed order of accessing the servers. Note that you might not want to put the master server on either of these lists to reduce the load on the master server.

In addition, you may find three more attributes worth consideration when planning for the server and network configuration. The `bindTimeLimit` attribute can be used to set the time-out value for a TCP connect request, the `searchTimeLimit` attribute can be used to set the time-out value for an LDAP search operation, and the `profileTTL` attribute can be used to control how often the LDAP client should download its profile from the servers. For a slow or unstable network, the `bindTimeLimit` and `searchTimeLimit` attributes might need a larger value than the defaults. For early stage testing of the deployment, you might want to reduce the value of the `profileTTL` attribute to have the clients pick up the frequent changes made to the profile stored in the LDAP servers.

Planning the Directory Information Tree (DIT)

As mentioned in the previous chapter, the LDAP naming services has a default Directory Information Tree (DIT) and the associated default schema. For example, the `ou=people` container contains the user account, password, and shadow information. The `ou=hosts` container contains information about systems in the network. Each entry in the `ou=people` container would be of objectclass `posixAccount` and `shadowAccount`. The default DIT is a well designed directory structure and is based on open standards. It should be sufficient for most of the naming service needs, and is recommended to be used without changes. If you choose to use the default DIT, the only thing you need to decide is from which node (base DN) on in the directory tree the naming service information will be searched for a given domain. This node is

specified with the `defaultSearchBase` attribute. Additionally, you might want to set the `defaultSearchScope` attribute to tell the clients the scope of search a naming service lookup should perform. Is it just searching one level under the DN (one), or the entire subtree under the DN (sub)?

There are times, however, that more flexibility is needed for the LDAP naming service to either work with an existing DIT or handle a more complicated DIT with naming service data scattered around the directory tree. For example, user account entries may exist in different part of the tree. The `serviceSearchDescriptor`, `attributeMap`, and `objectclassMap` attributes in the client profile are designed to handle these situations.

A service search descriptor can be used to override the default search base, search scope, and search filter for a particular service. See “Service Search Descriptors (SSDs) and Schema Mapping” on page 208.

The `AttributeMap` and `ObjectclassMap` attributes provide a way for schema mapping. They make it possible for the LDAP naming services to work with an existing DIT. You can map the `posixAccount` object class to an existing object class, `myAccount`, for example, an attributes in the `posixAccount` object class to attributes in the `myAccount` object class.

Multiple Directory Servers

Multiple LDAP servers can serve one DIT. For example, some subtrees of the DIT reside on other LDAP servers. In this case, an LDAP server may refer the LDAP client to a different server for the naming data it knows about but is not in its own database. If you plan such a DIT configuration, you should set the clients’ profile attribute `followReferrals` to indicate to the LDAP naming service to follow server referrals to continue naming service lookups. However, it is best to have all naming data for a given domain reside on a single directory server, if at all possible.

Referrals can be useful if you want to have clients access read-only replicas most of the time and follow referrals to a read/write master server only when necessary. In this way, the master server does not get overloaded with requests that could be handled by replicas.

Data Sharing With Other Applications

To make best use of LDAP, you should have a single LDAP entry for each logical entry. For example, for a user you can have not only company white-page information, but also Solaris account information, and possibly application-specific data. Since `posixAccount` and `shadowAccount` are auxiliary object classes, they can be added to any entry in the directory. This will require careful planning, setup, and administration.

Choosing the Directory Suffix

See Chapter 11 for information about how to choose an appropriate directory suffix.

Replica Servers

There are three different strategies to employ when setting up replica servers.

- Single-master replication
- Floating-master replication
- Multi-master replication

Single-master

With single-master replication, only one master server for any given partition or non-partitioned network holds writable copies of directory entries. Any replica servers have read-only copies of the directory entries. While both replicas and masters can perform searches, compares, and bind operations, only the master server can perform write operations.

The potential disadvantage to the single-master replication strategy is that the master server is a single point of failure. If the master server goes down, none of the replicas can process write operations.

Floating-master

The floating-master strategy is similar to the single-master strategy in that there is only one master server with write capabilities at any given time for a given partitioned or non-partitioned network. However, when implementing the floating-master strategy, when the master server goes down, a replica is automatically transformed into a master server by way of an algorithm.

One potential disadvantage to the floating-master replication strategy is that if your network becomes partitioned and replicas on either side of the partition become masters, the process of reconciling the new masters can be very complicated if the network is rejoined.

Multi-master

With multi-master replication, there are multiple master servers with their own read-write copies of the directory entry data. While the multi-master strategy eliminates the problem of having a single point of failure, update conflicts can occur between servers. In other words, if an entry's attribute is modified around the same time on two masters, an update conflict resolution policy, such as "last writer wins," must be in place.

Refer to the *Sun ONE Directory Server 5.1 Administrator's Guide* for information on how to set up replica servers.

Planning the Security Model

To plan for the security model, you should first consider what identity the LDAP client should be using to talk to the LDAP server. For example, you must decide if you want strong authentication to protect the user password flow across the wire, and/or if it is needed to encrypt the session between the LDAP client and the LDAP server to protect the LDAP data transmitted.

The `credentialLevel` and `authenticationMethod` attributes in the profile are used for this. There are three possible credential levels for `credentialLevel`: `anonymous`, `proxy`, and `proxy anonymous`. See “LDAP Naming Services Security Model” on page 214 for a detailed discussion of LDAP naming service security concepts.

The main decisions you need to make when planning your security model are the following.

- What credential level and authentication methods will LDAP clients use?
- Will you use TLS?
- Do you need to be backward compatible with NIS or NIS+? In other words, will clients use `pam_unix` or `pam_ldap`?
- What will the servers' `passwordStorageScheme` attribute settings be?
- How will you set up the Access Control Information? For more information on ACIs, consult the *Sun ONE Directory Server 5.1 Administrator's Guide*.

Planning Client Profiles and Default Attribute Values

By going through the previous planning steps (network model, DIT, and security model), you should have some idea of the values for the following profile attributes.

- `cn`
- `defaultServerList`
- `preferredServerList`
- `bindTimeLimit`

- `searchTimeLimit`
- `profileTTL`
- `defaultSearchBase`
- `defaultSearchScope`
- `serviceSearchDescriptor`
- `attributeMap`
- `objectclassMap`
- `followReferrals`
- `credentialLevel`
- `authenticationMethod`
- `serviceCredentialLevel`
- `serviceAuthenticationMethod`

Of the preceding attributes, only `cn`, `defaultServerList`, and `defaultSearchBase` are required. They have no default values. The rest are optional, and some have default values.

See Chapter 16 for more information about setting up LDAP clients.

Planning the Data Population

To populate the LDAP server with the LDAP naming service data, after the LDAP server has been configured with the proper DIT and schema, it is best to use the new `ldapaddent` tool. This tool will create entries in LDAP containers from their corresponding `/etc` files. It can be used to populate data into the containers for the following type of data: `aliases`, `auto_*`, `bootparams`, `ethers`, `group`, `hosts` (including IPv6 addresses), `netgroup`, `netmasks`, `networks`, `passwd`, `shadow`, `protocols`, `publickey`, `rpc`, and `services`.

By default, `ldapaddent` reads from the standard input and adds this data to the LDAP container associated with the database specified on the command line. But an input file from which data should be read can be specified using the `-f` option.

The entries are stored in the directory based on the client's configuration, thus the client must be configured to use the LDAP naming service.

For better performance, the recommended order in which the databases should be loaded is as follows.

1. `passwd` database followed by `shadow` database
2. `networks` database followed by `netmasks` database
3. `bootparams` database followed by `ethers` database

Note that when adding automounter entries, the database name is in the form of `auto_*` (for example, `auto_home`).

If you have `/etc` files from different hosts to add to the LDAP server, you can either merge all of them into the same `/etc` file and then use `ldapaddent` on one host to add the files, or perform `ldapaddent` on the different hosts one by one, with the expectation that each host is already configured as a LDAP client.

If your naming service data is already in an NIS server, and you want to move the data to the LDAP server for LDAP naming services, use the `ypcat` (or `niscat`) command to dump the NIS map into files. Then, run `ldapaddent` against these files to add the data to the LDAP server.

Note – `ldapaddent` can only be run on an LDAP client.

The following procedure assumes that the tables are to be extracted from a `yp` client.

▼ How to Populate a Server With `host` Entries Using `ldapaddent`

1. Make sure that Sun ONE Directory Server 5.1 was set up using `idsconfig`.
2. Become superuser on a client machine.
3. Make the machine an LDAP client.

```
# ldapclient init -a profileName=new -a  
\domainName=west.example.com 192.168.0.0
```

4. Populate the server with data.

```
# ldapaddent -h 192.168.0.0 -D directory_manager -f /etc/hosts  
hosts
```

In this example, the `directory_manager` password would be sent in the clear, as currently, `ldapaddent` binds using the `simple` authentication method.

You can also populate your directory server with NIS+ data using the proper settings in `rpc.nisd`. See Chapter 19.

Setting Up Sun ONE Directory Server 5.1 (Tasks)

This chapter describes how to configure Sun™ ONE Directory Server 5.1 (formerly iPlanet™ Directory Server 5.1) to support a network of Solaris LDAP naming services clients. The information is specific to the Sun ONE Directory Server.

Note – You must have already performed all the procedures described in Chapter 11 before you can configure Sun ONE Directory Server 5.1 to work with Solaris LDAP clients.

Note – A directory server (an LDAP server) *cannot* be its own client.

This chapter covers the following topics.

- “Configuring Sun ONE Directory Server 5.1 Using `idsconfig`” on page 232
- “Using Service Search Descriptors to Modify Client Access to Various Services” on page 235
- “Running `idsconfig`” on page 236
- “Populating the Directory Server Using `ldapaddent`” on page 240
- “Managing Printer Entries” on page 241
- “Populating the Directory Server With Additional Profiles” on page 242
- “Configuring the Directory Server to Enable Password Management ” on page 242

Configuring Sun ONE Directory Server

5.1 Using `idsconfig`

Creating a Checklist Based on Your Server Installation

During the server installation process, you will have defined crucial variables, with which you should create a checklist similar to the one below before launching `idsconfig`. You can use the blank checklist provided in “Blank Checklists” on page 261.

Note – The information included below will serve as the basis for all examples that follow in the LDAP related chapters. The example domain is of an widget company, Example, Inc. with stores nationwide. The examples will deal with the West Coast Division, with the domain `west.example.com`

TABLE 15-1 Server Variables Defined

Variable	Definition for Example Network
Port number at which an instance of the directory server is installed (DEFAULT=389)	<code>default</code>
Name of server	<code>ipdserver</code> (from the FQDN <code>ipdserver.west.example.com</code> or <code>192.168.0.0</code>)
Replica server(s) (IPnumber:port number)	<code>192.168.0.1</code> [for <code>ipdrep.west.example.com</code>]
Directory manager [<code>dn: cn=directory manager</code>]	<code>default</code>
Domain name to be served	<code>west.example.com</code>
Maximum time (in seconds) to process client requests before timing out	<code>-1</code>
Maximum number of entries returned for each search request	<code>-1</code>

Note – If you are using hostnames in defining `defaultServerList` or `preferredServerList`, you **MUST** ensure LDAP is not used for hosts lookup. This means `ldap` must not be in `/etc/nsswitch.conf` `hosts` line.

TABLE 15-2 Client Profile Variables Defined

Variable	Definition for Example Network
Profile name	WestUserProfile
Server list (defaults to the local subnet)	west.example.com or 192.168.0.0
Preferred server list (listed in order of which server to try first, second, and so on)	none
Search scope (number of levels down through the directory tree. 'One', the default, or 'Sub')	one (default)
Credential used to gain access to server. Default is anonymous	proxy
Follow Referrals? (a pointer to another server if the main server is unavailable) Default is no.	Y
Search time limit (default is 30 seconds) for waiting for server to return information.	default
Bind time limit (default is 10 seconds) for contacting server. The default is seconds.	2
Authentication method Default is none.	simple

Note – Client profiles are defined per domain. At least one profile must be defined for a given domain.

Attribute Indexes

`idsconfig` indexes the following list of attributes for improved performance.

<code>member nisnetgroup</code>	<code>pres, eq, sub</code>
<code>nisnetgrouptriple</code>	<code>pres, eq, sub</code>
<code>memberuid</code>	<code>pres, eq</code>

uidNumber	pres,eq
gidNumber	pres,eq
ipHostNumber	pres,eq
ipNetworkNumber	pres,eq
ipProtocolNumber	pres,eq
oncRpcNumber	pres,eq

Schema Definitions

idsconfig(1M) automatically adds the necessary schema definitions. Unless you are very experienced in LDAP administration, do not manually modify the server schema. See Chapter 18 for an extended list of schemas used by the LDAP naming service.

Using Browsing Indices

The browsing index functionality of the Sun ONE Directory Server, otherwise known as the virtual list view, provides a way in which a client can view a select group or number of entries from very long list, thus making the search process less time consuming for each client. Browsing Indexes provide optimized, predefined search parameters with which the Solaris LDAP naming client can access specific information from the various services more quickly. Keep in mind that if you do not create browsing indexes, the clients may not get all the entries of a given type because the server limits for search time or number of entries might not be enforced.

Indexes are configured on the directory server and the proxy user has read access to these indexes.

Refer to the Sun ONE Directory Server Administrators Guide 5.1 for information on configuring indexes on the Sun ONE Directory Server as well as the performance cost associated with using them.

In the following example, note that the `-n` option denotes the name of the database with the entries to be indexed and the `-s` option denotes the instance of the directory server.

Note – `idsconfig` creates all the default VLV indices.

```
directoryserver -s ipdserver vlindex -n userRoot -T getgrent
directoryserver -s ipdserver vlindex -n userRoot -T gethostent
directoryserver -s ipdserver vlindex -n userRoot -T getnetent
directoryserver -s ipdserver vlindex -n userRoot -T getpwent
directoryserver -s ipdserver vlindex -n userRoot -T getrpcent
directoryserver -s ipdserver vlindex -n userRoot -T getspent
```

Using Service Search Descriptors to Modify Client Access to Various Services

A service search descriptor (SSD) changes the default search request for a given operation in LDAP to a search you define. SSDs are particularly useful if, for example, you have been using LDAP with customized container definitions or another operating system and are now transitioning to Solaris 9. Using SSDs, you can configure Solaris 9 LDAP naming services without having to change your existing LDAP database and data.

Setting Up SSDs Using `idsconfig`

Assume your predecessor at Example, Inc. had configured LDAP, storing users in `ou=Users` container. You are now upgrading to Solaris 9. By definition, Solaris 9 LDAP assumes that user entries are stored in `ou=People` container. Thus, when it comes to searching the `passwd` service, LDAP will search the `ou=people` level of the DIT and not find the correct values.

One rather laborious solution to the above problem would be to completely overwrite Example, Inc.'s existing DIT and to rewrite all the exiting applications on Example, Inc.'s network so that they are compatible with the new LDAP naming service. A second, far preferable solution would be to use an SSD that would tell LDAP to look for user info in an `ou=Users` container instead the default `ou=people` container.

You would define the necessary SSD during the configuration of the Sun ONE Directory Server 5.1 using `idsconfig`. The prompt line appears as follows.

```
Do you wish to setup Service Search Descriptors (y/n/h)? y
A Add a Service Search Descriptor
D Delete a SSD
M Modify a SSD
P Display all SSD's
```

```
H Help
X Clear all SSD's

Q Exit menu
Enter menu choice: [Quit] a
Enter the service id: passwd
Enter the base: service ou=user,dc=west,dc=example,dc=com
Enter the scope: one[default]
A Add a Service Search Descriptor
D Delete a SSD
M Modify a SSD
P Display all SSD's
H Help
X Clear all SSD's

Q Exit menu
Enter menu choice: [Quit] p

Current Service Search Descriptors:
=====
Passwd:ou=Users,ou=west,ou=example,ou=com?

Hit return to continue.

A Add a Service Search Descriptor
D Delete a SSD
M Modify a SSD
P Display all SSD's
H Help
X Clear all SSD's

Q Exit menu
Enter menu choice: [Quit] q
```

Running `idsconfig`

Note – You do not need special rights to run `idsconfig`, nor do you need to be an LDAP naming client. Remember to create a checklist as mentioned in “Creating a Checklist Based on Your Server Installation” on page 232 in preparation for running `idsconfig`. You do not have to run `idsconfig` from a server or an LDAP naming service client machine. You can run `idsconfig` from any Solaris machine on the network.



Caution – `idsconfig` sends the Directory Manager’s password in the clear. If you do not want this to happen, you must run `idsconfig` on the directory server itself, not on a client.

▼ How to Configure Sun ONE Directory Server Using `idsconfig`

1. Make sure the target Sun ONE Directory Server 5.1 is up and running.
2. Run `idsconfig`.
`# /usr/lib/ldap/idsconfig`
3. Answer the questions prompted. Note that ‘no’ [n] is the default user input. If you need clarification on any given question, type

h
and a brief help paragraph will appear.

Refer to the following example run of `idsconfig` using the definitions listed in the server and client checklists at the beginning of this chapter in “Creating a Checklist Based on Your Server Installation” on page 232. It is an example of a simple setup, without modifying many of the defaults. The most complicated method of modifying client profiles is by creating SSDs. Refer to “Using Service Search Descriptors to Modify Client Access to Various Services” on page 235 for a detailed discussion.

A carriage return sign after the prompt means that you are accepting the [default] by hitting enter.

EXAMPLE 15-1 Running `idsconfig` for the Example, Inc. Network

```
(sysadmin@test) [3:10pm] ns_ldap [31] % sh idsconfig.sh
```

```
It is strongly recommended that you BACKUP the directory server  
before running idsconfig.sh.
```

```
Hit Ctrl-C at any time before the final confirmation to exit.
```

```
Do you wish to continue with server setup (y/n/h)? [n] Y
```

```
Enter the Sun ONE Directory Server’s (Sun ONE Directory Server)  
hostname to setup: IPDSERVER
```

```
Enter the port number for Sun ONE Directory Server (h=help): [389]
```

```
Enter the directory manager DN: [cn=Directory Manager]
```

```
Enter passwd for cn=Directory Manager :
```

```
Enter the domainname to be served (h=help): [west.example.com]
```

```
Enter LDAP Base DN (h=help): [dc=west,dc=example,dc=com]
```

```
Enter the profile name (h=help): [default]
```

EXAMPLE 15-1 Running `idsconfig` for the Example, Inc. Network (Continued)

```
Default server list (h=help): [192.168.0.0]
Preferred server list (h=help):
Choose desired search scope (one, sub, h=help): [one]
The following are the supported credential levels:
  1 anonymous
  2 proxy
  3 proxy anonymous
Choose Credential level [h=help]: [1] 2

The following are the supported Authentication Methods:
  1 none
  2 simple
  3 sasl/DIGEST-MD5
  4 tls:simple
  5 tls:sasl/DIGEST-MD5
Choose Authentication Method (h=help): [1] 2

Current authenticationMethod: simple

Do you want to add another Authentication Method? N

Do you want the clients to follow referrals (y/n/h)? [n] Y

Do you want to modify the server timelimit value (y/n/h)? [n] Y

Enter the time limit for Sun ONE Directory Server (current=3600): [-1]

Do you want to modify the server sizelimit value (y/n/h)? [n] Y

Enter the size limit for Sun ONE Directory Server (current=2000): [-1]

Do you want to store passwords in "crypt" format (y/n/h)? [n] Y

Do you want to setup a Service Authentication Methods (y/n/h)? [n]
Client search time limit in seconds (h=help): [30]
Profile Time To Live in seconds (h=help): [43200]

Bind time limit in seconds (h=help): [10] 2

Do you wish to setup Service Search Descriptors (y/n/h)? [n]
```

Summary of Configuration

```
1 Domain to serve           : west.example.com
2 Base DN to setup          : dc=west,dc=example,dc=com
3 Profile name to create    : default
4 Default Server List       : 192.168.0.0
5 Preferred Server List     :
6 Default Search Scope      : one
7 Credential Level          : proxy
8 Authentication Method     : simple
9 Enable Follow Referrals   : TRUE
10 Sun ONE Directory Server Time Limit      : -1
11 Sun ONE Directory Server Size Limit     : -1
12 Enable crypt password storage : TRUE
```

EXAMPLE 15-1 Running `idsconfig` for the Example, Inc. Network (Continued)

```
13 Service Auth Method pam_ldap :
14 Service Auth Method keysserv :
15 Service Auth Method passwd-cmd:
16 Search Time Limit             : 30
17 Profile Time to Live          : 43200
18 Bind Limit                    : 2
19 Service Search Descriptors Menu
```

```
Enter config value to change: (1-19 0=commit changes) [0]
Enter DN for proxy agent: [cn=proxyagent,ou=profile,dc=west,dc=example,dc=com]
Enter passwd for proxyagent:
Re-enter passwd:
```

WARNING: About to start committing changes. (y=continue, n=EXIT) Y

1. Changed `timelimit` to -1 in `cn=config`.
2. Changed `sizelimit` to -1 in `cn=config`.
3. Changed `passwordstoragescheme` to "crypt" in `cn=config`.
4. Schema attributes have been updated.
5. Schema objectclass definitions have been added.
6. Created DN component `dc=west`.
7. `NisDomainObject` added to `dc=west,dc=example,dc=com`.
8. Top level "ou" containers complete.
9. Nis maps: `auto_home auto_direct auto_master auto_shared` processed.
10. ACI for `dc=west,dc=example,dc=com` modified to disable self modify.
11. Add of VLV Access Control Information (ACI).
12. Proxy Agent `cn=proxyagent,ou=profile,dc=west,dc=example,dc=com` added.
13. Give `cn=proxyagent,ou=profile,dc=west,dc=example,dc=com` read permission for password.
14. Generated client profile and loaded on server.
15. Processing `eq,pres` indexes:
 - `ipHostNumber (eq,pres)` Finished indexing.
 - `uidNumber (eq,pres)` Finished indexing.
 - `ipNetworkNumber (eq,pres)` Finished indexing.
 - `gidnumber (eq,pres)` Finished indexing.
 - `oncrpcnumber (eq,pres)` Finished indexing.
16. Processing `eq,pres,sub` indexes:
 - `membersnisnetgroup (eq,pres,sub)` Finished indexing.
 - `nisnetgrouptriple (eq,pres,sub)` Finished indexing.
17. Processing VLV indexes:
 - `getgrent vlv_index` Entry created
 - `gethostent vlv_index` Entry created
 - `getnetent vlv_index` Entry created
 - `getpwent vlv_index` Entry created
 - `getrpcent vlv_index` Entry created
 - `getspent vlv_index` Entry created

`idsconfig.sh`: Setup of Sun ONE Directory Server server `ipdserver` is complete.

Note: `idsconfig` has created entries for VLV indexes. Use the `directoryserver(1m)` script on `ipdserver` to stop

EXAMPLE 15-1 Running `idsconfig` for the Example, Inc. Network (Continued)

the server and then enter the following `vlvindex` sub-commands to create the actual VLV indexes:

```
directoryserver -s ipdserver vlvindex -n userRoot -T getgrent
directoryserver -s ipdserver vlvindex -n userRoot -T gethostent
directoryserver -s ipdserver vlvindex -n userRoot -T getnetent
directoryserver -s ipdserver vlvindex -n userRoot -T getpwent
directoryserver -s ipdserver vlvindex -n userRoot -T getrpcent
directoryserver -s ipdserver vlvindex -n userRoot -T getspent
```

Note – Any parameters left blank in the summary screen will not be set up.

After `idsconfig` has completed the setup of the directory, you need to run the specified commands on the server before the server setup is complete and the server is ready to serve clients.

Populating the Directory Server Using `ldapaddent`

Note – Before populating the directory server with data, you must configure the server to store passwords in Unix Crypt format if you are using `pam_unix`. If you are using `pam_ldap`, you can store passwords in any format. For more information about setting the password in UNIX crypt format, see the Sun ONE Directory Server documents.

Note – `ldapaddent(1M)` can only run on a client which is already configured for the LDAP naming service.

`ldapaddent` reads from the standard input (that being an `/etc/filename` like `passwd`) and places this data to the container associated with the service. Client configuration determines how the data will be written by default.

▼ How to Populate Sun ONE Directory Server 5.1 With User Password Data Using `ldapaddent`

- Use the `ldapaddent` command to add `/etc/passwd` entries to the server.

```
# ldapaddent -D "cn=directory manager" -f /etc/passwd passwd
```

See `ldapaddent(1M)`. See Chapter 13 for information about LDAP security and write-access to the directory server.

Managing Printer Entries

Adding Printers

To add printer entries to the LDAP directory, use either the `printmgr` configuration tool or the `lpset -n ldap` command-line utility. See `lpset(1M)`. Note that the printer objects added to the directory only define the connection parameter, required by print system clients, of printers. Local print server configuration data is still held in files. A typical printer entry would look like the following:

```
printer-uri=myprinter,ou=printers,dc=mkg,dc=example,dc=com
objectclass=top
objectclass=printerService
objectclass=printerAbstract
objectclass=sunPrinter
printer-name=myprinter
sun-printer-bsdaddr=printsvr.example.com,myprinter,Solaris
sun-printer-kvp=description=HP LaserJet (PS)
printer-uri=myprinter
```

Using `lpget`

`lpget(1M)` can be used to list all printer entries known by the LDAP client's LDAP directory. If the LDAP client's LDAP server is a replica server, then printers listed might not be the same as that in the master LDAP server depending on the update replication agreement. See `lpget(1M)` for more information.

For example, to list all printers for a given base DN, type the following:

```
# lpget -n ldap list

myprinter:
  dn=myprinter,ou=printers,dc=mkt,dc=example,dc=com
  bsdaddr=printsvr.example.com,myprinter,Solaris
  description=HP LaserJet (PS)
```

Populating the Directory Server With Additional Profiles

Use `ldapclient` with the `genprofile` option to create an LDIF representation of a configuration profile, based on the attributes specified. The profile you create can then be loaded into an LDAP server to be used as the client profile. The client profile can be downloaded by the client by using `ldapclient init`.

Refer to `ldapclient(1M)` for information about using `ldapclient genprofile`.

▼ How to Populate the Directory Server With Additional Profiles Using `ldapclient`

1. Become superuser.

2. Use `ldapclient` with the `genprofile` command.

```
# ldapclient genprofile -a profileName=myprofile \  
-a defaultSearchBase=dc=west,dc=example,dc=com \  
-a "defaultServerList=192.168.0.0 192.168.0.1:386" \  
> myprofile.ldif
```

3. Upload the new profile to the server.

```
# ldapadd -h 192.168.0.0 -D "cn=directory manager" -f  
myprofile.ldif
```

Configuring the Directory Server to Enable Password Management

See the “User Account Management” chapter in the *Sun ONE Directory Server 5.1 Administrator’s Guide* for how to use the Directory Server Console or `ldapmodify` to configure the password management policy for the LDAP directory. In order for `pam_ldap` to work properly, the password and account lockout policy must be properly configured on the server.

Passwords for proxy users should *never* be allowed to expire. If proxy passwords expire, clients using the proxy credential level cannot retrieve naming service information from the server. To ensure that proxy users have passwords that do not expire, modify the proxy accounts with the following script.

```
# ldapmodify -h ldapsrvr -D administrator DN \  
-w administrator password <<EOF  
dn: proxy user DN  
DNchangetype: modify  
replace: passwordexpirationtime  
passwordexpirationtime: 20380119031407Z  
EOF
```

Note – pam_ldap password management relies on Sun ONE Directory Server 5.1 to maintain and provide password aging and account expiration information for users. The directory server does not interpret the corresponding data from shadow entries to validate user accounts. pam_unix, however, examines the shadow data to determine if accounts are locked or if passwords are aged. Since the shadow data is not kept up to date by the LDAP naming services or the directory server, pam_unix should not grant access based on the shadow data. The shadow data is retrieved using the proxy identity. Therefore, do not allow proxy users to have read access to the userPassword Attribute. Denying proxy users read access to userPassword prevents pam_unix from making an invalid account validation.

Setting Up Clients (Task)

This chapter describes how to set up a Solaris LDAP naming services client.

This chapter covers the following topics.

- “Prerequisites” on page 245
- “Initializing a Client” on page 246
- “Using Profiles to Initialize a Client” on page 246
- “Using Proxy Credentials” on page 247
- “Initializing a Client Manually” on page 247
- “Modifying a Manual Client Configuration” on page 248
- “Uninitializing a Client” on page 248
- “Setting Up TLS Security” on page 249

Prerequisites

In order for a Solaris client to use LDAP as a naming service the following needs to be in place.

- The client’s domain name must be served by the LDAP server
- The `nsswitch.conf` file needs to point to LDAP for the required services. For information about the `nsswitch.conf` file, see Chapter 2
- The client needs to be configured with all the given parameters that define its behavior
- `ldap_cachemgr` needs to be running on the client
- At least one server for which a client is configured must be up and running

The `ldapclient` utility is the key to setting up an LDAP client, as it performs all of the above steps, except for starting the server. The rest of this chapter will show examples of how to use the `ldapclient` utility to setup a LDAP client and use the various other LDAP utilities to get information about, and check the status of an LDAP client.

Initializing a Client

`ldapclient(1M)` is an utility used to setup LDAP clients in a Solaris operating environment. `ldapclient` assumes the server has already been configured with the appropriate client profiles. You must install and configure the server with the appropriate profiles before you can set up any clients.

There are two ways to set up a client using `ldapclient`.

- *Profile*

At a minimum, you need to specify the server address containing the profile and domain you wish to use. If no profile is specified, then the “default” profile is assumed. The server will provide the rest of the required information, except for proxy and certificate database information. If a client’s credential level is `proxy` or `proxy anonymous`, you must supply the proxy bind DN and password. See “Assigning Client Credential Levels” on page 215 for more information.

- *Manual*

You configure the profile on the client itself, which means defining all parameters from the command line. Thus, the profile information is stored in cache files and is never refreshed by the server.

Note – Though you can manually configure clients, it is not recommended. Using the configuration profiles decreases the complexity and cost of managing clients.

Using Profiles to Initialize a Client

▼ How to Initialize a Client Using Profiles

1. **Become superuser.**
2. **Run `ldapclient` with `init`.**

```
# ldapclient init -a profileName=new -a \
```

```
domainName=west.example.com 192.168.0.0
System successfully configured
```

Using Proxy Credentials

▼ How to Initialize a Client Using Proxy Credentials

1. Become superuser.
2. Run `ldapclient` (defining proxy values).

```
# ldapclient init -a
proxyDn=cn=proxyagent,ou=profile,dc=west,dc=example,dc=com -a
domainname=west.example.com -a profilename=pit1 -a
proxypassword=test1234 192.168.0.0

System successfully configured
```

The `-a proxyDn` and `-a proxypassword` are *required* if the profile to be used is setup for proxy. As the credentials are not stored in the profile saved on the server, you need to supply the information when you initialize the client. This method is more secure than the older method of storing the proxy credentials on the server.

The proxy info will be used to create the `/var/ldap/ldap_client_cred` and the rest of the information will be put in `/var/ldap/ldap_client_file`.

Note – DO NOT edit either the client configuration files directly. Use `ldapclient` to create or modify the content of these files.

Initializing a Client Manually

Superusers can perform manual client configurations. However, many of the checks are bypassed during the process, so it is relatively easy to mis-configure your system. In addition, you must change settings *on every machine*, instead of in one central place, as is done when using profiles.

▼ How to Initialize a Client Manually

1. Become superuser.
2. Use `ldapclient manual`.

```
# ldapclient manual -a domainName=dc=west.example.com \
```

```
-a credentialLevel=proxy -a defaultSearchBase=dc=west,  
dc=example, dc=com \  
-a proxyDN=cn=proxyagent,ou=profile,dc=west,dc=example,dc=com \  
-a proxyPassword=testtest 192.168.0.0
```

3. Use `ldapclient list` to verify.

```
NS_LDAP_FILE_VERSION= 2.0  
NS_LDAP_BINDDN= cn=proxyagent,ou=profile,dc=west,dc=example,dc=com  
NS_LDAP_BINDPASSWD= {NS1}4a3788e8c053424f  
NS_LDAP_SERVERS= 192.168.0.0  
NS_LDAP_SEARCH_BASEDN= dc=west,dc=example,dc=com  
NS_LDAP_CREDENTIAL_LEVEL= proxy
```

Modifying a Manual Client Configuration

▼ How to Modify a Manual Configuration

1. Become superuser
2. Use the `ldapclient mod` to change the authentication method to `simple`.
`# ldapclient mod -a authenticationMethod=simple`
3. Use `ldapclient list` to verify the change was made.

```
# ldapclient list  
  
NS_LDAP_FILE_VERSION= 2.0  
NS_LDAP_BINDDN= cn=proxyagent,ou=profile,dc=west,dc=example,dc=com  
NS_LDAP_BINDPASSWD= {NS1}4a3788e8c053424f  
NS_LDAP_SERVERS= 192.168.0.0  
NS_LDAP_SEARCH_BASEDN= dc=west,dc=example,dc=com  
NS_LDAP_AUTH= simple  
NS_LDAP_CREDENTIAL_LEVEL= proxy
```

Uninitializing a Client

▼ How to Uninitialize a Client

1. Become superuser.
2. Use `ldapclient uninit`.

```
# ldapclient uninit  
  
System successfully recovered
```

`ldapclient uninit` restores the client name service to what it was prior to the most recent `init`, `modify`, or `manual` operation. In other words, it performs an “undo” on the last step taken. For example, if the client was configured to use `profile1` and was then changed to use `profile2`, using `ldapclient uninit` would revert the client back to using `profile1`.

Setting Up TLS Security



Caution – The `cert7.db` and `key3.db` files must be readable by everyone. Be sure not to include any private keys in the `key3.db` file.

If using TLS, the necessary security databases must be installed. In particular, the files `cert7.db` and `key3.db` are needed. The `cert7.db` file contains the database of trusted certificates. The `key3.db` file contains the client’s keys. Although the LDAP naming service client does not use client keys, this file must be present.

Before running `ldapclient`, you should set up and install the needed security database files described in this section.

See the section ‘Configuring LDAP Clients to Use SSL’ in the Managing SSL chapter of the Sun ONE Directory Server 5.1 Administrator’s Guide for information on how to create and manage these files. Once configured, these files must be stored in the location expected by the LDAP naming service client. The attribute `certificatePath` is used to determine this location. This is by default `/var/ldap`.

For example, after setting up the necessary `cert7.db` and `key3.db` files using Netscape Communicator, copy them to the default location.

```
# cp $HOME/.netscape/cert7.db /var/ldap
```

```
# cp $HOME/.netscape/key3.db /var/ldap
```

Next, give everyone read access.

```
# chmod 444 /var/ldap/cert7.db
```

```
# chmod 444 /var/ldap/key3.db
```

Note – Netscape will manage the `cert7.db` and `key3.db` in the `$HOME/.netscape` directory. Copies of these security databases must be stored on a local file system if you are using them for the LDAP naming service client.

Configuring PAM

Using pam_ldap Without Password Management Support

If you are using `pam_ldap` *without* password management support, follow the sample `pam.conf` file in “Example `pam.conf` File for `pam_ldap`” on page 264. Add the lines that contain `pam_ldap.so.1` to the client’s `/etc/pam.conf` file. For details, see `pam.conf(4)`.

Configuring pam_ldap for Password Management Support

If you need to configure `pam_ldap` for password management support, copy the sample `pam.conf` file in “Example `pam.conf` file for `pam_ldap` Configured for Password Management” on page 266. Then, add the lines that contain `pam_ldap.so.1` to the client’s `/etc/pam.conf` file. In addition, if any PAM module in the sample `pam.conf` file specifies the `binding` control flag and the `server_policy` option, use the same flag and option for the corresponding module in the client’s `/etc/pam.conf` file. Also, add the `server_policy` option to the line that contains the service module `pam_authtok_store.so.1`.

- The `binding` control flag
Using the `binding` control flag allows a local password override of a remote (LDAP) password. For example, if a user account is found on both the local files and the LDAP namespace, the password associated with the local account takes precedence over the remote password. Thus, if the local password expires, authentication fails even if the remote LDAP password is still valid.
- The `server_policy` option
The `server_policy` option instructs `pam_unix` to ignore a user found in the LDAP namespace and to allow `pam_ldap` to perform authentication or account validation. In the case of `pam_authtok_store.so.1`, a new password is passed to the LDAP server without encryption. The password is thereby stored in the directory according to the password encryption scheme configured on the server. For more information, see `pam.conf(4)` and `pam_ldap(5)`.

Retrieving LDAP Naming Services Information

Using `ldaplist`

`ldaplist` is an LDAP utility to list the naming information from the LDAP servers in LDIF format. It can be useful for troubleshooting. See `ldaplist(1)` for further information.

Listing All LDAP Containers

`ldaplist` displays its output with a blank line separating records, which is helpful for big multiline records.

Note – The output of `ldaplist` depends upon the client configuration. For example, if the value of `ns_ldap_search` is `sub` rather than `one`, `ldaplist` lists all the entries under the current search `baseDN`.

The following is an example of `ldaplist` output.

```
# ldaplist
dn: ou=people,dc=west,dc=example,dc=com
dn: ou=group,dc=west,dc=example,dc=com
dn: ou=rpc,dc=west,dc=example,dc=com
dn: ou=protocols,dc=west,dc=example,dc=com
dn: ou=networks,dc=west,dc=example,dc=com
dn: ou=netgroup,dc=west,dc=example,dc=com
dn: ou=aliases,dc=west,dc=example,dc=com
dn: ou=hosts,dc=west,dc=example,dc=com
dn: ou=services,dc=west,dc=example,dc=com
dn: ou=ethers,dc=west,dc=example,dc=com
```

```
dn: ou=profile,dc=west,dc=example,dc=com
dn: automountmap=auto_home,dc=west,dc=example,dc=com
dn: automountmap=auto_direct,dc=west,dc=example,dc=com
dn: automountmap=auto_master,dc=west,dc=example,dc=com
dn: automountmap=auto_shared,dc=west,dc=example,dc=com
```

Listing All User Entry Attributes

To list specific information such as a user's passwd entry, use `getent` as follows.

```
# getent passwd user1
```

```
user1::30641:10:Joe Q. User:/home/user1:/bin/csh
```

If you want to list all attributes, use `ldaplist` with the `-l` option.

```
# ldaplist -l passwd user1
```

```
dn: uid=user1,ou=People,dc=west,dc=example,dc=com
   uid: user1
   cn: user1
   uidNumber: 30641
   gidNumber: 10
   gecos: Joe Q. User
   homeDirectory: /home/user1
   loginShell: /bin/csh
   objectClass: top
   objectClass: shadowAccount
   objectClass: account
   objectClass: posixAccount
   shadowLastChange: 6445
   userPassword: {crypt}J6v1YXRU.sW8c
```

Customizing the Client Environment

There are a couple of things you can tune in your client environment to make things work the way you want.

Modifying the `nsswitch.conf` File

You can modify your `/etc/nsswitch.conf` file to customize where each service gets its information. The default settings are stored in `/etc/nsswitch.ldap` and `ldapclient` uses this file to create your `/etc/nsswitch.conf` file when the client is initialized.

If you want to enable DNS by setting up a `/etc/resolv.conf` file, you will want to add DNS to your hosts lines as shown below.

```
hosts:      ldap dns [NOTFOUND=return] files
```

You can change any of the services, but be careful, because if the data is not populated on the server for the service specified things will stop working. In some cases files may not be setup by default as well.

LDAP Troubleshooting (Reference)

This chapter describes configuration problems and suggested solutions.

Monitoring Client Status

This section shows various commands that can be used to help determine the state of the LDAP client environment. For more information see the section on troubleshooting which will give more information on common problems and how to solve them. Also see the man pages for additional information on the options that can be used.

Verifying `ldap_cachemgr` is running

The `ldap_cachemgr` daemon must be running and functioning correctly at all times. Otherwise, nothing works. There are two ways to check if `ldap_cachemgr` is running.

- Use `ps -ef`.

```
# ps -ef | grep ldap_cachemgr
```

- Pass the `-g` option to `ldap_cachemgr`.

This causes it to dump the following status information, which is useful when you must diagnose a problem.

```
# /usr/lib/ldap/ldap_cachemgr -g

cachemgr configuration:
server debug level      0
server log file "/var/ldap/cachemgr.log"
number of calls to ldapcachemgr 19
```

```

cachemgr cache data statistics:
Configuration refresh information:
  Previous refresh time: 2001/11/16 18:33:28
  Next refresh time:    2001/11/16 18:43:28
Server information:
  Previous refresh time: 2001/11/16 18:33:28
  Next refresh time:    2001/11/16 18:36:08
  server: 192.168.0.0, status: UP
  server: 192.168.0.1, status: ERROR
    error message: Can't connect to the LDAP server
Cache data information:
  Maximum cache entries:      256
  Number of cache entries:    2

```

Checking the Current Profile Information

Become superuser and run `ldapclient` with the `list` option.

```

# ldapclient list

NS_LDAP_FILE_VERSION= 2.0
NS_LDAP_BINDDN= cn=proxyagent,ou=profile,dc=west,dc=example,dc=com
NS_LDAP_BINDPASSWD= {NS1}4a3788e8c053424f
NS_LDAP_SERVERS= 192.168.0.0, 192.168.0.1
NS_LDAP_SEARCH_BASEDN= dc=west,dc=example,dc=com
NS_LDAP_AUTH= simple
NS_LDAP_SEARCH_REF= TRUE
NS_LDAP_SEARCH_SCOPE= one
NS_LDAP_SEARCH_TIME= 30
NS_LDAP_SERVER_PREF= 192.168.0.0
NS_LDAP_PROFILE= pit1
NS_LDAP_CREDENTIAL_LEVEL= proxy
NS_LDAP_SERVICE_SEARCH_DESC= passwd:ou=people,?sub
NS_LDAP_SERVICE_SEARCH_DESC= group:ou=group,dc=west,dc=example,dc=com?one
NS_LDAP_BIND_TIME= 5

```

Currently the `/var/ldap` files are in ASCII format, but that could change to binary at some time and cating the files would cause problems. `ldapclient list` is the supported method for accessing this information.

Verifying Basic Client/Server Communication

The best way to show that your client is talking to the LDAP server is with the `ldaplist` command. The simplest form, `ldaplist` with no arguments will dump all the containers on the server. This works as long as the containers exist, and do not have to be populated. If the first step works, you can try `ldaplist passwd username` or `ldaplist hosts hostname` but if they contain lots of data you might want to pick a less populated service, or pipe them to head or more.

Checking Server Data From a Non-client Machine

Most of the commands above assume you are already an LDAP client. If you have not created a client and want to check the data on the server, use the `ldapsearch` command. The following example lists all of the containers.

```
# ldapsearch -h server1 -b "dc=west,dc=example,dc=com" -s one  
"objectclass=*" 
```

Configuration Problems and Solutions

The following discussion briefly describes LDAP configuration problems and suggested solutions to the problems.

Unresolved Hostname

The Solaris operating environment LDAP client back end returns fully qualified hostnames for host lookups, such as hostnames returned by `gethostbyname(3N)` and `getipnodebyname(3N)`. If the name stored is qualified that is contains at least one dot, the client returns the name as is. For example, if the name stored is `hostB.eng`, the returned name is `hostB.eng`.

If the name stored in the LDAP directory is not qualified (it does not contain any dot), the client back end appends the domain part to the name. For example, if the name stored is `hostA`, the returned name is `hostA.domainname`.

Unable to Reach Systems in the LDAP Domain Remotely

If the DNS domain name is different from the LDAP domain name, then the LDAP naming service cannot be used to serve host names unless the host names are stored fully qualified.

Login Does Not Work

LDAP clients use the `pam(3)` modules for user authentication during the logins. When using the standard UNIX™ PAM module, the password is read from the server and checked on the client side. This can fail due to one of the following reasons.

1. `ldap` is not used by the `passwd` service in the `/etc/nsswitch.conf` file.
2. The user's `userPassword` attribute on the server list is not readable by the proxy agent. You need to allow at least the proxy agent to read the password because the proxy agent returns it to the client for comparison. `pam_ldap` does not require read access to the password.
3. The proxy agent might not have the correct password.
4. The entry does not have the `shadowAccount` object class.
5. No password is defined for the user.

When you use `ldapaddent`, you must use the `-p` option to ensure that the password is added to the user entry. If you use `ldapaddent` without the `-p` option, the user's password is not stored in the directory unless you also add the `/etc/shadow` file by using `ldapaddent`.

6. No LDAP servers are reachable.
Check the status of the servers.

```
# /usr/lib/ldap/ldap_cachemgr -g
```
7. `pam.conf` is configured incorrectly.
8. The user is not defined in the LDAP namespace.
9. `NS_LDAP_CREDENTIAL_LEVEL` is set to `anonymous` for `pam_unix`, and `userPassword` is not available to anonymous users.
10. The password is not stored in `crypt` format.
11. If `pam_ldap` is configured to support password management, login failure could be the result of one of the following:
 - The user's password has expired.
 - The user's account is locked out due to too many failed login attempts.
 - The user's account has been deactivated by the administrator.

Lookup Too Slow

The LDAP database relies on indexes to improve search performance. A major performance degradation occurs when indexes are improperly configured. The documentation includes a common set of attributes that should be indexed. You can also add your own indexes to improve performance at your site.

ldapclient Cannot Bind to Server

`ldapclient` failed to initialize the client when using the `init` profile option. Possible reasons for failure include the following:

1. The incorrect domain name was specified on the command line.
2. The `nisDomain` attribute is not set in the DIT to represent the entry point for the specified client domain.

3. Access control information is not set up properly on the server, thus disallowing anonymous search in the LDAP database.
4. Incorrect server address passed to the `ldapclient` command. Use `ldapsearch(1)` to verify the server address
5. Incorrect profile name passed to the `ldapclient` command. Use `ldapsearch(1)` to verify the profile name in the DIT.
6. Use `snoop(1M)` on the client's network interface to see what sort of traffic is going out, and determine to which server it is talking.

Using `ldap_cachemgr` for Debugging

Using `ldap_cachemgr` with the `-g` option can be a useful way to debug, as you can view the current client configuration and statistics. For example,

```
# ldap_cachemgr -g
```

would print current configuration and statistics to standard output, including the status of all LDAP servers, as mentioned previously. Note that you do *not* need to become super user to execute this command.

`ldapclient` Hangs During Setup

If the `ldapclient` command hangs, hitting Ctrl-C will exit after restoring the previous environment. If this happens, check with the server administrator to make sure the server is running.

Also check the server list attributes on either the profile or the command line and make sure the server information is correct.

Frequently Asked Questions

- Can I use LDAP naming services with older Solaris Releases?
Currently, LDAP is only supported in Solaris 8 and Solaris 9. For differences between the two see “New LDAP Naming Services Features for Solaris 9” on page 200.
- What are the DIT Default Locations in Solaris LDAP Naming Services?
See “Default Directory Information Tree (DIT)” on page 206.

LDAP General Reference (Reference)

1. "Blank Checklists" on page 261
2. "Upgrade Information" on page 262
3. "LDAP Commands" on page 263
4. "Example pam.conf File for pam_ldap" on page 264
5. "IETF Schemas" on page 267
6. "Directory User Agent Profile (DUAProfile) Schema" on page 274
7. "Solaris Schemas" on page 276
8. "Internet Print Protocol Information" on page 278
9. "Generic Directory Server Requirements" on page 286
10. "Default Filters Used by LDAP Naming Services" on page 287

Blank Checklists

TABLE 18-1 Server Variables Defined

Variable	Definition for _____ Network
Port number at which an instance of the directory server is installed (DEFAULT=389)	
Name of server	
Replica server(s) (IP number:port number)	
Directory manager [dn: cn=directory manager]	
Domain name to be served	

TABLE 18-1 Server Variables Defined (Continued)

Variable	Definition for _____ Network
Maximum time (in seconds) to process client requests before timing out	
Maximum number of entries returned for each search request	

TABLE 18-2 Client Profile Variables Defined

Variable	Definition for _____ Network
Profile name	
Server list (defaults to the local subnet)	
Preferred server list (listed in order of which server to try first, second, and so on)	
Search scope (number of levels down through the directory tree. 'One' or 'Sub')	
Credential used to gain access to server. Default is anonymous	
Follow Referrals? (a pointer to another server if the main server is unavailable) Default is no.	
Search time limit (in seconds, default 30) for waiting for server to return information.	
Bind time limit (in seconds, default 30) for contacting server. The default is seconds.	
Authentication method Default is none.	

Upgrade Information

Solaris 9 clients are fully compatible with directory servers set up to serve Solaris 8 clients. `ldapclient(1M)` can simply download such a profile and configure the client using version 1 profiles. However to take advantage of new features built into Solaris 9 and to use the new security model, version 2 profiles must be used.

Servers can serve a mix of both old and new clients so that both clients see the same results from the server as long as schema mapping is not enabled and version 2 profiles are not configured to use special filters in `serviceSearchDescriptors`. Obviously if the server is not using the default schema, older clients cannot use that server as Solaris 8 clients cannot arbitrarily map their schema.

One additional change that also should be considered is that in Solaris 8 clients, running `ldap_cachemgr()` was recommended, but optional. In Solaris 9, `ldap_cachemgr()` *must* be running at all times. The daemon is *required* for the client to function properly.

New automount Schema

By default, Solaris 9 uses a new schema for `automount` entries instead of using generic NIS map schema which Solaris 8 clients used. This means that if you set up a server with Solaris 9 tools, Solaris 8 clients cannot see the `automount` entries. For sites where the server being setup is to serve both Solaris 9 and Solaris 8 clients, a profile can be created to map the schema to the old one before adding `automounter` entries. This would ensure that `ldapaddent(1M)` adds the entries using the old schema. However, note that this would also mean that all Solaris 9 clients must use a profile where the schema for `automount` is mapped.

You need to add the following mapping attributes to your profile for this mapping to take effect.

```
attributeMap:      automount:automountMapName=nisMapName
attributeMap:      automount:automountKey=cn
attributeMap:      automount:automountInformation=nisMapEntry
objectclassMap:    automount:automountMap=nisMap
objectclassMap:    automount:automount=nisObject
```

LDAP Commands

There are two sets of LDAP-related commands in Solaris. One set is the general LDAP tools, which do not require the client to be configured with the LDAP naming service. The second set uses the common LDAP configuration on the client and therefore can only be used if the client is configured to use LDAP as its naming service.

General LDAP Tools

LDAP command-line tools support a common set of options, including authentication and bind parameters.

These commands can be used to manipulate directory entries directly. The `ldapsearch(1)`, `ldapmodify(1)`, `ldapadd(1)`, and `ldapdelete(1)` tools support a common text-based format for representing directory information called the LDAP Data Interchange Format (LDIF).

LDAP Tools Requiring LDAP Naming Services

TABLE 18-3 Tools (From Section 1 Man Pages)

Tool	Function
ldapaddent(1M)	Used to create entries in LDAP containers from the corresponding <code>/etc</code> files. This tool allows populating the directory from files. For example, it reads <code>/etc/passwd</code> format file and populates <code>passwd</code> entries in the directory.
ldaplist(1)	Used to list contents of various services from the directory.
idsconfig(1M)	Used to set up Sun ONE Directory Server 5.1 to serve LDAP naming service clients.

Example `pam.conf` File for `pam_ldap`

```
#
# Authentication management
#
# login service (explicit because of pam_dial_auth)
#
login    auth required          pam_authtok_get.so.1
login    auth required          pam_dhkeys.so.1
login    auth required          pam_dial_auth.so.1
login    auth sufficient         pam_unix_auth.so.1
login    auth required          pam_ldap.so.1 try_first_pass
#
# rlogin service (explicit because of pam_rhost_auth)
#
rlogin   auth sufficient         pam_rhosts_auth.so.1
rlogin   auth required          pam_authtok_get.so.1
rlogin   auth required          pam_dhkeys.so.1
rlogin   auth sufficient         pam_unix_auth.so.1
rlogin   auth required          pam_ldap.so.1 try_first_pass
#
# rsh service (explicit because of pam_rhost_auth)
#
rsh      auth sufficient         pam_rhosts_auth.so.1
rsh      auth required          pam_authtok_get.so.1
rsh      auth required          pam_dhkeys.so.1
rsh      auth sufficient         pam_unix_auth.so.1
rsh      auth required          pam_ldap.so.1 try_first_pass
#
# PPP service (explicit because of pam_dial_auth)
#
```

```

ppp      auth required          pam_authtok_get.so.1
ppp      auth required          pam_dhkeys.so.1
ppp      auth required          pam_dial_auth.so.1
ppp      auth sufficient        pam_unix_auth.so.1
ppp      auth required          pam_ldap.so.1 try_first_pass
#
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication

#
other    auth required          pam_authtok_get.so.1
other    auth required          pam_dhkeys.so.1
other    auth sufficient        pam_unix_auth.so.1
other    auth required          pam_ldap.so.1 try_first_pass
#
# passwd command (explicit because of a different authentication module)

#
passwd  auth sufficient         pam_passwd_auth.so.1
passwd  auth required          pam_ldap.so.1 try_first_pass
#
# cron service (explicit because of non-usage of pam_roles.so.1)
#
cron    account required        pam_projects.so.1
cron    account required        pam_unix_account.so.1
#
# Default definition for Account management
# Used when service name is not explicitly mentioned for account
management
#
other    account requisite      pam_roles.so.1
other    account required       pam_projects.so.1
other    account required       pam_unix_account.so.1
#
# Default definition for Session management
# Used when service name is not explicitly mentioned for session
management
#
other    session required        pam_unix_session.so.1
#
# Default definition for Password management
# Used when service name is not explicitly mentioned for password
management
#
other    password required       pam_dhkeys.so.1
other    password required       pam_authtok_get.so.1
other    password required       pam_authtok_check.so.1
other    password sufficient     pam_authtok_store.so.1
other    password required       pam_ldap.so.1
#
# Support for Kerberos V5 authentication (uncomment to use Kerberos)
#
#rlogin      auth optional          pam_krb5.so.1 try_first_pass
#login       auth optional          pam_krb5.so.1 try_first_pass
#other       auth optional          pam_krb5.so.1 try_first_pass

```

```

#cron          account optional      pam_krb5.so.1
#other        account optional      pam_krb5.so.1
#other        session optional      pam_krb5.so.1
#other        password optional     pam_krb5.so.1 try_first_pass

```

Example pam_conf file for pam_ldap Configured for Password Management

```

# PAM configuration
#
# Authentication management
#
# login service (explicit because of pam_dial_auth)
#
login  auth      requisite pam_authtok_get.so.1
login  auth      required  pam_dhkeys.so.1
login  auth      required  pam_dial_auth.so.1
login  auth      binding   pam_unix_auth.so.1 server_policy
login  auth      required  pam_ldap.so.1
#
# rlogin service (explicit because of pam_rhost_auth)
#
rlogin auth      sufficient pam_rhosts_auth.so.1
rlogin auth      requisite  pam_authtok_get.so.1
rlogin auth      required  pam_dhkeys.so.1
rlogin auth      binding   pam_unix_auth.so.1 server_policy
rlogin auth      required  pam_ldap.so.1
#
# rsh service (explicit because of pam_rhost_auth,
# and pam_unix_auth for meaningful pam_setcred)
#
rsh    auth sufficient      pam_rhosts_auth.so.1
rsh    auth required       pam_unix_auth.so.1
#
# PPP service (explicit because of pam_dial_auth)
#
ppp    auth requisite      pam_authtok_get.so.1
ppp    auth required       pam_dhkeys.so.1
ppp    auth required       pam_dial_auth.so.1
ppp    auth binding        pam_unix_auth.so.1 server_policy
ppp    auth required       pam_ldap.so.1
#
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
other  auth requisite      pam_authtok_get.so.1
other  auth required       pam_dhkeys.so.1
other  auth binding        pam_unix_auth.so.1 server_policy

```

```

other    auth required          pam_ldap.so.1
#
# passwd command (explicit because of a different authentication module)
#
passwd auth    binding    pam_passwd_auth.so.1 server_policy
passwd auth    required   pam_ldap.so.1
#
# cron service (explicit because of non-usage of pam_roles.so.1)
#
cron    account required          pam_projects.so.1
cron    account required          pam_unix_account.so.1
#
# Default definition for Account management
# Used when service name is not explicitly mentioned for account management
#
other    account requisite pam_roles.so.1
other    account required   pam_projects.so.1
other    account binding    pam_unix_account.so.1 server_policy
other    account required   pam_ldap.so.1
#
# Default definition for Session management
# Used when service name is not explicitly mentioned for session management
#
other    session required          pam_unix_session.so.1
#
# Default definition for Password management
# Used when service name is not explicitly mentioned for password management
#
other    password required   pam_dhkeys.so.1
other    password requisite  pam_authtok_get.so.1
other    password requisite  pam_authtok_check.so.1
other    password required   pam_authtok_store.so.1 server_policy
#
# Support for Kerberos V5 authentication (uncomment to use Kerberos)
#
#rlogin          auth optional          pam_krb5.so.1 try_first_pass
#login           auth optional          pam_krb5.so.1 try_first_pass
#other           auth optional          pam_krb5.so.1 try_first_pass
#cron            account optional       pam_krb5.so.1
#other           account optional       pam_krb5.so.1
#other           session optional       pam_krb5.so.1
#other           password optional      pam_krb5.so.1 try_first_pass

```

IETF Schemas

Schemas are definitions describing what types of information can be stored as entries in a server's directory.

In order for a directory server to support Solaris 9 LDAP naming clients, schemas defined in this chapter must be configured in the server unless schema is mapped using the schema mapping feature of the clients.

There are four required LDAP schemas defined by IETF: the RFC 2307 Network Information Service schema, the LDAP mailgroups Internet draft, and the LDAP Internet Print Protocol (IPP) draft schema (listed under Solaris schemas). To support the Naming Information Service, the definition of these schemas must be added to the directory server. The various RFCs can also be accessed on the IETF Web site. <http://www.ietf.org>.

Note – Internet drafts are draft documents valid for a maximum of six months and might be updated, or rendered obsolete by other documents at any time.

RFC 2307 Network Information Service Schema

The LDAP servers must be configured to support the revised RFC 2307.

The nisSchema OID is 1.3.6.1.1. The RFC 2307 Attributes are the following.

```
( nisSchema.1.0 NAME 'uidNumber'
DESC 'An integer uniquely identifying a user in an
      administrative domain'
EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )
```

```
( nisSchema.1.1 NAME 'gidNumber'
DESC 'An integer uniquely identifying a group in an
      administrative domain'
EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )
```

```
( nisSchema.1.2 NAME 'gecos'
DESC 'The GECOS field; the common name'
EQUALITY caseIgnoreIA5Match
SUBSTRINGS caseIgnoreIA5SubstringsMatch
SYNTAX 'IA5String' SINGLE-VALUE )
```

```
( nisSchema.1.3 NAME 'homeDirectory'
DESC 'The absolute path to the home directory'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' SINGLE-VALUE )
```

```
( nisSchema.1.4 NAME 'loginShell'
DESC 'The path to the login shell'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' SINGLE-VALUE )
```

```
( nisSchema.1.5 NAME 'shadowLastChange'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )
```

```

( nisSchema.1.6 NAME 'shadowMin'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.7 NAME 'shadowMax'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.8 NAME 'shadowWarning'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.9 NAME 'shadowInactive'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.10 NAME 'shadowExpire'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.11 NAME 'shadowFlag'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.12 NAME 'memberUid'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String' )

( nisSchema.1.13 NAME 'memberNisNetgroup'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String' )

( nisSchema.1.14 NAME 'nisNetgroupTriple'
DESC 'Netgroup triple'
SYNTAX 'nisNetgroupTripleSyntax' )

( nisSchema.1.15 NAME 'ipServicePort'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.16 NAME 'ipServiceProtocol'
SUP name )

( nisSchema.1.17 NAME 'ipProtocolNumber'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.18 NAME 'oncrpcNumber'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.19 NAME 'ipHostNumber'

```

```

DESC 'IP address as a dotted decimal, eg. 192.168.1.1
      omitting leading zeros'
SUP name )

( nisSchema.1.20 NAME 'ipNetworkNumber'
DESC 'IP network as a dotted decimal, eg. 192.168,
      omitting leading zeros'
SUP name SINGLE-VALUE )

( nisSchema.1.21 NAME 'ipNetmaskNumber'
DESC 'IP netmask as a dotted decimal, eg. 255.255.255.0,
      omitting leading zeros'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}' SINGLE-VALUE )

( nisSchema.1.22 NAME 'macAddress'
DESC 'MAC address in maximal, colon separated hex
      notation, eg. 00:00:92:90:ee:e2'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}' )

( nisSchema.1.23 NAME 'bootParameter'
DESC 'rpc.bootparamd parameter'
SYNTAX 'bootParameterSyntax' )

( nisSchema.1.24 NAME 'bootFile'
DESC 'Boot image name'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' )

( nisSchema.1.26 NAME 'nisMapName'
SUP name )

( nisSchema.1.27 NAME 'nisMapEntry'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String{1024}' SINGLE-VALUE )

( nisSchema.1.28 NAME 'nisPublicKey'
DESC 'NIS public key'
SYNTAX 'nisPublicKeySyntax' )

( nisSchema.1.29 NAME 'nisSecretKey'
DESC 'NIS secret key'
SYNTAX 'nisSecretKeySyntax' )

( nisSchema.1.30 NAME 'nisDomain'
DESC 'NIS domain'
SYNTAX 'IA5String' )

( nisSchema.1.31 NAME 'automountMapName'
DESC 'automount Map Name'
EQUALITY caseExactIA5Match
SUBSTR caseExactIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )

```

```
( nisSchema.1.32 NAME 'automountKey'
DESC 'Automount Key value'
EQUALITY caseExactIA5Match
SUBSTR caseExactIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

```
( nisSchema.1.33 NAME 'automountInformation'
DESC 'Automount information'
EQUALITY caseExactIA5Match
SUBSTR caseExactIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

The nisSchema OID is 1.3.6.1.1. The RFC 2307 objectClasses are the following.

```
( nisSchema.2.0 NAME 'posixAccount' SUP top AUXILIARY
DESC 'Abstraction of an account with POSIX attributes'
MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
MAY ( userPassword $ loginShell $ gecos $ description ) )
```

```
( nisSchema.2.1 NAME 'shadowAccount' SUP top AUXILIARY
DESC 'Additional attributes for shadow passwords'
MUST uid
MAY ( userPassword $ shadowLastChange $ shadowMin
shadowMax $ shadowWarning $ shadowInactive $
shadowExpire $ shadowFlag $ description ) )
```

```
( nisSchema.2.2 NAME 'posixGroup' SUP top STRUCTURAL
DESC 'Abstraction of a group of accounts'
MUST ( cn $ gidNumber )
MAY ( userPassword $ memberUid $ description ) )
```

```
( nisSchema.2.3 NAME 'ipService' SUP top STRUCTURAL
DESC 'Abstraction an Internet Protocol service.
Maps an IP port and protocol (such as tcp or udp)
to one or more names; the distinguished value of
the cn attribute denotes the service's canonical
name'
MUST ( cn $ ipServicePort $ ipServiceProtocol )
MAY ( description ) )
```

```
( nisSchema.2.4 NAME 'ipProtocol' SUP top STRUCTURAL
DESC 'Abstraction of an IP protocol. Maps a protocol number
to one or more names. The distinguished value of the cn
attribute denotes the protocol's canonical name'
MUST ( cn $ ipProtocolNumber )
MAY description )
```

```
( nisSchema.2.5 NAME 'oncRpc' SUP top STRUCTURAL
DESC 'Abstraction of an Open Network Computing (ONC)
[RFC1057] Remote Procedure Call (RPC) binding.
This class maps an ONC RPC number to a name.
The distinguished value of the cn attribute denotes
the RPC service's canonical name'
```

```

MUST ( cn $ oncRpcNumber $ description )
MAY description )

( nisSchema.2.6 NAME 'ipHost' SUP top AUXILIARY
  DESC 'Abstraction of a host, an IP device. The distinguished
        value of the cn attribute denotes the host's canonical
        name. Device SHOULD be used as a structural class'
  MUST ( cn $ ipHostNumber )
  MAY ( 1 $ description $ manager $ userPassword ) )

( nisSchema.2.7 NAME 'ipNetwork' SUP top STRUCTURAL
  DESC 'Abstraction of a network. The distinguished value of
        the cn attribute denotes the network's canonical name'
  MUST ipNetworkNumber
  MAY ( cn $ ipNetmaskNumber $ 1 $ description $ manager ) )

( nisSchema.2.8 NAME 'nisNetgroup' SUP top STRUCTURAL
  DESC 'Abstraction of a netgroup. May refer to other netgroups'
  MUST cn
  MAY ( nisNetgroupTriple $ memberNisNetgroup $ description ) )

( nisSchema.2.9 NAME 'nisMap' SUP top STRUCTURAL
  DESC 'A generic abstraction of a NIS map'
  MUST nisMapName
  MAY description )

( nisSchema.2.10 NAME 'nisObject' SUP top STRUCTURAL
  DESC 'An entry in a NIS map'
  MUST ( cn $ nisMapEntry $ nisMapName )
  MAY description )

( nisSchema.2.11 NAME 'ieee802Device' SUP top AUXILIARY
  DESC 'A device with a MAC address; device SHOULD be
        used as a structural class'
  MAY macAddress )

( nisSchema.2.12 NAME 'bootableDevice' SUP top AUXILIARY
  DESC 'A device with boot parameters; device SHOULD be
        used as a structural class'
  MAY ( bootFile $ bootParameter ) )

( nisSchema.2.14 NAME 'nisKeyObject' SUP top AUXILIARY
  DESC 'An object with a public and secret key'
  MUST ( cn $ nisPublicKey $ nisSecretKey )
  MAY ( uidNumber $ description ) )

( nisSchema.2.15 NAME 'nisDomainObject' SUP top AUXILIARY
  DESC 'Associates a NIS domain with a naming context'
  MUST nisDomain )

( nisSchema.2.16 NAME 'automountMap' SUP top STRUCTURAL
  MUST ( automountMapName )
  MAY description )

( nisSchema.2.17 NAME 'automount' SUP top STRUCTURAL

```

```
DESC 'Automount information'
MUST ( automountKey $ automountInformation )
MAY description )
```

Mail Alias Schema

Mail alias information uses the schema defined by the LDAP Mailgroups Internet draft, formerly known as the draft-steinback-ldap-mailgroups draft. Until a new schema becomes available, Solaris LDAP clients will continue to use this schema for mail alias information.

The original LDAP Mailgroups schema contains a large number of attributes and object classes. Only two attributes and a single object class are used by Solaris clients. These are listed below.

The mail alias Attributes are the following.

```
( 0.9.2342.19200300.100.1.3
  NAME 'mail'
  DESC 'RFC822 email address for this person'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(256)'
  SINGLE-VALUE )

( 2.16.840.1.113730.3.1.30
  NAME 'mgrpRFC822MailMember'
  DESC 'RFC822 mail address of email only member of group'
  EQUALITY CaseIgnoreIA5Match
  SYNTAX 'IA5String(256)' )
```

The mail alias objectClass is the following.

```
( 2.16.840.1.113730.3.2.4
  NAME 'mailGroup'
  SUP top
  STRUCTURAL
  MUST mail
  MAY ( cn $ mailAlternateAddress $ mailHost $ mailRequireAuth $
    mgrpAddHeader $ mgrpAllowedBroadcaster $ mgrpAllowedDomain $
    mgrpApprovePassword $ mgrpBroadcasterModeration $ mgrpDeliverTo $
    mgrpErrorsTo $ mgrpModerator $ mgrpMsgMaxSize $
    mgrpMsgRejectAction $ mgrpMsgRejectText $ mgrpNoMatchAddrs $
    mgrpRemoveHeader $ mgrpRFC822MailMember ))
```

Directory User Agent Profile (DUAProfile) Schema

The DUAConfSchemaOID is 1.3.6.1.4.1.11.1.3.1.

```
DESC 'Default LDAP server host address used by a DUA'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )

( DUAConfSchemaOID.1.1 NAME 'defaultSearchBase'
DESC 'Default LDAP base DN used by a DUA'
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
SINGLE-VALUE )

( DUAConfSchemaOID.1.2 NAME 'preferredServerList'
DESC 'Preferred LDAP server host addresses to be used by a
DUA'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )

( DUAConfSchemaOID.1.3 NAME 'searchTimeLimit'
DESC 'Maximum time in seconds a DUA should allow for a
search to complete'
EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE )

( DUAConfSchemaOID.1.4 NAME 'bindTimeLimit'
DESC 'Maximum time in seconds a DUA should allow for the
bind operation to complete'
EQUALITY integerMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE )

( DUAConfSchemaOID.1.5 NAME 'followReferrals'
DESC 'Tells DUA if it should follow referrals
returned by a DSA search result'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
SINGLE-VALUE )

( DUAConfSchemaOID.1.6 NAME 'authenticationMethod'
DESC 'A kestring which identifies the type of
authentication method used to contact the DSA'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )
```

```

( DUAConfSchemaOID.1.7 NAME 'profileTTL'
  DESC 'Time to live, in seconds, before a client DUA
  should re-read this configuration profile'
  'serviceSearchDescriptor'
  DESC 'LDAP search descriptor list used by a DUA'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

( DUAConfSchemaOID.1.9 NAME 'attributeMap'
  DESC 'Attribute mappings used by a DUA'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

( DUAConfSchemaOID.1.10 NAME 'credentialLevel'
  DESC 'Identifies type of credentials a DUA should
  use when binding to the LDAP server'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
  SINGLE-VALUE )

( DUAConfSchemaOID.1.11 NAME 'objectclassMap'
  DESC 'Objectclass mappings used by a DUA'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

( DUAConfSchemaOID.1.12 NAME 'defaultSearchScope' SINGLE-VALUE )

( DUAConfSchemaOID.1.13 NAME 'serviceCredentialLevel'
  DESC 'Identifies type of credentials a DUA
  should use when binding to the LDAP server for a
  specific service'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

( DUAConfSchemaOID.1.15 NAME 'serviceAuthenticationMethod'
  DESC 'Authentication Method used by a service of the DUA'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

  ( DUAConfSchemaOID.2.4 NAME 'DUAConfigProfile'
    SUP top STRUCTURAL
    DESC 'Abstraction of a base configuration for a DUA'
    MUST ( cn )
    MAY ( defaultServerList $ preferredServerList $
    defaultSearchBase $ defaultSearchScope $
    searchTimeLimit $ bindTimeLimit $
    credentialLevel $ authenticationMethod $
    followReferrals $ serviceSearchDescriptor $
    serviceCredentialLevel $ serviceAuthenticationMethod $
    objectclassMap $ attributeMap $
    profileTTL ) )

```

Solaris Schemas

The schemas required for the Solaris operating environment are the following.

- Solaris Projects schema
- Role-based access control and execution profile schemas
- Printer schemas

Solaris Projects Schema

`/etc/project` is a local source of attributes associated with projects. For more information, see `project(4)`.

The Project Attributes are the following.

```
( 1.3.6.1.4.1.42.2.27.5.1.1 NAME 'SolarisProjectID'
  DESC 'Unique ID for a Solaris Project entry'
  EQUALITY integerMatch
  SYNTAX INTEGER SINGLE )

( 1.3.6.1.4.1.42.2.27.5.1.2 NAME 'SolarisProjectName'
  DESC 'Name of a Solaris Project entry'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String SINGLE )

( 1.3.6.1.4.1.42.2.27.5.1.3 NAME 'SolarisProjectAttr'
  DESC 'Attributes of a Solaris Project entry'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String )

( 1.3.6.1.4.1.42.2.27.5.1.30 NAME 'memberGid'
  DESC 'Posix Group Name'
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' )
```

The Project objectClass is the following.

```
( 1.3.6.1.4.1.42.2.27.5.2.1 NAME 'SolarisProject'
  SUP top STRUCTURAL
  MUST ( SolarisProjectID $ SolarisProjectName )
  MAY ( memberUid $ memberGid $ description $ SolarisProjectAttr ) )
```

Role-Based Access Control and Execution Profile Schema

`/etc/user_attr` is a local source of extended attributes associated with users and roles. For more information, see `user_attr(4)`.

The role-based access control Attributes are the following.

```
( 1.3.6.1.4.1.42.2.27.5.1.4 NAME 'SolarisAttrKeyValue'
  DESC 'Semi-colon separated key=value pairs of attributes'
  EQUALITY caseIgnoreIA5Match
  SUBSTRINGS caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.7 NAME 'SolarisAttrShortDesc'
  DESC 'Short description about an entry, used by GUIs'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.8 NAME 'SolarisAttrLongDesc'
  DESC 'Detail description about an entry'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.9 NAME 'SolarisKernelSecurityPolicy'
  DESC 'Solaris kernel security policy'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.10 NAME 'SolarisProfileType'
  DESC 'Type of object defined in profile'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.11 NAME 'SolarisProfileId'
  DESC 'Identifier of object defined in profile'
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.12 NAME 'SolarisUserQualifier'
  DESC 'Per-user login attributes'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.13 NAME 'SolarisReserved1'
  DESC 'Reserved for future use'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.14 NAME 'SolarisReserved2'
  DESC 'Reserved for future use'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )
```

The role based access control objectClasses are the following.

```
( 1.3.6.1.4.1.42.2.27.5.2.3 NAME 'SolarisUserAttr' SUP top AUXILIARY
  DESC 'User attributes'
  MAY ( SolarisUserQualifier $ SolarisAttrReserved1 $ \
        SolarisAttrReserved2 $ SolarisAttrKeyValue ) )
```

```

( 1.3.6.1.4.1.42.2.27.5.2.4 NAME 'SolarisAuthAttr' SUP top STRUCTURAL
DESC 'Authorizations data'
MUST cn
MAY ( SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
      SolarisAttrShortDesc $ SolarisAttrLongDesc $ \
      SolarisAttrKeyValue ) )

( 1.3.6.1.4.1.42.2.27.5.2.5 NAME 'SolarisProfAttr' SUP top STRUCTURAL
DESC 'Profiles data'
MUST cn
MAY ( SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
      SolarisAttrLongDesc $ SolarisAttrKeyValue ) )

( 1.3.6.1.4.1.42.2.27.5.2.6 NAME 'SolarisExecAttr' SUP top AUXILIARY
DESC 'Profiles execution attributes'
MAY ( SolarisKernelSecurityPolicy $ SolarisProfileType $ \
      SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
      SolarisProfileId $ SolarisAttrKeyValue ) )

```

Internet Print Protocol Information

Internet Print Protocol (IPP) Attributes

```

( 1.3.18.0.2.4.1140
NAME 'printer-uri'
DESC 'A URI supported by this printer.
This URI SHOULD be used as a relative distinguished name (RDN).
If printer-xri-supported is implemented, then this URI value
MUST be listed in a member value of printer-xri-supported.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )

( 1.3.18.0.2.4.1107
NAME 'printer-xri-supported'
DESC 'The unordered list of XRI (extended resource identifiers) supported
by this printer.
Each member of the list consists of a URI (uniform resource identifier)
followed by optional authentication and security metaparameters.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

( 1.3.18.0.2.4.1135
NAME 'printer-name'
DESC 'The site-specific administrative name of this printer, more end-user

```

```

friendly than a URI.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} SINGLE-VALUE )

( 1.3.18.0.2.4.1119
NAME 'printer-natural-language-configured'
DESC 'The configured language in which error and status messages will be
generated (by default) by this printer.
Also, a possible language for printer string attributes set by operator,
system administrator, or manufacturer.
Also, the (declared) language of the "printer-name", "printer-location",
"printer-info", and "printer-make-and-model" attributes of this printer.
For example: "en-us" (US English) or "fr-fr" (French in France) Legal values of
language tags conform to [RFC3066] "Tags for the Identification of Languages".'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} SINGLE-VALUE )

( 1.3.18.0.2.4.1136
NAME 'printer-location'
DESC 'Identifies the location of the printer. This could include
things like: "in Room 123A", "second floor of building XYZ".'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} SINGLE-VALUE )

( 1.3.18.0.2.4.1139
NAME 'printer-info'
DESC 'Identifies the descriptive information about this printer.
This could include things like: "This printer can be used for
printing color transparencies for HR presentations", or
"Out of courtesy for others, please print only small (1-5 page)
jobs at this printer", or even "This printer is going away on July 1, 1997,
please find a new printer".'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE )

( 1.3.18.0.2.4.1134
NAME 'printer-more-info'
DESC 'A URI used to obtain more information about this specific printer.
For example, this could be an HTTP type URI referencing an HTML page
accessible to a Web Browser.
The information obtained from this URI is intended for end user consumption.'
EQUALITY caseIgnoreMatch ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )

( 1.3.18.0.2.4.1138
NAME 'printer-make-and-model'
DESC 'Identifies the make and model of the device.

```

The device manufacturer MAY initially populate this attribute.'

EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} SINGLE-VALUE)

(1.3.18.0.2.4.1133
NAME 'printer-ipp-versions-supported'
DESC 'Identifies the IPP protocol version(s) that this printer supports, including major and minor versions, i.e., the version numbers for which this Printer implementation meets the conformance requirements.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127})

(1.3.18.0.2.4.1132
NAME 'printer-multiple-document-jobs-supported'
DESC 'Indicates whether or not the printer supports more than one document per job, i.e., more than one Send-Document or Send-Data operation with document data.'
EQUALITY booleanMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE)

(1.3.18.0.2.4.1109
NAME 'printer-charset-configured'
DESC 'The configured charset in which error and status messages will be generated (by default) by this printer. Also, a possible charset for printer string attributes set by operator, system administrator, or manufacturer. For example: "utf-8" (ISO 10646/Unicode) or "iso-8859-1" (Latin1). Legal values are defined by the IANA Registry of Coded Character Sets and the "(preferred MIME name)" SHALL be used as the tag. For coherence with IPP Model, charset tags in this attribute SHALL be lowercase normalized. This attribute SHOULD be static (time of registration) and SHOULD NOT be dynamically refreshed attributetypes: (subsequently).'

EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{63} SINGLE-VALUE)

(1.3.18.0.2.4.1131
NAME 'printer-charset-supported'
DESC 'Identifies the set of charsets supported for attribute type values of type Directory String for this directory entry. For example: "utf-8" (ISO 10646/Unicode) or "iso-8859-1" (Latin1). Legal values are defined by the IANA Registry of Coded Character Sets and the preferred MIME name.'

EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{63})

(1.3.18.0.2.4.1137
NAME 'printer-generated-natural-language-supported'
DESC 'Identifies the natural language(s) supported for this directory entry. For example: "en-us" (US English) or "fr-fr" (French in France). Legal values conform to [RFC3066], Tags for the Identification of Languages.'

EQUALITY caseIgnoreMatch

```

ORDERING caseIgnoreOrderingMatch SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{63} )

( 1.3.18.0.2.4.1130
NAME 'printer-document-format-supported'
DESC 'The possible document formats in which data may be interpreted
and printed by this printer.
Legal values are MIME types come from the IANA Registry of Internet Media Types.'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} )

( 1.3.18.0.2.4.1129
NAME 'printer-color-supported'
DESC 'Indicates whether this printer is capable of any type of color printing
at all, including highlight color.'
EQUALITY booleanMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE )

( 1.3.18.0.2.4.1128
NAME 'printer-compression-supported'
DESC 'Compression algorithms supported by this printer.
For example: "deflate, gzip". Legal values include; "none", "deflate"
attributetypes: (public domain ZIP), "gzip" (GNU ZIP), "compress" (UNIX).'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{255} )

( 1.3.18.0.2.4.1127
NAME 'printer-pages-per-minute'
DESC 'The nominal number of pages per minute which may be output by this
printer (e.g., a simplex or black-and-white printer).
This attribute is informative, NOT a service guarantee.
Typically, it is the value used in marketing literature to describe this printer.'
EQUALITY integerMatch
ORDERING integerOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

( 1.3.18.0.2.4.1126 NAME 'printer-pages-per-minute-color'
DESC 'The nominal number of color pages per minute which may be output by this
printer (e.g., a simplex or color printer).
This attribute is informative, NOT a service guarantee.
Typically, it is the value used in marketing literature to describe this printer.'
EQUALITY integerMatch
ORDERING integerOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

( 1.3.18.0.2.4.1125 NAME 'printer-finishings-supported'
DESC 'The possible finishing operations supported by this printer.
Legal values include; "none", "staple", "punch", "cover", "bind", "saddle-stitch",
"edge-stitch", "staple-top-left", "staple-bottom-left", "staple-top-right",
"staple-bottom-right", "edge-stitch-left", "edge-stitch-top", "edge-stitch-right",
"edge-stitch-bottom", "staple-dual-left", "staple-dual-top", "staple-dual-right",
"staple-dual-bottom".'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{255} )

```

```

( 1.3.18.0.2.4.1124 NAME 'printer-number-up-supported'
DESC 'The possible numbers of print-stream pages to impose upon a single side of
an instance of a selected medium. Legal values include; 1, 2, and 4.
Implementations may support other values.'
EQUALITY integerMatch
ORDERING integerOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

( 1.3.18.0.2.4.1123 NAME 'printer-sides-supported'
DESC 'The number of impression sides (one or two) and the two-sided impression
rotations supported by this printer.
Legal values include; "one-sided", "two-sided-long-edge", "two-sided-short-edge".'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} )

( 1.3.18.0.2.4.1122 NAME 'printer-media-supported'
DESC 'The standard names/types/sizes (and optional color suffixes) of the media
supported by this printer.
For example: "iso-a4", "envelope", or "na-letter-white".
Legal values conform to ISO 10175, Document Printing Application (DPA), and any
IANA registered extensions.'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{255} )

( 1.3.18.0.2.4.1117 NAME 'printer-media-local-supported'
DESC 'Site-specific names of media supported by this printer, in the language in
"printer-natural-language-configured".
For example: "purchasing-form" (site-specific name) as opposed to
(in "printer-media-supported"): "na-letter" (standard keyword from ISO 10175).'
EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{255} )

( 1.3.18.0.2.4.1121 NAME 'printer-resolution-supported'
DESC 'List of resolutions supported for printing documents by this printer.
Each resolution value is a string with 3 fields:
1) Cross feed direction resolution (positive integer), 2) Feed direction
resolution (positive integer), 3) Resolution unit.
Legal values are "dpi" (dots per inch) and "dpcm" (dots per centimeter).
Each resolution field is delimited by ">". For example: "300> 300> dpi>".
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{255} )

( 1.3.18.0.2.4.1120 NAME 'printer-print-quality-supported'
DESC 'List of print qualities supported for printing documents on this printer.
For example: "draft, normal". Legal values include; "unknown", "draft", "normal",
"high".'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} )

( 1.3.18.0.2.4.1110 NAME 'printer-job-priority-supported'
DESC 'Indicates the number of job priority levels supported.
An IPP conformant printer which supports job priority must always support a
full range of priorities from "1" to "100"
(to ensure consistent behavior), therefore this attribute describes the
"granularity".

```

```

    Legal values of this attribute are from "1" to "100".'
EQUALITY integerMatch
ORDERING integerOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

( 1.3.18.0.2.4.1118
NAME 'printer-copies-supported'
DESC 'The maximum number of copies of a document that may be printed as a single job.
A value of "0" indicates no maximum limit.
A value of "-1" indicates unknown.'
EQUALITY integerMatch
ORDERING integerOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

( 1.3.18.0.2.4.1111
NAME 'printer-job-k-octets-supported'
DESC 'The maximum size in kilobytes (1,024 octets actually) incoming print job that
this printer will accept.
A value of "0" indicates no maximum limit. A value of "-1" indicates unknown.'
EQUALITY integerMatch
ORDERING integerOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )

( 1.3.18.0.2.4.1113
NAME 'printer-service-person'
DESC 'The name of the current human service person responsible for servicing this
printer.
It is suggested that this string include information that would enable other humans
to reach the service person, such as a phone number.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE )

( 1.3.18.0.2.4.1114
NAME 'printer-delivery-orientation-supported'
DESC 'The possible delivery orientations of pages as they are printed and ejected
from this printer.
Legal values include; "unknown", "face-up", and "face-down".'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} )

( 1.3.18.0.2.4.1115
NAME 'printer-stacking-order-supported'
DESC 'The possible stacking order of pages as they are printed and ejected from
this printer.
Legal values include; "unknown", "first-to-last", "last-to-first".'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} )

( 1.3.18.0.2.4.1116
NAME 'printer-output-features-supported'
DESC 'The possible output features supported by this printer.
Legal values include; "unknown", "bursting", "decollating", "page-collating",
"offset-stacking".'
EQUALITY caseIgnoreMatch

```

```

SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} )

( 1.3.18.0.2.4.1108
NAME 'printer-aliases'
DESC 'Site-specific administrative names of this printer in addition the printer
name specified for printer-name.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{127} )

( 1.3.6.1.4.1.42.2.27.5.1.63
NAME 'sun-printer-bsdaddr'
DESC 'Sets the server, print queue destination name and whether the client generates
protocol extensions.
"Solaris" specifies a Solaris print server extension. The value is represented b the
following value: server "," destination ", Solaris".'
SYNTAX '1.3.6.1.4.1.1466.115.121.1.15' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.64
NAME 'sun-printer-kvp'
DESC 'This attribute contains a set of key value pairs which may have meaning to the
print subsystem or may be user defined.
Each value is represented by the following: key "=" value.'
SYNTAX '1.3.6.1.4.1.1466.115.121.1.15' )

```

Internet Print Protocol (IPP) ObjectClasses

```

objectclasses: ( 1.3.18.0.2.6.2549
NAME 'slpService'
DESC 'DUMMY definition'
SUP 'top' MUST (objectclass) MAY ()

objectclasses: ( 1.3.18.0.2.6.254
NAME 'slpServicePrinter'
DESC 'Service Location Protocol (SLP) information.'
AUXILIARY SUP 'slpService')

objectclasses: ( 1.3.18.0.2.6.258
NAME 'printerAbstract'
DESC 'Printer related information.'
ABSTRACT SUP 'top' MAY ( printer-name
$ printer-natural-language-configured
$ printer-location
$ printer-info
$ printer-more-info
$ printer-make-and-model
$ printer-multiple-document-jobs-supported
$ printer-charset-configured
$ printer-charset-supported
$ printer-generated-natural-language-supported
$ printer-document-format-supported
$ printer-color-supported

```

```

$ printer-compression-supported
$ printer-pages-per-minute
$ printer-pages-per-minute-color
$ printer-finishings-supported
$ printer-number-up-supported
$ printer-sides-supported
$ printer-media-supported
$ printer-media-local-supported
$ printer-resolution-supported
$ printer-print-quality-supported
$ printer-job-priority-supported
$ printer-copies-supported
$ printer-job-k-octets-supported
$ printer-current-operator
$ printer-service-person
$ printer-delivery-orientation-supported
$ printer-stacking-order-supported $ printer! -output-features-supported ))

objectclasses: ( 1.3.18.0.2.6.255
NAME 'printerService'
DESC 'Printer information.'
STRUCTURAL SUP 'printerAbstract' MAY ( printer-uri
$ printer-xri-supported ))

objectclasses: ( 1.3.18.0.2.6.257
NAME 'printerServiceAuxClass'
DESC 'Printer information.'
AUXILIARY SUP 'printerAbstract' MAY ( printer-uri $ printer-xri-supported ))

objectclasses: ( 1.3.18.0.2.6.256
NAME 'printerIPP'
DESC 'Internet Printing Protocol (IPP) information.'
AUXILIARY SUP 'top' MAY ( printer-ipp-versions-supported $
printer-multiple-document-jobs-supported ))

objectclasses: ( 1.3.18.0.2.6.253
NAME 'printerLPR'
DESC 'LPR information.'
AUXILIARY SUP 'top' MUST ( printer-name ) MAY ( printer-aliases))

objectclasses: ( 1.3.6.1.4.1.42.2.27.5.2.14
NAME 'sunPrinter'
DESC 'Sun printer information'
SUP 'top' AUXILIARY MUST (objectclass $ printer-name) MAY
(sun-printer-bsdaddr $ sun-printer-kvp))

```

Sun Printer Attributes

```

ATTRIBUTE ( 1.3.6.1.4.1.42.2.27.5.1.63
NAME sun-printer-bsdaddr
DESC 'Sets the server, print queue destination name and whether the
client generates protocol extensions. "Solaris" specifies a
Solaris print server extension. The value is represented by

```

```

        the following value: server "," destination ", Solaris".'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
)

ATTRIBUTE ( 1.3.6.1.4.1.42.2.27.5.1.64
NAME sun-printer-kvp
DESC 'This attribute contains a set of key value pairs which may have
      meaning to the print subsystem or may be user defined. Each
      value is represented by the following: key "=" value.'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

```

Sun Printer Object Classes

```

OBJECTCLASS ( 1.3.6.1.4.1.42.2.27.5.2.14
NAME sunPrinter
DESC 'Sun printer information'
SUP top
AUXILIARY
MUST ( printer-name )
MAY ( sun-printer-bsdaddr $ sun-printer-kvp ))

```

Generic Directory Server Requirements

To support Solaris 9 LDAP clients, the server, regardless of what brand, must support the LDAP v3 protocol and compound naming and auxiliary object classes. In addition, at least one of the following controls must be supported.

- Simple paged-mode (RFC 2696)
- Virtual List View controls

The server must support at least one of the following authentication methods.

```

anonymous
simple
sasl/cram-MD5
sasl/digest-MD5

```

If using `pam_unix`, the server must support storing passwords in UNIX crypt format.

If using TLS, the server must support SSL or TLS.

Default Filters Used by LDAP Naming Services

If you do not manually specify a parameter for a given service using an SSD, the default filter is used. To list the default filters for a given service, use `ldaplist` with the `-v` option.

In the following example, `filter=(&(objectclass=iphost) (cn=abcde))` defines the default filters.

```
database=hosts
filter=(&(objectclass=iphost) (cn=abcde))
user data=(&(%s) (cn=abcde))
```

`ldaplist` generates the following list of default filters, where `%s` signifies a string and `%d`, a number.

```
hosts
(&(objectclass=iphost) (cn=%s))
-----
passwd
(&(objectclass=posixaccount) (uid=%s))
-----
services
(&(objectclass=ipservice) (cn=%s))
-----
group
(&(objectclass=posixgroup) (cn=%s))
-----
netgroup
(&(objectclass=nisnetgroup) (cn=%s))
-----
networks
(&(objectclass=ipnetwork) (ipnetworknumber=%s))
-----
netmasks
(&(objectclass=ipnetwork) (ipnetworknumber=%s))
-----
rpc
(&(objectclass=oncrpc) (cn=%s))
-----
protocols
(&(objectclass=ipprotocol) (cn=%s))
-----
bootparams
(&(objectclass=bootableDevice) (cn=%s))
-----
ethers
(&(objectclass=ieee802Device) (cn=%s))
-----
```

```

publickey
(&(objectclass=niskeyobject) (cn=%s))
or
(&(objectclass=niskeyobject) (uidnumber=%d))
-----
aliases
(&(objectclass=mailGroup) (cn=%s))
-----

```

TABLE 18-4 LDAP Filters Used in getXbyY Calls

Filter	Definition
bootparamByName	(&(objectClass=bootableDevice) (cn=%s))
etherByHost	(&(objectClass=ieee802Device) (cn=%s))
etherByEther	(&(objectClass=ieee802Device) (macAddress=%s))
groupByName	(&(objectClass=posixGroup) (cn=%s))
groupByGID	(&(objectClass=posixGroup) (gidNumber=%ld))
groupByMember	(&(objectClass=posixGroup) (memberUid=%s))
hostsByName	(&(objectClass=ipHost) (cn=%s))
hostsByAddr	(&(objectClass=ipHost) (ipHostNumber=%s))
keyByUID	(&(objectClass=nisKeyObject) (uidNumber=%s))
keyByHost	(&(objectClass=nisKeyObject) (cn=%s))
netByName	(&(objectClass=ipNetwork) (cn=%s))
netByAddr	(&(objectClass=ipNetwork) (ipNetworkNumber=%s))
nisgroupMember	(membernisnetgroup=%s)
maskByNet	(&(objectClass=ipNetwork) (ipNetworkNumber=%s))
printerByName	(&(objectClass=sunPrinter) (printer-name=%s))
projectByName	(&(objectClass=SolarisProject) (SolarisProjectName=%s))
projectByID	(&(objectClass=SolarisProject) (SolarisProjectID=%ld))
protoByName	(&(objectClass=ipProtocol) (cn=%s))
protoByNumber	(&(objectClass=ipProtocol) (ipProtocolNumber=%d))
passwordByName	(&(objectClass=posixAccount) (uid=%s))
passwordByNumber	(&(objectClass=posixAccount) (uidNumber=%ld))

TABLE 18-4 LDAP Filters Used in getXbyY Calls (Continued)

Filter	Definition
rpcByName	(&(objectClass=oncRpc) (cn=%s))
rpcByNumber	(&(objectClass=oncRpc) (oncRpcNumber=%d))
serverByName	(&(objectClass=ipService) (cn=%s))
serverByPort	(&(objectClass=ipService) (ipServicePort=%ld))
serverByNameAndProto	(&(objectClass=ipService) (cn=%s) (ipServiceProtocol=%s))
specialByNameserver	(ipServiceProtocol=%s))
ByPortAndProto	(&(objectClass=shadowAccount) (uid=%s))
netgroupByTriple	(&(objectClass=nisNetGroup) (nisnetgrouptriple= %s,%s,%s))
netgroupByMember	(&(objectClass=nisNetGroup) (membernisnetgroup=%s)
authName	(&(objectClass=SolarisAuthAttr) (cn=%s))
auditUserByName	(&(objectClass=SolarisAuditUser) (uid=%s))
execByName	(&(objectClass=SolarisExecAttr) (cn=%s) (SolarisKernelSecurityPolicy=%s) (SolarisProfileType=%s))
execByPolicy	(&(objectClass=SolarisExecAttr) (SolarisProfileId=%s) (SolarisKernelSecurityPolicy=%s) (SolarisProfileType=%s))
profileByName	(&(objectClass=SolarisProfAttr) (cn=%s))
userByName	(&(objectClass=SolarisUserAttr) (uid=%s))

The following table lists the getent attribute filters.

TABLE 18-5 getent Attribute Filters

Filter	Definition
aliases	(objectClass=rfc822MailGroup)
auth_attr	(objectClass=SolarisAuthAttr)
audit_user	(objectClass=SolarisAuditUser)
exec_attr	(objectClass=SolarisExecAttr)
group	(objectClass=posixGroup)

TABLE 18-5 getent Attribute Filters (Continued)

Filter	Definition
hosts	(objectClass=ipHost)
networks	(objectClass=ipNetwork)
prof_attr	(objectClass=SolarisProfAttr)
protocols	(objectClass=ipProtocol)
passwd	(objectClass=posixAccount)
printers	(objectClass=sunPrinter)
rpc	(objectClass=oncRpc)
services	(objectClass=ipService)
shadow	(objectclass=shadowAccount)
project	(objectClass=SolarisProject)
usr_attr	(objectClass=SolarisUserAttr)

Transitioning From NIS+ to LDAP

This chapter describes how to make the transition from using the NIS+ naming service to the LDAP naming service.

Overview

The NIS+ server daemon, `rpc.nisd`, stores NIS+ data in proprietary-format files in the `/var/nis/data` directory. While it is entirely possible to keep NIS+ data synchronized with LDAP, such synchronization has previously required an external agent. However, the NIS+ daemon now enables you to use an LDAP server as a data repository for NIS+ data. Since this makes it possible for NIS+ and LDAP clients to share the same naming service information, it is easier to transition from using NIS+ as the main naming service, to using LDAP for the same role.

By default, the `rpc.nisd` daemon continues to work as before, relying only on the `/var/nis/data` NIS+ database. If desired, the system administrator can choose to use an LDAP server as the authoritative data repository for any subset of the NIS+ database. In this case, the `/var/nis/data` files serve as a cache for the `rpc.nisd` daemon, reducing LDAP lookup traffic, and enabling the `rpc.nisd` to continue working if the LDAP server is temporarily unavailable. In addition to continuous synchronization between NIS+ and LDAP, you can also perform uploads of NIS+ data to LDAP, or downloads of LDAP data to NIS+.

Mapping of data to and from LDAP is controlled by a flexible configuration file syntax. (All standard NIS+ tables (except for `client_info.org_dir` and `timezone.org_dir`) are covered by a template mapping file, `/var/nis/NIS+LDAPmapping.template`), which should require little or no change for most NIS+ installations. (See “`client_info` and `timezone` Tables” on page 318 for information on `client_info.org_dir` and `timezone.org_dir`.) In addition to locations for NIS+ data in the LDAP Directory Information Tree (DIT), the mapping

file also allows establishing time-to-live (TTL) for NIS+ data sourced from LDAP. While there often is a one-to-one mapping between NIS+ column values and LDAP attribute values, the mapping file can be used to maintain more complicated relationships as well.

The new `/etc/default/rpc.nisd` file is used to select LDAP server and authentication, and controls some general `rpc.nisd` behavior. See `rpc.nisd(4)`. The details of the mapping is specified via the `/var/nis/NIS+LDAPmapping` file. For more information, see `NIS+LDAPmapping(4)`. The name of this file can be changed using the `-m` command-line option of `rpc.nisd`. For more information, see `rpc.nisd(1M)`.

The following terms are used in this chapter.

- **Container**
A container is the location in the LDAP DIT where all related entries are stored. For example, user account information is often stored in the `ou=People` container, while host address information can be stored in the `ou=Hosts` container.
- **Netname**
A netname is an entity in secure RPC (user or machine) that can be authenticated.
- **Mapping**
Mapping is the relationship between an NIS+ object and an LDAP entry. For example, data from the `name` column in the `passwd.org_dir` NIS+ table (such as the user name of an account) corresponds to the LDAP `uid` attribute of the `posixAccount` object class in the `ou=People` container. The configuration can establish a mapping between the `name` column and the `uid` attribute. You can also say that the `name` column is mapped to the `uid` attribute (or vice versa).
- **Principal**
A principal is an entity in NIS+ (user or machine) that can be authenticated. Usually, there is a one-to-one correspondence between netnames and principal names.

Configuration Files

Two configuration files control `rpc.nisd` operation.

- `/etc/default/rpc.nisd`
This file contains information regarding the LDAP server and authentication, the NIS+ base domain, the LDAP default search base, exception processing, and general `rpc.nisd` configuration, which applies whether or not LDAP mapping is in effect.
- `/var/nis/NIS+LDAPmapping`
This file contains information on mapping of NIS+ data to and from LDAP. The template file (`/var/nis/NIS+LDAPmapping.template`) covers all standard NIS+ objects, except `client_info.org_dir` and `timezone.org_dir`. See

“client_info and timezone Tables” on page 318 and NIS+LDAPmapping(4).

Configuration is done by assigning values to pre-defined attributes. In addition to the configuration files, the configuration attributes can also be read from LDAP (see “Storing Configuration Information in LDAP” on page 325) or can be specified on the `rpc.nisd` command line by way of the `-x` option. If the same attribute is specified in more than one place, the priority order is (from higher to lower) as follows.

1. `rpc.nisd -x` option
2. Configuration file
3. LDAP

Creating Attributes and Object Classes

Depending on how you configure the NIS+/LDAP mapping, you might need to create a number of new LDAP attributes and object classes. The examples show how to do this by specifying LDIF data that can be used as input to the `ldapadd` command. Create a file containing the LDIF data, and then invoke `ldapadd(1)`.

```
# ldapadd -D bind-DN-f ldif -file
```

This method works with Sun ONE™ Directory Server 5.1 , and might work with other LDAP servers as well.

Note – Except for the `defaultSearchBase`, `preferredServerList`, and `authenticationMethod` attributes, as well as the SYNTAX specifications, the object identifiers (OIDs) used in this chapter are intended for illustration only. As no official OIDs have been assigned, you are free to use any suitable OIDs.

Getting Started

For an introduction to the configuration needed to start using an LDAP repository for NIS+ data, see NIS+LDAPmapping(4). The remainder of this section goes into more detail about the organization of the configuration files.

```
/etc/default/rpc.nisd
```

All assignments in the `/etc/default/rpc.nisd` file are of the `attributeName=value` type.

General Configuration

The following attributes control general configuration of the `rpc.nisd`, and are active whether or not LDAP mapping is in effect. They should generally be left at their default values. See `rpc.nisd(4)` for more information.

- `nisplusNumberOfServiceThreads`
- `nisplusThreadCreationErrorAction`
- `nisplusThreadCreationErrorAttempts`
- `nisplusThreadCreationErrorTimeout`
- `nisplusDumpErrorAction`
- `nisplusDumpErrorAttempts`
- `nisplusDumpErrorTimeout`
- `nisplusResyncService`
- `nisplusUpdateBatching`
- `nisplusUpdateBatchingTimeout`

Configuration Data From LDAP

The following attributes control the reading of other configuration attributes from LDAP. These attributes cannot themselves reside in LDAP. They are read only from the command line or the configuration file. See `rpc.nisd(4)` for more information.

- `nisplusLDAPconfigDN`
- `nisplusLDAPconfigPreferredServerList`
- `nisplusLDAPconfigAuthenticationMethod`
- `nisplusLDAPconfigTLS`
- `nisplusLDAPconfigTLSCertificateDBPath`
- `nisplusLDAPconfigProxyUser`
- `nisplusLDAPconfigProxyPassword`

Server Selection

- `preferredServerList`

Specify the LDAP server and port number.

```
# LDAP server can be found at port 389
# LDAP server can be found at port 389
on the local machine
# preferredServerList=127.0.0.1
# Could also be written
# preferredServerList=127.0.0.0.1:389
LDAP server on the machine at IP
# address "1.2.3.4", at port 65042
# preferredServerList=1.2.3.4:65042
```

Authentication and Security

- `authenticationMethod`
- `nisplusLDAPproxyUser`
- `nisplusLDAPproxyPassword`

The authentication method and, if appropriate for the method selected, the proxy user (bind distinguished name [DN]) and password (key or other shared secret) to be used between the `rpc.nisd` daemon and the LDAP server. See “Security and Authentication” on page 308 for more information.

- `nisplusLDAPTLS`
- `nisplusLDAPTLSCertificateDBPath`

Optionally use SSL, and specify the location of the certificate file. See “Using SSL” on page 310 for more information.

Default Location in LDAP and NIS+

- `defaultSearchBase`

The point in the LDAP DIT where the containers for RFC 2307- style naming services data live. This is the default used when individual container DNs do not specify a full search base. See “`nisplusLDAPobjectDN`” on page 299 for more information.

- `nisplusLDAPbaseDomain`

The default NIS+ domain name to use when NIS+ object specifications (see “`nisplusLDAPdatabaseIdMapping`” on page 297) are not fully qualified.

Timeout/Size Limits and Referral Action for LDAP Communication

- `nisplusLDAPbindTimeout`
- `nisplusLDAPmodifyTimeout`
- `nisplusLDAPaddTimeout`
- `nisplusLDAPdeleteTimeout`

The above parameters are timeouts for the `ldap bind`, `modify`, `add`, and `delete` operations, respectively. They should generally be left at their default values.

- `nisplusLDAPsearchTimeout`
- `nisplusLDAPsearchTimeLimit`

The above parameters set the timeout for the LDAP search operation, and request a server-side search time limit, respectively. Since the `nisplusLDAPsearchTimeLimit` will control how much time the LDAP server spends on the search request, make sure that `nisplusLDAPsearchTimeLimit` is not

smaller than `nisplusLDAPsearchTimeout`. Depending on the performance of the NIS+ server, the LDAP server, and the connection between them, you might have to increase the search limits from the default values. Watch for timeout syslog messages from `rpc.nisd` as a clue to making these values larger.

- `nisplusLDAPsearchSizeLimit`

The above parameter requests a limit on the amount of LDAP data returned for an LDAP search request. The default is to ask for no limitation. This is a server side limit. The LDAP server might impose restrictions on the maximum, and these restrictions might be tied to the proxy user (bind DN) used. Make sure that the LDAP server allows the `rpc.nisd` to transfer enough data to account for the largest container (depending on the site, often the container used for `passwd.org_dir`, `mail_aliases.org_dir`, or `netgroup.org_dir`). Consult your LDAP server documentation for more information.

- `nisplusLDAPfollowReferral`

The above parameter defines the action to be taken when an LDAP operation results in a referral to another LDAP server. The default is to *not* follow referrals. Enable follow referrals if you want or need referrals to be honored. Keep in mind that while referrals are convenient, they can also slow down operations by making the `rpc.nisd` talk to multiple LDAP servers for each request. The `rpc.nisd` should generally be pointed directly to an LDAP server that can handle all LDAP requests that the `rpc.nisd` might make.

Error Actions

The following parameters define the actions to take when an error occurs during an LDAP operation. You should generally leave these at their defaults. See `rpc.nisd(4)` for more information.

- `nisplusLDAPinitialUpdateAction`
- `nisplusLDAPinitialUpdateOnly`
- `nisplusLDAPpretrieveErrorAction`
- `nisplusLDAPpretrieveErrorAttempts`
- `nisplusLDAPpretrieveErrorTimeout`
- `nisplusLDAPstoreErrorAction`
- `nisplusLDAPstoreErrorAttempts`
- `nisplusLDAPstoreErrorTimeout`
- `nisplusLDAPprefreshErrorAction`
- `nisplusLDAPprefreshErrorAttempts`
- `nisplusLDAPprefreshErrorTimeout`

General LDAP Operation Control

- `nisplusLDAPmatchFetchAction`

The above parameter determines whether or not LDAP data should be pre-fetched for NIS+ match operations. In most cases, leave this value at the default. See `rpc.nisd(4)` for more information.

`/var/nis/NIS+LDAPmapping`

The name of the above configuration can be changed via the `-m` option of `rpc.nisd(1M)`. The presence of the `NIS+LDAPmapping` file serves as a master switch for NIS+/LDAP mapping.

If you use a name other than the default for the mapping file, you will have to edit the `/etc/init.d/rpc` boot script to specify the mapping file name on the `rpc.nisd` startup line.

For each NIS+ object that should be mapped to and/or from LDAP, the `NIS+LDAPmapping` file specifies two to five attributes, depending on the object and whether or not the default values are sufficient.

`nisplusLDAPdatabaseIdMapping`

You must establish an alias to be used in the other mapping attributes. If the NIS+ object name is not fully qualified (does not end in a dot), the value of the `nisplusLDAPbaseDomain` is appended.

For example,

```
nisplusLDAPdatabaseIdMapping    rpc:rpc.org_dir
```

defines the database id `rpc` as an alias for the NIS+ `rpc.org_dir` table.

Note that NIS+ table objects might appear twice with two different database ids, once for the table object itself (if the object should be mapped to LDAP), and once for the table entries. For example,

```
nisplusLDAPdatabaseIdMapping    rpc_table:rpc.org_dir
nisplusLDAPdatabaseIdMapping    rpc:rpc.org_dir
```

defines the database ids `rpc_table` and `rpc` as aliases for the `rpc.org_dir` table. Later definitions will make it clear that `rpc_table` is used for the `rpc.org_dir` table object, and `rpc` for the entries in that table.

`nisplusLDAPentryTtl`

Since the `rpc.nisd` daemon's local database (in memory and on disk) functions as a cache for LDAP data, the `nisplusLDAPentryTtl` attribute allows you to set the time-to-live (TTL) values of entries in that cache. There are three TTLs for each database ID. The first two control the initial TTL when the `rpc.nisd` first loads the corresponding NIS+ object data from disk, and the third TTL is assigned to an object when it is read or refreshed from LDAP.

For example the following results in the `rpc.org_dir` table object getting an initial TTL randomly selected in the range 21600 to 43200 seconds.

```
nisplusLDAPentryTtl    rpc_table:21600:43200:43200
```

When that initial TTL expires and the table object is refreshed from LDAP, the TTL will be set to 43200 seconds.

Similarly the following will assign an initial TTL between 1800 and 3600 seconds to the entries in the `rpc.org_dir` table when it is first loaded.

```
nisplusLDAPentryTtl    rpc:1800:3600:3600
```

Each entry gets its own randomly selected TTL in the range specified. When a table entry expires and is refreshed, the TTL is set to 3600 seconds.

When selecting TTL values, consider the trade-off between performance and consistency. If the TTLs used for LDAP data cached by the `rpc.nisd` are very long, performance is the same as if `rpc.nisd` was not mapping data from LDAP at all. However, if the LDAP data is changed (by some entity other than `rpc.nisd`), it can also take a very long time before that change is visible in NIS+.

Conversely, selecting a very short (or even zero) TTL means that changes to LDAP data are quickly visible in NIS+, but can also impose a significant performance penalty. Typically, an NIS+ operation that also reads data from or writes data to LDAP will take at least two to three times longer (plus the LDAP lookup overhead) than the same operation without LDAP communication. Although performance can vary greatly depending on the hardware resources, scanning a large (tens of thousands or hundreds of thousands of entries) LDAP container to identify NIS+ entries that should be refreshed can take a long time. The `rpc.nisd` daemon performs this scan in the background, continuing to serve possibly stale data while it is running, but the background scan still consumes CPU and memory on the NIS+ server.

Carefully consider how critical it is to have NIS+ data in close synchronization with LDAP, and select the longest TTL that is acceptable for each NIS+ object. The default (when no `nisplusLDAPentryTtl` is specified) is 1 hour. The template mapping file `/var/nis/NIS+LDAPmapping.template` changes this to 12 hours for objects other than table entries. However, there is no auto-recognition of non-entry objects, so if you add mapping for a non-entry object, the TTL will default to 1 hour.

Note – There are no TTLs for nonexistent objects. Hence, no matter which TTLs are in effect for LDAP-mapped entries in an NIS+ table, a request for an entry that does not exist in NIS+ will query LDAP for that entry.

nisplusLDAPobjectDN

For each mapped NIS+ object, `nisplusLDAPobjectDN` establishes the location in the LDAP DIT where the object data resides. It also allows specification of the action to take when an LDAP entry is deleted. Each `nisplusLDAPobjectDN` value has three parts. The first specifies where LDAP data is read from, the second to where it is written, and the third what should happen when LDAP data is deleted. Refer to the following example.

```
nisplusLDAPobjectDN   rpc_table:\
                        cn=rpc,ou=nisPlus,?base?\
                        objectClass=nisplusObjectContainer:\
                        cn=rpc,ou=nisPlus,?base?\
                        objectClass=nisplusObjectContainer,\
                        objectClass=top
```

The above example shows that the `rpc.org_dir` table object should be read from the DN `cn=rpc,ou=nisPlus`, (since the value ends in a comma, the value of the `defaultSearchBase` attribute is appended), with scope `base`, and that entries with a value of `nisplusObjectContainer` for the `ObjectClass` attribute are selected.

The table object is written to the same place. The delete specification is missing, which implies the default action, which is as follows. If the NIS+ table object is deleted, the entire LDAP entry should also be deleted.

If data should be read from, but not written to LDAP, omit the write portion (and the colon separating it from the read part).

```
nisplusLDAPobjectDN   rpc_table:\
                        cn=rpc,ou=nisPlus,?base?\
                        objectClass=nisplusObjectContainer
```

Note that the `nisplusObjectContainer` object class is not part of RFC 2307. In order to use it, you must configure your LDAP server as detailed in “Mapping NIS+ Objects Other Than Table Entries” on page 311.

For the `rpc.org_dir` table entries, you could use the following example.

```
nisplusLDAPobjectDN   rpc:ou=Rpc,?one?objectClass=oncRpc:\
                        ou=Rpc,?one?objectClass=oncRpc,objectClass=top
```

The above shows that the table entries are read from and written to the base `ou=Rpc`. Again, the trailing comma appends the `defaultSearchBase` value. Select entries that have an `objectClass` attribute value of `oncRpc`. When creating an entry in the `ou=Rpc` container in LDAP, you also must specify `top` as an `objectClass` value.

As an example showing a non-default delete specification, consider the following.

```
nisplusLDAPobjectDN    user_attr:\
                        ou=People,?one?objectClass=SolarisUserAttr,\
                        solarisAttrKeyValue=*\
                        ou=People,?one?objectClass=SolarisUserAttr:\
                        dbid=user_attr_del
```

The `user_attr.org_dir` data resides in the `ou=People` LDAP container, which it shares with account information from other sources, such as the `passwd.org_dir` NIS+ table.

Select entries in that container that have the `solarisAttrKeyValue` attribute, since only those contain `user_attr.org_dir` data. The `dbid=user_attr_del` portion of the `nisplusLDAPobjectDN` shows that when an entry in the `user_attr.org_dir` NIS+ table entry is deleted, deletion of the corresponding LDAP entry (if any) should follow the rules in the rule set identified by the `user_attr_del` database ID. See “`nisplusLDAPcolumnFromAttribute`” on page 300 for more information.

`nisplusLDAPattributeFromColumn`

`nisplusLDAPattributeFromColumn` specifies the rules used to map NIS+ data to LDAP. Mapping rules for the other direction is controlled by `nisplusLDAPcolumnFromAttribute`.

`nisplusLDAPcolumnFromAttribute`

`nisplusLDAPcolumnFromAttribute` specifies the rules used to map LDAP data to NIS+.

The full entry mapping syntax can be found on `NIS+LDAPmapping(4)`. However, a few examples should make things clearer.

The NIS+ `rpc.org_dir` table contains four columns called `cname`, `name`, `numbe`, and `comment`. Therefore, the entries for the NIS+ RPC program number (100300) with the canonical name `nisd` and the aliases `rpc.nisd` and `nisplusd` could be represented by the following NIS+ entries in `rpc.org_dir`.

```
nisd nisd 100300    NIS+ server
nisd rpc.nisd 100300    NIS+ server
nisd nisplusd 100300    NIS+ server
```

Assuming the default `searchBase` value is `dc=some,dc=domain`, the corresponding LDAP entry, as listed by `ldapsearch(1)`, would be the following.

```
cn=nisd,ou=Ppc,dc=some,dc=domain
cn=nisd
cn=rpc.nsid
```

```

cn=nisplud
oncrocnnumber=100300
description=NIS+ server
objectclass=oncRpc
objectclass=top

```

This makes for a simple one-to-one mapping between NIS+ and LDAP data, and the corresponding mapping attribute value going from NIS+ to LDAP is the following.

```

nisplusLDAPAttributeFromColumn \
rpc:      dn=("cn=%s,", name), \
          cn=cname, \
          cn=name, \
          oncRpcNumber=number, \
          description=comment

```

This constructs the DN for the entry to be `cn=%s,` with the value of the `cname` column substituted for `%s`.

```
cn=nisd,
```

Since the value ends in a comma, the read base value from the `nisplusObjectDN` is appended, and you have the following.

```
cn=nisd,ou=Rpc,dc=some,dc=domain
```

The `oncRpcNumber` and `description` attribute values are just simple assignments of the corresponding NIS+ column values. The `rpc.nisd` will collect the multiple NIS+ entries into one LDAP entry, with multiple `cn` values to represent the different name column values.

Similarly, the mapping from LDAP to NIS+ would be as follows.

```

nisplusLDAPColumnFromAttribute \
rpc:      cname=cn, \
          (name)=(cn), \
          number=oncRpcNumber, \
          comment=description

```

The above assigns the `oncRpcNumber` and `description` values to the corresponding NIS+ columns. The multi-valued `cn` (denoted by `(cn)`) is mapped to multiple name column values (denoted by `(name)`). Since the name column cannot be multi-valued, the `rpc.nisd` creates one NIS+ entry for each `cn` value.

Finally, the `nisplusLDAPAttributeFromColumn` value is an example of rule sets used for deletion.

```

nisplusLDAPAttributeFromColumn \
user_attr_del:  dn=("uid=%s,", name), \
               SolarisUserQualifier=, \
               SolarisAttrReserved1=, \
               SolarisAttrReserved2=, \
               SolarisAttrKeyValue=

```

Again, the `user_attr.org_dir` data shares the `ou=People` container with other account information (from the `passwd.org_dir` and other tables). If an entry in the `user_attr.org_dir` table is deleted, you probably do not want to delete the entire `ou=People` entry. Instead, the delete entry above says that when a `user_attr.org_dir` entry is deleted, the `SolarisUserQualifier`, `SolarisAttrReserved1`, `SolarisAttrReserved2`, and `SolarisAttrKeyValue` attributes (if any) are deleted from the `ou=People` entry specified by the following rule.

```
dn=("uid=%s,", name)
```

The rest of the LDAP entry is left unchanged.

NIS+ to LDAP Migration Scenarios

Likely scenarios for a migration from NIS+ to LDAP include the following.

- *Convert all NIS+ clients to LDAP in one operation.* You can use the `rpc.nisd` daemon to upload any NIS+ data that does not yet exist in LDAP. See “How to Convert All NIS+ Data to LDAP in One Operation” on page 302.
- *Do a gradual migration from NIS+ to LDAP.* Start by converting NIS+ data to LDAP (see “How to Convert All NIS+ Data to LDAP in One Operation” on page 302). You could have both NIS+ and LDAP clients sharing the same naming service data, and let the `rpc.nisd` automatically keep NIS+ and LDAP data synchronized. Initially, perhaps, NIS+ would be authoritative, and the LDAP server(s) would maintain a duplicate of the NIS+ data for the benefit of LDAP clients. At a convenient time, LDAP can become the authoritative naming service, and NIS+ service gradually phased out, until there are no more NIS+ clients.
- LDAP is already used as a naming service, so you need to merge the NIS+ and LDAP data. There are three possible ways to perform this merge.
 - *Add the NIS+ data to LDAP.* Entries that exist in NIS+, but not in LDAP, are added to LDAP. Entries that appear both in NIS+ and LDAP, but with different data, end up with the NIS+ data. See “How to Convert All NIS+ Data to LDAP in One Operation” on page 302.
 - *Overwrite the NIS+ data with the LDAP data.* If there are entries that exist in NIS+ but not in LDAP, they will disappear from NIS+. Entries that exist both in NIS+ and LDAP end up with the LDAP data. See “How to Convert All LDAP Data to NIS+ in One Operation” on page 303.
 - *Merge NIS+ and LDAP data, resolving conflicts on an individual basis.* See “Merging NIS+ and LDAP Data” on page 303.

▼ How to Convert All NIS+ Data to LDAP in One Operation

- **Use the `rpc.nisd` to upload any NIS+ data that does not yet exist in LDAP.**

Assuming all NIS+/LDAP data mappings have been established in the default location (`/var/nis/NIS+LDAPmapping`), use the following command.

```
# /usr/sbin/rpc.nisd -D \  
-x nisplusLDAPinitialUpdateAction=to_ldap \  
-x nisplusLDAPinitialUpdateOnly=yes
```

The above would make the `rpc.nisd` upload data to LDAP, and then exit. The NIS+ data would be unaffected by this operation.

See the `nisplusLDAPinitialUpdateAction` attribute on `rpc.nisd(4)`.

▼ How to Convert All LDAP Data to NIS+ in One Operation

- Use the `rpc.nisd` to download all LDAP data to NIS+, overwriting existing NIS+ data.

Assuming all NIS+/LDAP data mappings have been established in the default location (`/var/nis/NIS+LDAPmapping`), use the following command.

```
# /usr/sbin/rpc.nisd -D \  
-x nisplusLDAPinitialUpdateAction=from_ldap \  
-x nisplusLDAPinitialUpdateOnly=yes
```

The above would make the `rpc.nisd` daemon download data from LDAP, and then exit. The LDAP data would be unaffected by this operation.

See the `nisplusLDAPinitialUpdateAction` attribute on `rpc.nisd(4)`.

Merging NIS+ and LDAP Data

“NIS+ to LDAP Migration Scenarios” on page 302 showed how to synchronize NIS+ and LDAP data when data conflicts between the two should be resolved by letting either the NIS+ or the LDAP data be authoritative. Merging data requires a more complicated procedure.

The example procedure in this section assumes the following.

- You are putting a backup of the NIS+ data in the `/nisbackup` directory.
- Valid mapping configuration already exists in `/etc/default/rpc.nisd` and `/var/nis/tmpmap` (for tables that should be merged).
- Flat file representations of the NIS+ data before the merge are stored in `/before`, and after-merge representations in `/after`.
- `niscat` is used to dump flat file representations of custom NIS+ tables not supported by `nisaddent(1M)`. You might have your own commands or scripts for dumping and loading such custom tables from and to NIS+. If so, those commands/scripts should be used in preference to `niscat` since the latter has no convenient counterpart to load data back into NIS+.

If you are forced to dump data using `niscat(1)`, you can use `nistbladm(1)` to load entries back into NIS+ one by one.

- Your command path includes `/usr/lib/nis` (which is where `nisaddent(1M)` resides).

▼ How to merge NIS+ and LDAP data



Caution – If the LDAP data should change between the download in Step 4 and the upload in Step 10, the upload might overwrite those changes. For this reason, you should try to prevent modifications to the LDAP data during this procedure. Consult your LDAP server documentation for more information.

1. **Back up all NIS+ data using the `nisbackup` command.**

```
# nisbackup -a /nisbackup
```

2. **Identify the NIS+ tables that have data which must be merged with LDAP. Dump the contents of these tables to flat files. For example, dump the contents of `group.org_dir` using `nisaddent` as follows.**

```
# nisaddent -d group | sort > /before/group
```

Piping the `nisaddent` output to `sort` will make for convenient comparison later on.

3. **Stop the `rpc.nisd` daemon.**

```
# pkill rpc.nisd
```

4. **Download LDAP data to NIS+.**

```
# /usr/sbin/rpc.nisd -D -m tmpmap \  
-x nisplusLDAPinitialUpdateAction=from_ldap \  
-x nisplusLDAPinitialUpdateOnly=yes
```

5. **Start the `rpc.nisd` daemon.**

```
# /usr/sbin/rpc.nisd
```

The `rpc.nisd` daemon will now be serving the data downloaded from LDAP. If the conflicts to be resolved are such that NIS+ clients should not be exposed to them, make sure to perform this and the following steps when there are few (preferably no) active NIS+ clients.

6. **Dump the NIS+ data for the affected tables.**

The following example uses the `group.org_dir` table.

```
# nisaddent -d group | sort > /after/group
```

7. **Use your favorite file merge procedure to produce merged versions of the tables. If no other tools are available, you can use `diff(1)` to collect differences between the `/before` and `/after` files, and merge manually with a text editor.**

The following example assumes that the merged results are available in `/after`.

8. Load the merged data into NIS+. The following example uses the group table.

```
# nisaddent -m -f /after/group group
```

9. Remove LDAP entries that should not exist after the merge.

A. If there are LDAP entries that do not exist in the (now merged) NIS+ data, and that should not exist in LDAP after the upload, you must remove those LDAP entries.

Your LDAP server might provide a convenient method for removing multiple entries, such as a way to delete all entries in a container. If this is not the case, you can use `ldapsearch(1)` to generate a list of entries for each container. For example, to generate a list of all entries in the `ou=Rpc` container, use `ldapsearch(1)` as follows.

```
# ldapsearch -h server-address -D bind-DN -w password \  
-b ou=Rpc,search-base 'objectClass=*' dn | \  
grep -i ou=Rpc | grep -v -i ^ou=Rpc > \  
/tmp/delete-dn
```

See “Performance and Indexing” on page 310 for an explanation of the meta-arguments (*server-address*, *bind-DN*, for example).

B. You can now edit the result file (`/tmp/delete-dn`) to specify only those entries that should be removed. Alternatively, in order to remove all entries in the container, use the file as is, and rely on the NIS+ upload to restore the LDAP data. Either way, you should backup the LDAP data before performing the `ldapdelete` operation below.

C. Use `ldapdelete` to remove LDAP entries, redirecting `stdout` (which usually is one blank line for each entry removed) to `/dev/null`.

```
# ldapdelete -h server-address -D bind-DN -w password \  
/tmp/delete-dn /dev/null
```

D. Repeat the above procedure for each container that has at least one entry which must be removed.

10. NIS+ now contains the merged data, which can be uploaded to LDAP. Do the following.

Stop the `rpc.nisd` daemon.

```
# pkill rpc.nisd
```

Perform the upload.

```
# /usr/sbin/rpc.nisd -D -m tmpmap \  
-x nisplusLDAPinitialUpdateAction=to_ldap \  
-x nisplusLDAPinitialUpdateOnly=yes
```

11. Restart the `rpc.nisd` daemon.

- If the `rpc.nisd` daemon uses the LDAP repository, specify an appropriate mapping file.
 - If the `rpc.nisd` daemon provides NIS (YP) emulation, specify the `-Y` option.
- ```
/usr/sbin/rpc.nisd -m mappingfile [-Y]
```

Alternatively, omit `-x nisplusLDAPinitialUpdateOnly=yes` from the upload command in Step 10. This will make the `rpc.nisd` daemon start serving NIS+ data when the upload is done.

---

## Masters and Replicas

Only NIS+ masters are allowed to write data to LDAP. NIS+ replicas can obtain updates either from the NIS+ master (which might or might not have obtained it from LDAP), or they can read data directly from an LDAP server. A combination of the two is also possible. Therefore, there are two principal ways to arrange for NIS+ replication.

- *Leave NIS+ replicas unchanged, and let them obtain their data updates from the NIS+ master.*

This arrangement has the advantage of configurational simplicity (only the NIS+ master need have a connection to an LDAP server), and also maintains the old replication relationship (master knows about new data first, replicas later). It is probably the most convenient solution while NIS+ remains authoritative for naming service data. However, it also lengthens the path between LDAP and NIS+ replica servers.

- *Let NIS+ replicas obtain their data directly from LDAP instead of from the NIS+ master.*

In this case, replicas could have updated data before or after the NIS+ master, depending on lookup traffic and TTLs for data derived from LDAP. This arrangement is more complicated, but can be convenient when LDAP is the authoritative naming services repository, and few or no updates are made directly to NIS+ data.

## Replication Timestamps

When an NIS+ replica is obtaining data for at least one object in a particular NIS+ directory from LDAP, the update timestamps printed by `nisping(1M)` do not necessarily indicate the degree of data consistency between the NIS+ master and the replica. For example, assume that the NIS+ directory `dir1` contains the tables `table1` and `table2`. When the replica is obtaining data for both `table1` and `table2` from the NIS+ master, you might see an output like the following.

```
nisping dir1
```

```
Master server is "master.some.domain."
Last update occurred at Mon Aug 5 22:11:09 2002

Replica server is "replica.some.domain."
```

Last Update seen was Mon Aug 5 22:11:09 2002

The above indicates that the master and replica have exactly the same data. However, if the replica is getting data for either or both of `table1` and `table2` from LDAP, the output only shows that the replica has received an `NIS_PING` from the master, and updated its resynchronization time stamp for housekeeping purposes. The data in the table or tables mapped from LDAP might differ from that on the NIS+ master if either of the following are true.

- The LDAP data differs from that on the NIS+ master.
- The replica has data in its cache (its local version of the NIS+ database) that has not expired, but that is not up to date with LDAP.

If you cannot accept this type of data inconsistency, let all NIS+ replicas obtain their data from the NIS+ master only. Once you have configured the NIS+ master to get data from LDAP, you do not need to make modifications to the replicas.

---

## The Directory Server

The LDAP mapping portion of the `rpc.nisd` daemon uses LDAP protocol version 3 to talk to the LDAP server. The default mapping configuration (`/var/nis/NIS+LDAPmapping.template`) expects that the LDAP server supports an extended version of RFC 2307. RFCs can be retrieved from <http://www.ietf.org/rfc.html>. While the mapping between NIS+ and LDAP data can be modified using `NIS+LDAPmapping(4)`, there is a basic assumption that the LDAP data is organized along the principles laid out in RFC 2307.

For example, in order to share account information between direct LDAP clients and NIS+ clients, the LDAP server must support storing account (user) passwords in the UNIX `crypt` format. If the LDAP server cannot be configured to do so, you can still store NIS+ data, including accounts, in LDAP. However, you will not be able to fully share account information between NIS+ users and LDAP `bindDNs`.

### Configuring the Sun ONE Directory Server 5.1

Refer to the Sun ONE™ Directory Server 5.1 Collection for detailed instructions on the installation, setup and administration of Sun ONE Directory Server 5.1.

You can use `idsconfig(1M)` to configure Sun ONE Directory Server version 5.1 for LDAP clients using LDAP as a naming service. The setup provided by `idsconfig(1M)` is also appropriate when using NIS+ with an LDAP data repository.

---

**Note** – If you are using an LDAP server other than Sun ONE Directory Server 5.1, you must manually configure the server to support the RFC 2307 schemas.

---

## Assigning Server Address and Port Number

The `/etc/default/rpc.nisd` file is set up to use a local LDAP server at port 389. If this is not correct in your configuration, establish a new value for the `preferredServerList` attribute. For example, to use an LDAP server at IP address `192.0.0.1` and port `65535`, you specify the following.

```
preferredServerList=192.0.0.1:65535
```

## Security and Authentication

Authentication between NIS+ clients and the NIS+ server is not affected when the NIS+ server is obtaining data from LDAP. However, in order to maintain the integrity of the NIS+ data when it is stored in LDAP, consider configuring authentication between the `rpc.nisd` daemon and the LDAP server. Several different types of authentication are available, depending on the capabilities of the LDAP server.

The LDAP authentication supported by the `rpc.nisd` daemon includes the following.

- `none`

The `none` authentication method is the default. While using `none` requires no setup, it also provides no security. It is only suitable for use in environments that have no security requirements at all.

To use the `none` authentication, make sure that the `authenticationMethod` attribute has the following value.

```
authenticationMethod=none
```

The authentication methods that actually provide at least some security typically require that you associate a shared secret (a password or key) with a DN in LDAP. The DN you select for use by the `rpc.nisd` daemon can be unique, or can also be used for other purposes. It should have appropriate capabilities to support the expected LDAP traffic. For example, if the `rpc.nisd` daemon should be able to write data to LDAP, the selected DN must have the right to add/update/delete LDAP data in the containers used for the NIS+ data. Also, the LDAP server might, by default, impose limitations on resource usage (such as search time limits or search result size limitations). If this is the case, the selected DN must have sufficient capabilities to support enumeration of the NIS+ data containers.

- `simple`

The simple authentication method provides authentication by unencrypted exchange of a password string. Since the password is sent in the clear between the LDAP client (the `rpc.nisd` daemon) and LDAP server, the simple method is suitable only when information exchange between the NIS+ and LDAP servers is protected by some other method.

For instance, transport layer encryption of LDAP traffic, or the special case where the NIS+ and LDAP server is one and the same system, and the NIS+/LDAP traffic stays in the kernel, protected from the eyes of unauthorized users.

Modify the configuration of the `rpc.nisd` daemon with the DN and password to use for the simple authentication. For example, if the DN is `cn=nisplusAdmin,ou=People,dc=some,dc=domain`, and the password `aword`, establish the following.

```
authenticationMethod=simple
nisplusLDAPproxyUser=cn=nisplusAdmin,ou=People,dc=some,dc=domain
nisplusLDAPproxyPassword=aword
```

Be sure to protect the place where the password is stored from unauthorized access. Remember that if the password is specified on the `rpc.nisd` command line, it might be visible to any user on the system via commands such as `ps(1)`.

- `sasl/digest-md5`

The `sasl/digest-md5` authentication method provides authentication using the `digest/md5` algorithm.

Consult your LDAP server documentation for information on how to set up an authorization identity for use with `digest-md5`, and modify the `/etc/default/rpc.nisd` file to specify this identity and its associated password.

```
authenticationMethod=sasl/digest-md5
nisplusLDAPproxyUser=cn=nisplusAdmin,ou=People,dc=some,dc=domain
nisplusLDAPproxyPassword=aword
```

Be sure to protect the file where the password is stored from unauthorized access.

- `sasl/cram-md5`

Authentication using the `cram/md5` algorithm. Probably only supported by the obsolete SunDS LDAP server.

Consult your LDAP server documentation for information on how to set up a bind DN for use with `cram-md5`, and modify the `/etc/default/rpc.nisd` file to specify this DN and its associated password.

```
authenticationMethod=sasl/cram-md5
nisplusLDAPproxyUser=cn=nisplusAdmin,ou=People,dc=some,dc=domain
nisplusLDAPproxyPassword=aword
```

Be sure to protect the file where the password is stored from unauthorized access.

## Using SSL

The `rpc.nisd` daemon also supports transport layer encryption of LDAP traffic using SSL. Consult your LDAP server documentation to generate an SSL certificate for LDAP server authentication. Store the certificate in a file on the NIS+ server (`/var/nis/cert7.db`, for example) and modify `/etc/default/rpc.nisd` as follows.

```
nisplusLDAPTLS=ssl
nisplusLDAPTLSCertificateDBPath=/var/nis/cert7.db
```

Be sure to protect the certificate file from unauthorized access. Note that the above provides session encryption and authentication of the LDAP server to the `rpc.nisd`. It does not provide authentication of the `rpc.nisd` to the LDAP server, since the certificate does not contain anything that identifies the LDAP client (`rpc.nisd`). However, you can combine SSL with another authentication method (`simple`, `sasl/digest-md5`) in order to achieve mutual authentication.

## Performance and Indexing

When the `rpc.nisd` daemon is asked to enumerate an NIS+ table (using `niscat(1)` for example) that is mapped from LDAP, it will enumerate the corresponding LDAP container if at least one entry in the table has an expired TTL. Although this container enumeration is done in the background, so that LDAP performance is of limited importance, it can nevertheless be beneficial to establish LDAP indices to speed up container enumeration for large containers.

To obtain an estimate of the amount of time required for enumeration of a particular container, you can use a command like the following.

```
% /bin/time ldapsearch -h server-address -D bind-DN -w password \
-b container, search-base 'cn=*' /dev/null
```

where

- *server-address*  
IP address portion of `preferredServerList` value from `/etc/default/rpc.nisd`
- *bind-DN*  
`nisplusLDAPproxyUser` value from `/etc/default/rpc.nisd`
- *password*  
`nisplusLDAPproxyPassword` value from `/etc/default/rpc.nisd`
- *container*  
One of the RFC 2307 container names (`ou=Services`, `ou=Rpc`, and so on.)
- *search-base*

defaultSearchBase value from `/etc/default/rpc.nisd`

The "real" value printed by `/bin/time` is the elapsed (wall-clock) time. If this value exceeds a significant fraction (25 percent or more) of the TTL for the corresponding table entries (see "Authentication and Security" on page 295), it might be beneficial to index the LDAP container.

The `rpc.nisd` supports the `simple` page and VLV indexing methods. Refer to your LDAP server documentation to find out which indexing methods it supports, and how to create such indices.

---

## Mapping NIS+ Objects Other Than Table Entries

You can store NIS+ objects other than table entries in LDAP. However, doing so has no particular value unless you also have NIS+ replicas that obtain those NIS+ objects from LDAP. The recommended choices are the following.

- *There are no replicas, or the replicas obtain their data from the NIS+ master only.*

Edit the mapping configuration file (see `NIS+LDAPmapping(4)`) to remove the following attribute values for all non-table-entry objects.

```
nisplusLDAPdatabaseIdMapping
nisplusLDAPentryTtl
nisplusLDAPobjectDN
```

For example, if you started out from the `/var/nis/NIS+LDAPmapping.template` file, the sections you need to remove (or disable by commenting) are as follows.

```
Standard NIS+ directories
nisplusLDAPdatabaseIdMapping basedir:
.
.
.

nisplusLDAPdatabaseIdMapping user_attr_table:user_attr.org_dir
nisplusLDAPdatabaseIdMapping audit_user_table:audit_user.org_dir

Standard NIS+ directories
nisplusLDAPentryTtl basedir:21600:43200:43200
.
.
.

nisplusLDAPentryTtl user_attr_table:21600:43200:43200
nisplusLDAPentryTtl audit_user_table:21600:43200:43200
```

```

Standard NIS+ directories
nisplusLDAPobjectDN basedir:cn=basedir,ou=nisPlus,?base?\
 objectClass=nisplusObjectContainer:\
 cn=basedir,ou=nisPlus,?base?\
 objectClass=nisplusObjectContainer,\
 objectClass=top
.
.
.
nisplusLDAPobjectDN audit_user_table:cn=audit_user,ou=nisPlus,?base?\
 objectClass=nisplusObjectContainer:\
 cn=audit_user,ou=nisPlus,?base?\
 objectClass=nisplusObjectContainer,\
 objectClass=top

```

- *NIS+ replicas obtain their data from LDAP server.*

Create the `nisplusObject` attribute and `nisplusObjectContainer` object class as shown in the following example (LDIF data is suitable for `ldapadd(1)`. Attribute and object class OIDs are for illustration only.)

```

dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.1.0 NAME 'nisplusObject' \
 DESC 'An opaque representation of an NIS+ object' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 SINGLE-VALUE)

dn: cn=schema
changetype: modify
add: objectclasses

objectclasses(1.3.6.1.4.1.42.2.27.5.42.42.2.0 NAME'nisplusObjectContainer'\
SUP top STRUCTURAL DESC 'Abstraction of an NIS+ object
MUST (cn $ nisplusObject))

```

You also need to create a container for the NIS+ objects. The following LDIF syntax shows how to create the `ou=nisPlus,dc=some,dc=domain` container, and can be used as input to `ldapadd(1)`.

```

dn: ou=nisPlus,dc=some,dc=domain
ou: nisPlus
objectClass: top
objectClass: organizationalUnit

```

---

## NIS+ Entry Owner, Group, Access, and TTL

When NIS+ table entries are created from LDAP data, the default behavior is to initialize the entry object owner, group, access rights, and TTL using the corresponding values from the table object in which the entry object lives. This is normally sufficient, but there might be cases where these NIS+ entry attributes must be established individually. An example of this would be a site that did not use the `rpc.nispasswd(1M)` daemon. In order to allow individual users to change their NIS+ passwords (and re-encrypt their Diffie-Hellman keys stored in the `cred.org_dir` table), `passwd.org_dir` and `cred.org_dir` entries for the user should be owned by the user, and have modify rights for the entry owner.

If you need to store table entry owner, group, access, or TTL in LDAP for one or more NIS+ tables, you need to do the following.

### ▼ How to Store Additional Entry Attributes in LDAP

1. Consult your LDAP server documentation, and create the following new attributes and object class. (LDIF data is suitable for `ldapadd`. Attribute and object class OIDs are for illustration only.)

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.4.0 NAME 'nisplusEntryOwner' \
DESC 'Opaque representation of NIS+ entry owner' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.4.1 NAME 'nisplusEntryGroup' \
DESC 'Opaque representation of NIS+ entry group' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.4.2 NAME 'nisplusEntryAccess' \
DESC 'Opaque representation of NIS+ entry access' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.4.3 NAME 'nisplusEntryTtl' \
DESC 'Opaque representation of NIS+ entry TTL' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)

dn: cn=schema
changetype: modify
add: objectclasses

objectclasses:(1.3.6.1.4.1.42.2.27.5.42.42.5.0 NAME 'nisplusEntryData'\
SUP top STRUCTURAL DESC 'NIS+ entry object non-column data'\

MUST (cn) MAY (nisplusEntryOwner $ nisplusEntryGroup $ \
nisplusEntryAccess $ nisplusEntryTtl))
```

**2. Modify the `nisplusLDAPobjectDN` attribute value for the relevant table(s) so that the write portion includes the newly created `nisplusEntryData` object class.**

For example, for the `passwd.org_dir` table, assuming that you are using a mapping file based on `/var/nis/NIS+LDAPmapping.template`, edit as follows.

```
nisplusLDAPobjectDN passwd:ou=People,?one?objectClass=shadowAccount,\
 objectClass=posixAccount:\
 ou=People,?one?objectClass=shadowAccount,\
 objectClass=posixAccount,\
 objectClass=account,objectClass=top
```

Edit the attribute value as follows.

```
nisplusLDAPobjectDN passwd:ou=People,?one?objectClass=shadowAccount,\
 objectClass=posixAccount:\
 ou=People,?one?objectClass=shadowAccount,\
 objectClass=posixAccount,\
 objectClass=nisplusEntryData,\
 objectClass=account,objectClass=top
```

**3. Edit the `nisplusLDAPattributeFromColumn` and `nisplusLDAPcolumnFromAttribute` attribute values to specify any desired subset of owner, group, access, or TTL.**

In Step 2, you created the LDAP attributes used to store these values. For NIS+, there are predefined pseudo-column names called `zo_owner`, `zo_group`, `zo_access`, and `zo_ttl`, respectively. For example, in order to store owner, group, and access for `passwd.org_dir` entries in LDAP, modify the `nisplusLDAPattributeFromColumn` value from the following.

```
nisplusLDAPattributeFromColumn \
passwd: dn=("uid=%s,", name), \
 cn=name, \
 uid=name, \
 userPassword="{crypt$}%s", passwd), \
 uidNumber=uid, \
 gidNumber=gid, \
 gecos=gecos, \
 homeDirectory=home, \
 loginShell=shell, \
 (shadowLastChange,shadowMin,shadowMax, \
 shadowWarning, shadowInactive,shadowExpire)=\
 (shadow, ":")
```

Edit to read as follows.

```
nisplusLDAPattributeFromColumn \
passwd: dn=("uid=%s,", name), \
 cn=name, \
 uid=name, \
 userPassword="{crypt$}%s", passwd), \
 uidNumber=uid, \
 gidNumber=gid, \
 gecos=gecos, \
 homeDirectory=home, \
 loginShell=shell, \
```

```

 (shadowLastChange,shadowMin,shadowMax, \
 shadowWarning, shadowInactive,shadowExpire)=\
 (shadow, ":"), \
 nisplusEntryOwner=zo_owner, \
 nisplusEntryGroup=zo_group, \
 nisplusEntryAccess=zo_access

```

Similarly, to set NIS+ entry owner, group, and access from LDAP data for the `passwd.org_dir` table, modify the following.

```

nisplusLDAPcolumnFromAttribute \
passwd: name=uid, \
 ("crypt$%s", passwd)=userPassword, \
 uid=uidNumber, \
 gid=gidNumber, \
 gecos=gecos, \
 home=homeDirectory, \
 shell=loginShell, \
 shadow=("%s:%s:%s:%s:%s", \
 shadowLastChange, \
 shadowMin, \
 shadowMax, \
 shadowWarning, \
 shadowInactive, \
 shadowExpire)

```

Edit to read as follows.

```

nisplusLDAPcolumnFromAttribute \
passwd: name=uid, \
 ("crypt$%s", passwd)=authPassword, \
 uid=uidNumber, \
 gid=gidNumber, \
 gecos=gecos, \
 home=homeDirectory, \
 shell=loginShell, \
 shadow=("%s:%s:%s:%s:%s", \
 shadowLastChange, \
 shadowMin, \
 shadowMax, \
 shadowWarning, \
 shadowInactive, \
 shadowExpire), \
 zo_owner=nisplusEntryOwner, \
 zo_group=nisplusEntryGroup, \
 zo_access=nisplusEntryAccess

```

#### 4. Restart the `rpc.nisd` daemon in order to make the mapping change take effect.

First, however, you probably want to upload owner, group, access, and/or TTL entry data to LDAP. Perform an upload as shown in “How to Convert All NIS+ Data to LDAP in One Operation” on page 302.

---

## Principal Names and Netnames

NIS+ authentication relies on principal names (a user or host name, qualified by the domain name) and netnames (the secure RPC equivalent of principal names) to uniquely identify an entity (principal) that can be authenticated. While RFC 2307 provides for storing the Diffie-Hellman keys used for NIS+ authentication, there is no specified place for the principal names or netnames.

The `/var/nis/NIS+LDAPmapping.template` file works around this problem by deriving the domain portion of principal and netnames from the owner name (itself a principal name) of the `cred.org_dir` table. Hence, if the NIS+ domain is `x.y.z.`, and the owner of the `cred.org_dir` table is `aaa.x.y.z.`, all principal names for NIS+ entries created from LDAP data will be of the following form.

*user or system.x.y.z.*

Netnames are of the following form.

**unix.uid@x.y.z.**

**unix.nodename@x.y.z.**

While this method of constructing principal and netnames probably is sufficient for most NIS+ installations, there are also some cases in which it fails, as shown in the following.

- The owner name of the `cred.org_dir` table belongs to a different domain than the one shared by the principal and netnames in the `cred.org_dir` table. This could be the case for a `cred.org_dir` table in a subdomain, if the owner is a principal from the parent domain. You can fix this problem in one of the following ways.
  - *Change the owner of the `cred.org_dir` table to match the domain of the entries in the table.*
  - *Change the mapping rules for the `cred.org_dir` database IDs to use the owner of some other NIS+ object (which could be created especially for that purpose, if no suitable object already exists).*

For example, if the `cred.org_dir` table in the domain `sub.dom.ain.` is owned by `master.dom.ain.`, but principal and netnames in `cred.org_dir.sub.dom.ain.` should belong to `sub.dom.ain.`, you could create a link object as follows.

```
nislnd cred.org_dir.sub.dom.ain. \
credname.sub.dom.ain.
```

Set the owner of the link object to an appropriate principal in the `sub.dom.ain.` as follows.

```
nischown trusted.sub.dom.ain. credname.sub.dom.ain.
```

Edit the mapping file. Change

```
(nis+:zo_owner[]cred.org_dir, ".*%s"), \
```

to

```
(nis+:zo_owner[]credname.sub.dom.ain., ".*%s"), \
```

Note that the use of a link object called `credname` is an example. Any valid object type (except an entry object) and object name will do. The important point is to set the owner of the object to have the correct domain name.

- If you do not want to give ownership even of a special purpose object to a principal from the domain used for the principal and netnames, create `nisplusPrincipalName` and `nisplusNetname` attributes as detailed below.
- *The `cred.org_dir` table contains principal and netnames belonging to more than one domain.*

Consult the documentation for your LDAP server, and create the `nisplusPrincipalName` and `nisplusNetname` attributes, as well as the `nisplusAuthName` object class. (The following is LDIF data for `ldapadd`. Attribute and object class OIDs are for illustration only.)

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.7.0 NAME 'nisplusPrincipalName' \
DESC 'NIS+ principal name' \
SINGLE-VALUE \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.9.0 NAME 'nisplusNetname' \
DESC 'Secure RPC netname' \
SINGLE-VALUE \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.4.1.42.2.27.5.42.42.10.0 NAME 'nisplusAuthName' \
SUP top AUXILLIARY DESC 'NIS+ authentication identifiers' \
MAY (nisplusPrincipalName $ nisplusNetname))
```

You now need to enable the `cred.org_dir` mapping to use the newly created `nisplusNetname` and `nisplusPrincipalName` attributes. The template mapping file, `/var/nis/NIS+LDAPmapping.template`, contains commented-out lines for this purpose. See the `nisplusObjectDN` and `nisplusLDAPattributeFromColumn/nisplusLDAPcolumnFromAttribute` attribute values for the `credlocal`, `creduser`, and `crednode` database IDs. Once you have edited your mapping file to this effect, restart the `rpc.nisd` daemon. Do not forget to add the `-m` option if your mapping file is not the default, and the `-Y` option if the `rpc.nisd` daemon should provide NIS (YP) emulation.

```
pkill rpc.nisd

/usr/sbin/rpc.nisd -m mapping-file [-Y]
```

---

## client\_info and timezone Tables

Because RFC 2307 does not provide schemas for the information kept in the NIS+ `client_info.org_dir` and `timezone.org_dir` tables, mapping of these tables is not enabled by default in the template mapping file (`/var/nis/NIS+LDAPmapping.template`). If you want to keep the `client_info` and `timezone` information in LDAP, consult your LDAP server documentation, and create the new attributes and object classes discussed in the following sections.

## client\_info Attributes and Object Class

Create attributes and object class as below, and then create the container for the `client_info` data. The suggested container name is `ou=ClientInfo`. LDIF data is for `ldapadd(1)`. The attribute and object class OIDs used in the following are examples only.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.12.0 \
 NAME 'nisplusClientInfoAttr' \
 DESC 'NIS+ client_info table client column' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.12.1 \
 NAME 'nisplusClientInfoInfo' \
 DESC 'NIS+ client_info table info column' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.12.2 \
 NAME 'nisplusClientInfoFlags' \
 DESC 'NIS+ client_info table flags column' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.4.1.42.2.27.5.42.42.13.0 \
 NAME 'nisplusClientInfoData' \
 DESC 'NIS+ client_info table data' \
 SUP top STRUCTURAL MUST (cn) \
 MAY (nisplusClientInfoAttr $ nisplusClientInfoInfo $ nisplusClientInfoFlags))
```

To create the container, put the following LDIF data in a file. Substitute your actual search base for `searchBase`.)

```

dn: ou=ClientInfo, searchBase

objectClass: organizationalUnit

ou: ClientInfo

objectClass: top

```

Use the above file as input to the `ldapadd` command in order to create the `ou=ClientInfo` container. For example, if your LDAP administrator DN is `cn=directory manager`, and the file with the LDIF data is called `cifile`, do the following.

```
ldapadd -D cn="directory manager" -f cifile
```

Depending on the authentication required, the `ldapadd` command might prompt for a password.

The `/var/nis/NIS+LDAPmapping.template` file contains commented-out definitions for the `client_info.org_dir` table. Copy these to the actual mapping file, enable by removing the comment character `#`, and restart the `rpc.nisd` daemon. If necessary, synchronize NIS+ and LDAP data as described in “NIS+ to LDAP Migration Scenarios” on page 302.

## timezone Attributes and Object Class

Create attributes and object class as below, and then create the container for the timezone data. The suggested container name is `ou=Timezone`. (The LDIF data is suitable for `ldapadd(1)`. Attribute and object class OIDs are examples only.)

```

dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.15.0 NAME 'nisplusTimeZone' \
DESC 'tzone column from NIS+ timezone table' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.4.1.42.2.27.5.42.42.16.0 NAME 'nisplusTimeZoneData' \
DESC 'NIS+ timezone table data' \
SUP top STRUCTURAL MUST (cn) \
MAY (nisplusTimeZone $ description))

```

To create the `ou=Timezone` container, put the following LDIF data in a file. Substitute your actual search base for `searchBase`.

```

dn: ou=Timezone, searchBase ou: Timezone objectClass: top

objectClass: organizationalUnit

```

Use the above file as input to `ldapadd(1)` in order to create the `ou=Timezone` container. For example, if your LDAP administrator DN is `cn=directory manager`, and the file with the LDIF data is called `tzfile`.

```
ldapadd -D cn="directory manager" -f tzfile
```

Depending on the authentication required, the `ldapadd` command might prompt for a password.

The `/var/nis/NIS+LDAPmapping.template` file contains commented-out definitions for the `timezone.org_dir` table. Copy these to the actual mapping file, enable by removing the comment character `#`, and restart the `rpc.nisd` daemon. If necessary, synchronize NIS+ and LDAP data as described in “NIS+ to LDAP Migration Scenarios” on page 302.

---

## Adding New Object Mappings

The template mapping file, `/var/nis/NIS+LDAPmapping.template`, contains mapping information for all standard NIS+ objects. In order to support mapping of site or application specific objects, you will need to add new mapping entries. This is a simple task for non-entry (that is, directory, group, link, or table) objects, but can become complex for entry objects, if the LDAP organization of the corresponding entry data differs greatly from that used by NIS+. The following example shows the simple case.

### ▼ How to Map Non-Entry Objects

#### 1. Find the fully qualified name of the object to be mapped.

If this name resides under the domain name specified by the `nisplusLDAPbaseDomain` attribute, you can omit the portion that equals the `nisplusLDAPbaseDomain` value.

For example, if `nisplusLDAPbaseDomain` has the value `some.domain.`, and the object to be mapped is a table called `nodeinfo.some.domain.`, the object name can be shortened to `nodeinfo`.

#### 2. Invent a database id to identify the object.

The database id must be unique for the mapping configuration used, but is not otherwise interpreted. It does not show up in the LDAP data. In order to reduce confusion with entry object mappings, create a database id identifying the table object proper (not the table entries) using an explanatory string like `_table` at the end.

For this example, use the database id `nodeinfo_table`, and establish the connection between the database id and the object in the standard mapping file location (`/var/nis/NIS+LDAPmapping`) by adding the following.

```
nisplusLDAPdatabaseIdMapping nodeinfo_table:nodeinfo.some.domain.
```

Assuming that `nisplusLDAPbaseDomain` is `some.domain.`, the following would also work.

```
nisplusLDAPdatabaseIdMapping nodeinfo_table:nodeinfo
```

### 3. Decide on a TTL for the object.

This is the time during which the `rpc.nisd` daemon regards its local copy of the object as valid. When the TTL expires, the next reference to the object will initiate an LDAP lookup to refresh the object.

There are two different TTL values. The first is set when the `rpc.nisd` daemon first loads the object from disk (after a reboot or restart), and the second pertains to all refreshes from LDAP. The first TTL is selected randomly from a configured range. For example, if `nodeinfo_table` should be valid for a period of between one and three hours following initial load, and for twelve hours thereafter, specify the following.

```
nisplusLDAPentryTtl nodeinfo_table:3600:10800:43200
```

### 4. Decide where the object data should be stored in LDAP.

The template mapping file suggests putting non-entry object data in the `ou=nisPlus` container.

If you use this scheme, and have not yet created the appropriate attribute, object class, and container, see “Mapping NIS+ Objects Other Than Table Entries” on page 311.

For example, assume you want to store the `nodeinfo` object in the `ou=nisPlus,dc=some,dc=domain` container, and that the LDAP entry should have the `cn=nodeinfo`. Create the following `nisplusLDAPobjectDN`.

```
nisplusLDAPobjectDN nodeinfo_table:\
 cn=nodeinfo,ou=nisPlus,dc=some,dc=domain?base?\
 objectClass=nisplusObjectContainer:\
 cn=nodeinfo,ou=nisPlus,dc=some,dc=domain?base?\
 objectClass=nisplusObjectContainer,\
 objectClass=top
```

Since NIS+ replicas do not write data to LDAP, you can use the `nisplusLDAPobjectDN` above for both master and replicas.

### 5. (Skip this step if the NIS+ object to be mapped has not yet been created in NIS+.) Store the object data in LDAP. You could use the `rpc.nisd` daemon for this purpose, but it is easier to use the `nisldapmaptest (1M)` utility, as you can leave the `rpc.nisd` daemon running.

```
nisldapmaptest -m /var/nis/NIS+LDAPmapping -o -t nodeinfo -r
```

The `-o` option specifies the table object itself, not the table entries.

### 6. Verify that the object data is stored in LDAP. (This example assumes the LDAP server is running on the local machine at port 389.)

```
ldapsearch -b ou=nisPlus,dc=some,dc=domain cn=nodeinfo
```

The output would appear similar to the following.

```
cn=nodeinfo,ou=nisPlus,dc=some,dc=domain
nisplusobject=NOT ASCII
objectclass=nisplusObjectContainer
objectclass=top
cn=nodeinfo
```

7. Restart the `rpc.nisd` daemon so that it will start using the new mapping information. Do not forget the `-m` option if the mapping file has a name other than the default one. Append the `-Y` option if the `rpc.nisd` daemon is providing NIS (YP) service.

```
pkill rpc.nisd
/usr/sbin/rpc.nisd -m mappingfile [-Y]
```

## Adding Entry Objects

NIS+LDAPmapping(4) specifies the syntax and semantics of table entry mapping in detail, and also provides examples that show how to use each syntactic element. However, the simplest and least error-prone approach is usually to identify an already existing mapping that is similar to what you want to do, and then copy and modify that existing mapping.

For example, assume that you have an NIS+ table called `nodeinfo`, which is used to store inventory and owner information for nodes. Assume that the NIS+ table was created by the following command.

```
nistbladm -c -D access=og=rmcd,nw=r -s : nodeinfo_tbl \
cname=S inventory=S owner= nodeinfo.`domainname`.
```

The `cname` column is expected to contain the canonical name of the node. In other words, the same value as that of the `cname` column in the `hosts.org_dir` table for the node.

Also assume that the corresponding information is kept in the `ou=Hosts` container in LDAP, and that the `nodeInfo` object class (which is an invention for this example, and is not defined in any RFC) has `cn` as a MUST attribute, and that `nodeInventory` and `nodeOwner` are MAY attributes.

In order to upload existing `nodeinfo` data to LDAP, it will be convenient to create the new mapping attributes in a separate file. You could, for example, use `/var/nis/tmpmapping`.

1. Create a database id that identifies the NIS+ table to be mapped.

```
nisplusLDAPdatabaseIdMapping nodeinfo:nodeinfo
```
2. Set the TTL for entries in the `nodeinfo` table. Since the information is expected to change only rarely, use a twelve hour TTL. When the `rpc.nisd` daemon first loads the `nodeinfo` table from disk, the TTLs for entries in the table are randomly selected to be between six and twelve hours.

```
nisplusLDAPentryTtl nodeinfo:21600:43200:43200
```

3. Identify an existing mapping that has similar properties to the one you want to create. In this example, mapping the attribute values is trivial (straight assignment). Instead, the complication is that you store the LDAP data in an existing container, so that you have to be careful during removal of the `nodeinfo` data. You do not want to remove the entire `ou=Hosts` entry, just the `nodeInventory` and `nodeOwner` attributes. You will need a special deletion rule set for this purpose.

To summarize, you are looking for a mapping that shares a container, and has a delete rule set. One possible candidate is the `netmasks` mapping, which shares the `ou=Networks` container, and does have a delete rule set.

4. The template `netmasks` mapping has the default mapping (from `/var/nis/NIS+LDAPmapping.template`) as follows.

```
nisplusLDAPobjectDN netmasks:ou=Networks,?one?objectClass=ipNetwork,\
 ipNetMaskNumber=*\
 ou=Networks,?one?objectClass=ipNetwork:\
 dbid=netmasks_del
```

Transferred to the new mapping for `nodeinfo`, the database id should be `nodeinfo`, the container `ou=Hosts`, and the object class `nodeInfo`. Thus, the first line of the `nodeinfo` mapping becomes the following.

```
nisplusLDAPobjectDN nodeinfo:ou=Hosts,?one?objectClass=nodeInfo,\
```

The second line in the `netmasks` mapping is the part of the search filter that selects only those `ou=Networks` entries that contain the `ipNetMaskNumber` attribute. In this example, select the `ou=Hosts` entries that have the following `nodeInventory` attribute.

```
nodeInventory=*\
```

The third and fourth lines are the write portion of the `nisplusLDAPobjectDN`, and they specify where in LDAP `nodeinfo` data is written, as well as the rule set that is used when `nodeinfo` data is deleted. In this case, create a delete rule set identified by the database id `nodeinfo_del`. Because you are always writing to an existing entry in `ou=Hosts`, you only need to specify the object class for the `nodeinfo` data proper as follows.

```
ou=Hosts,?one?objectClass=nodeInfo:\
 dbid=nodeinfo_del
```

Putting it all together, our `nisplusLDAPobjectDN` is the following.

```
nisplusLDAPobjectDN nodeinfo:ou=Hosts,?one?objectClass=nodeInfo,\
 nodeInventory=*\
 ou=Hosts,?one?objectClass=nodeInfo:\
 dbid=nodeinfo_del
```

5. Create the rule set that maps `nodeinfo` data from NIS+ to LDAP. The template (from `netmasks`) is the following.

```

nisplusLDAPAttributeFromColumn \
 netmasks: dn=("ipNetworkNumber=%s", addr), \
 ipNetworkNumber=addr, \
 ipNetmaskNumber=mask, \
 description=comment

```

The `ou=Hosts` container has an additional complication in this case, as RFC 2307 specifies the `dn` should contain the IP address. However, the IP address is not stored in the `nodeinfo` table, so you must obtain it in another manner. Fortunately, the `crednode` mapping in the template file shows how to obtain the IP address.

```

nisplusLDAPAttributeFromColumn \
 crednode: dn=("cn=%s+ipHostNumber=%s", \
 (cname, "%s.*"), \
 ldap:ipHostNumber:?one?("cn=%s", (cname, "%s.*"))), \

```

Thus, you can copy that portion of the `crednode` mapping. In this case, however, the `cname` column value is the actual host name (not the principal name), so you do not need to extract just a portion of the `cname`. Making the obvious substitutions of attribute and column names, the `nodeinfo` mapping becomes the following.

```

nisplusLDAPAttributeFromColumn \
 nodeinfo: dn=("cn=%s+ipHostNumber=%s", cname, \
 ldap:ipHostNumber:?one?("cn=%s", cname)), \
 nodeInventory=inventory, \
 nodeOwner=owner

```

6. When mapping data from LDAP to NIS+, the template `netmasks` entry is as follows.

```

nisplusLDAPcolumnFromAttribute \
 netmasks: addr=ipNetworkNumber, \
 mask=ipNetmaskNumber, \
 comment=description

```

After substituting attribute and column names, this result is the following.

```

nisplusLDAPcolumnFromAttribute \
 nodeinfo: cname=cn, \
 inventory=nodeInventory, \
 owner=nodeOwner

```

7. The delete rule set for `netmasks` is as follows.

```

nisplusLDAPAttributeFromColumn \
 netmasks_del: dn=("ipNetworkNumber=%s", addr), \
 ipNetmaskNumber=

```

The above specifies that when a `netmasks` entry is deleted in NIS+, the `ipNetmaskNumber` attribute in the corresponding `ou=Networks` LDAP entry is deleted. In this case, delete the `nodeInventory` and `nodeOwner` attributes. Therefore, using the `dn` specification from item (5) above, results in the following.

```

nisplusLDAPattributeFromColumn \
nodeinfo_del: dn=("cn=%s+ipHostNumber=%s", cname, \
ldap:ipHostNumber:one?("cn=%s", cname)), \
nodeInventory=, \
nodeOwner=

```

8. The mapping information is complete. In order to begin using it, stop and later start the `rpc.nisd` daemon.

```
pkill rpc.nisd
```

9. If there already is data in the NIS+ `nodeinfo` table, upload that data to LDAP. Put the new `nodeinfo` mapping information into a separate file, `/var/nis/tmpmapping`.

```
/usr/sbin/rpc.nisd -D -m /var/nis/tmpmapping \
-x nisplusLDAPinitialUpdateAction=to_ldap \
-x nisplusLDAPinitialUpdateOnly=yes
```

10. Add the mapping information from the temporary file, `/var/nis/tmpmapping`, to the actual mapping file. Use an editor to do this, or append the data (assuming the actual mapping file is `/var/nis/NIS+LDAPmapping`) as follows.

```
cp -p /var/nis/NIS+LDAPmapping \
/var/nis/NIS+LDAPmapping.backup
cat /var/nis/tmpmapping >> /var/nis/NIS+LDAPmapping
```

---

**Note** – Note the double arrow redirection, "`>>`". A single arrow, "`>`", would overwrite the target file.

---

11. Restart the `rpc.nisd` daemon. Add the `-Y` option if the `rpc.nisd` daemon also serves NIS (YP) data as follows.

```
/usr/sbin/rpc.nisd -m /var/nis/NIS+LDAPmapping
```

## Storing Configuration Information in LDAP

In addition to keeping NIS+/LDAP configuration information in the configuration files and on the command line, configuration attributes can also be stored in LDAP. This is useful if the configuration information is shared by many NIS+ servers, and is expected to change on a regular basis.

To enable storing of configuration attributes in LDAP, consult your LDAP server documentation and create the following new attributes and object class. The configuration information is expected to reside at the location specified by the `nisplusLDAPconfigDN` value (from the `rpc.nisd` command line, or from `/etc/default/rpc.nisd`), with a `cn` equal to the `nisplusLDAPbaseDomain` value (as it is known to the `rpc.nisd` daemon before reading any configuration information from LDAP).

LDIF data is suitable for `ldapadd(1)` (attribute and object class OIDs are examples only).

The `defaultSearchBase`, `preferredServerList`, and `authenticationMethod` attributes derive from a draft "DUA config" schema, which is intended to become an IETF standard. In any case, the following definitions are sufficient for the purposes of `NIS+LDAPmapping(4)`.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.11.1.3.1.1.1 NAME 'defaultSearchBase' \
DESC 'Default LDAP base DN used by a DUA' \
EQUALITY distinguishedNameMatch \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.11.1.3.1.1.2 NAME 'preferredServerList' \
DESC 'Preferred LDAP server host addresses to be used by a DUA' \
EQUALITY caseIgnoreMatch \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.11.1.3.1.1.6 NAME 'authenticationMethod' \
DESC 'Identifies the authentication method used to connect to the DSA' \
EQUALITY caseIgnoreMatch \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE)
```

NIS+/LDAP configuration attributes are as follows.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.0 \
NAME 'nisplusLDAPTLS' \
DESC 'Transport Layer Security' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.1 \
NAME 'nisplusLDAPTLSCertificateDBPath' \
DESC 'Certificate file' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.2 \
NAME 'nisplusLDAPproxyUser' \
DESC 'Proxy user for data store/retrieval' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.3 \
NAME 'nisplusLDAPproxyPassword' \
DESC 'Password/key/shared secret for proxy user' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.4 \
```

```

 NAME 'nisplusLDAPinitialUpdateAction' \
 DESC 'Type of initial update' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.5 \
 NAME 'nisplusLDAPinitialUpdateOnly' \
 DESC 'Exit after update ?' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.6 \
 NAME 'nisplusLDAPretrieveErrorAction' \
 DESC 'Action following an LDAP search error' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.7 \
 NAME 'nisplusLDAPretrieveErrorAttempts' \
 DESC 'Number of times to retry an LDAP search' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.8 \
 NAME 'nisplusLDAPretrieveErrorTimeout' \
 DESC 'Timeout between each search attempt' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.9 \
 NAME 'nisplusLDAPstoreErrorAction' \
 DESC 'Action following an LDAP store error' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.10 \
 NAME 'nisplusLDAPstoreErrorAttempts' \
 DESC 'Number of times to retry an LDAP store' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.11 \
 NAME 'nisplusLDAPstoreErrorTimeout' \
 DESC 'Timeout between each store attempt' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.12 \
 NAME 'nisplusLDAPrefreshErrorAction' \
 DESC 'Action when refresh of NIS+ data from LDAP fails' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.13 \
 NAME 'nisplusLDAPrefreshErrorAttempts' \
 DESC 'Number of times to retry an LDAP refresh' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.14 \
 NAME 'nisplusLDAPrefreshErrorTimeout' \
 DESC 'Timeout between each refresh attempt' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.15 \
 NAME 'nisplusNumberOfServiceThreads' \
 DESC 'Max number of RPC service threads' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.16 \
 NAME 'nisplusThreadCreationErrorAction' \
 DESC 'Action when a non-RPC-service thread creation fails' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.17 \
 NAME 'nisplusThreadCreationErrorAttempts' \
 DESC 'Number of times to retry thread creation' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)

```

```

attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.18 \
 NAME 'nisplusThreadCreationErrorTimeout' \
 DESC 'Timeout between each thread creation attempt' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.19 \
 NAME 'nisplusDumpErrorAction' \
 DESC 'Action when an NIS+ dump fails' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.20 \
 NAME 'nisplusDumpErrorAttempts' \
 DESC 'Number of times to retry a failed dump' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.21 \
 NAME 'nisplusDumpErrorTimeout' \
 DESC 'Timeout between each dump attempt' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.22 \
 NAME 'nisplusResyncService' \
 DESC 'Service provided during a resync' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.23 \
 NAME 'nisplusUpdateBatching' \
 DESC 'Method for batching updates on master' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.24 \
 NAME 'nisplusUpdateBatchingTimeout' \
 DESC 'Minimum time to wait before pinging replicas' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.25 \
 NAME 'nisplusLDAPmatchFetchAction' \
 DESC 'Should pre-fetch be done ?' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.26 \
 NAME 'nisplusLDAPbaseDomain' \
 DESC 'Default domain name used in NIS+/LDAP mapping' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.27 \
 NAME 'nisplusLDAPdatabaseIdMapping' \
 DESC 'Defines a database id for an NIS+ object' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.28 \
 NAME 'nisplusLDAPentryTtl' \
 DESC 'TTL for cached objects derived from LDAP' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.29 \
 NAME 'nisplusLDAPobjectDN' \
 DESC 'Location in LDAP tree where NIS+ data is stored' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.30 \
 NAME 'nisplusLDAPcolumnFromAttribute' \
 DESC 'Rules for mapping LDAP attributes to NIS+ columns' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.42.18.31 \
 NAME 'nisplusLDAPattributeFromColumn' \
 DESC 'Rules for mapping NIS+ columns to LDAP attributes' \

```

```

SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.4.1.42.2.27.5.42.42.19.0 NAME 'nisplusLDAPconfig' \
DESC 'NIS+/LDAP mapping configuration' \
SUP top STRUCTURAL MUST (cn) \
MAY (preferredServerList $ defaultSearchBase $
authenticationMethod $ nisplusLDAPTLS $ nisplusLDAPTLSCertificateDBPath
$ nisplusLDAPproxyUser $ nisplusLDAPproxyPassword $ nisplusLDAPinitialUpdateAction
$ nisplusLDAPinitialUpdateOnly $ nisplusLDAPretrieveErrorAction
$ nisplusLDAPretrieveErrorAttempts $ nisplusLDAPretrieveErrorTimeout
$ nisplusLDAPstoreErrorAction $ nisplusLDAPstoreErrorAttempts
$ nisplusLDAPstoreErrorTimeout $ nisplusLDAPrefreshErrorAction
$ nisplusLDAPrefreshErrorAttempts $ nisplusLDAPrefreshErrorTimeout
$ nisplusNumberOfServiceThreads $nisplusThreadCreationErrorAction
$ nisplusThreadCreationErrorAttempts $ nisplusThreadCreationErrorTimeout
$ nisplusDumpErrorAction $ nisplusDumpErrorAttempts
$ nisplusDumpErrorTimeout $ nisplusResyncService $ nisplusUpdateBatching
$ nisplusUpdateBatchingTimeout $ nisplusLDAPmatchFetchAction
$ nisplusLDAPbaseDomain $ nisplusLDAPdatabaseIdMapping $ nisplusLDAPentryTtl
$ nisplusLDAPobjectDN $ nisplusLDAPcolumnFromAttribute !
$ nisplusLDAPattributeFromColumn))

```

Create a file containing the following LDIF data (substitute your actual search base for *searchBase*, and the fully qualified domain name for *domain*.)

```

dn: cn=domain, searchBase

cn: domain

objectClass: top objectClass: nisplusLDAPconfig

```

Use the above file as input to `ldapadd(1)` to create the NIS+/LDAP configuration entry. Initially, the entry is empty. Use `ldapmodify(1)` to add configuration attributes. For example, to set the `nisplusNumberOfServiceThreads` attribute to "32", create the following file (for input to `ldapmodify(1)`).

```

dn: cn=domain, searchBasenisplusNumberOfServiceThreads: 32

```



# System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) Updates

---

---

## Solaris 9 12/02 Updates

The LDAP naming services password management feature is enabled. New information concerning in the password management feature is in Chapters 13, 14, 16, 17 and 18.



# Glossary

---

|                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>application-level naming service</b> | Application-level naming services are incorporated in applications offering services such as files, mail, and printing. Application-level naming services are bound below enterprise-level naming services. The enterprise-level naming services provide contexts in which contexts of application-level naming services can be bound.                                                                                                                                                                              |
| <b>authentication</b>                   | The means by which a server can verify a client's identity.                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>cache manager</b>                    | The program that manages the local caches of NIS+ clients ( <code>NIS_SHARED_DIRCACHE</code> ), which are used to store location information about the NIS+ servers that support the directories most frequently used by those clients, including transport addresses, authentication information, and a time-to-live value.                                                                                                                                                                                        |
| <b>child domain</b>                     | See <i>domain</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>client</b>                           | (1) The client is a principal (machine or user) requesting an naming service from an naming server.<br><br>(2) In the client-server model for file systems, the client is a machine that remotely accesses resources of a compute server, such as compute power and large memory capacity.<br><br>(3) In the client-server model, the client is an <i>application</i> that accesses services from a "server process." In this model, the client and the server can run on the same machine or on separate machines. |
| <b>client-server model</b>              | A common way to describe network services and the model user processes (programs) of those services. Examples include the name-server/name-resolver paradigm of the <i>Domain Name System (DNS)</i> . See also <i>client</i> .                                                                                                                                                                                                                                                                                      |
| <b>credentials</b>                      | The authentication information that the client software sends along with each request to a naming server. This information verifies the identity of a user or machine.                                                                                                                                                                                                                                                                                                                                              |

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>data encrypting key</b>            | A key used to encipher and decipher data intended for programs that perform encryption. Contrast with <i>key encrypting key</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>data encryption standard (DES)</b> | A commonly used, highly sophisticated algorithm developed by the U.S. National Bureau of Standards for encrypting and decrypting data. See also SUN-DES-1.                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>decimal dotted notation</b>        | The syntactic representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. Used to represent IP addresses in the Internet as in: 192.67.67.20.                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>DES</b>                            | See <i>data encryption standard (DES)</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>directory</b>                      | (1) An LDAP directory is a container for LDAP objects. In UNIX, a container for files and subdirectories.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>directory cache</b>                | A local file used to store data associated with directory objects.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>directory information tree</b>     | The DIT is the distributed directory structure for a given network. By default, Solaris LDAP clients access the information assuming that the DIT has a given structure. For each domain supported by the LDAP server, there is an assumed subtree with an assumed structure.                                                                                                                                                                                                                                                                                                                                          |
| <b>distinguished name</b>             | A distinguished name is an entry in an X.500 directory information base (DIB) composed of selected attributes from each entry in the tree along a path leading from the root down to the named entry.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>DIT</b>                            | See <i>directory information tree</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>DNS</b>                            | See <i>Domain Name System</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>DNS-forwarding</b>                 | An NIS server or an NIS+ server with NIS compatibility set forwards requests it cannot answer to DNS servers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>DNS zones</b>                      | Administrative boundaries within a network domain, often made up of one or more subdomains.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>DNS zone files</b>                 | A set of files wherein the DNS software stores the names and IP addresses of all the workstations in a domain.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>domain</b>                         | <p>(1) In NIS+ a group of hierarchical objects managed by NIS+. There is one highest level domain (root domain) and zero or more subdomains. Domains and subdomains may be organized around geography, organizational or functional principles.</p> <ul style="list-style-type: none"> <li>■ <i>Parent domain</i>. Relative term for the domain immediately above the current domain in the hierarchy.</li> <li>■ <i>Child domain</i>. Relative term for the domain immediately below the current domain in the hierarchy.</li> <li>■ <i>Root domain</i>. Highest domain within the current NIS+ hierarchy.</li> </ul> |

(2) In the Internet, a part of a naming hierarchy usually corresponding to a Local Area Network (LAN) or Wide Area Network (WAN) or a portion of such a network. Syntactically, an Internet domain name consists of a sequence of names (labels) separated by periods (dots). For example, `sales.doc.com`.

(3) In International Organization for Standardization's open systems interconnection (OSI), "domain" is generally used as an administrative partition of a complex distributed system, as in MHS private management domain (PRMD), and directory management domain (DMD).

|                                    |                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>domain name</b>                 | The name assigned to a group of systems on a local network that share DNS administrative files. The domain name is required for the network information service database to work properly. See also <i>domain</i> .                                                                                                                                                                                     |
| <b>Domain naming service (DNS)</b> | A service that provides the naming policy and mechanisms for mapping domain and machine names to addresses outside of the enterprise, such as those on the Internet. DNS is the network information service used by the Internet.                                                                                                                                                                       |
| <b>encryption</b>                  | The means by which the privacy of data is protected.                                                                                                                                                                                                                                                                                                                                                    |
| <b>encryption key</b>              | See <i>data encrypting key</i> .                                                                                                                                                                                                                                                                                                                                                                        |
| <b>enterprise-level network</b>    | An "enterprise-level" network can be a single Local Area Network (LAN) communicating over cables, infra-red beams, or radio broadcast; or a cluster of two or more LANs linked together by cable or direct phone connections. Within an enterprise-level network, every machine is able to communicate with every other machine without reference to a global naming service such as DNS or X.500/LDAP. |
| <b>entry</b>                       | A single row of data in a database table.                                                                                                                                                                                                                                                                                                                                                               |
| <b>federated namespace</b>         | An FNS (XFN) term referring to the set of all possible names generated according to the policies that govern the relationships among member naming systems and their respective namespaces.                                                                                                                                                                                                             |
| <b>FNS</b>                         | See <i>Federated naming service</i> .                                                                                                                                                                                                                                                                                                                                                                   |
| <b>GID</b>                         | See <i>group ID</i> .                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>global naming service</b>       | A global naming service identifies (names) those enterprise-level networks around the world that are linked together via phone, satellite, or other communication systems. This world-wide collection of linked networks is known as the "Internet." In addition to naming networks, a global naming service also identifies individual machines and users within a given network.                      |
| <b>group ID</b>                    | A number that identifies the default <i>group</i> for a user.                                                                                                                                                                                                                                                                                                                                           |
| <b>indexed name</b>                | A naming format used to identify an entry in a table.                                                                                                                                                                                                                                                                                                                                                   |

|                                 |                                                                                                                                                                                                                                                                    |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Internet address</b>         | A 32-bit address assigned to hosts using <i>TCP/IP</i> . See <i>decimal dotted notation</i> .                                                                                                                                                                      |
| <b>IP</b>                       | Internet Protocol. The <i>network layer</i> protocol for the Internet protocol suite.                                                                                                                                                                              |
| <b>IP address</b>               | A unique number that identifies each host in a network.                                                                                                                                                                                                            |
| <b>key (encrypting)</b>         | A key used to encipher and decipher other keys, as part of a key management and distribution system. Contrast with <i>data encrypting key</i> .                                                                                                                    |
| <b>key server</b>               | A Solaris operating environment process that stores private keys.                                                                                                                                                                                                  |
| <b>LDAP</b>                     | Lightweight Directory Access Protocol is a standard, extensible directory access protocol used by LDAP naming service clients and servers to communicate with each other.                                                                                          |
| <b>local-area network (LAN)</b> | Multiple systems at a single geographical site connected together for the purpose of sharing and exchanging data and software.                                                                                                                                     |
| <b>mail exchange records</b>    | Files that contain a list of DNS domain names and their corresponding mail hosts.                                                                                                                                                                                  |
| <b>mail hosts</b>               | A workstation that functions as an email router and receiver for a site.                                                                                                                                                                                           |
| <b>master server</b>            | The server that maintains the master copy of the network information service database for a particular domain. Namespace changes are always made to the naming service database kept by the domain's master server. Each domain has only <i>one</i> master server. |
| <b>MIS</b>                      | Management information systems (or services)                                                                                                                                                                                                                       |
| <b>name resolution</b>          | The process of translating workstation or user names to addresses.                                                                                                                                                                                                 |
| <b>name server</b>              | Servers that run one or more network naming services.                                                                                                                                                                                                              |
| <b>naming service switch</b>    | A configuration file ( <i>/etc/nsswitch.conf</i> ) that defines the sources from which a naming client can obtain its network information.                                                                                                                         |
| <b>naming service</b>           | A network service that handles machine, user, printer, domain, router, and other network names and addresses.                                                                                                                                                      |
| <b>namespace</b>                | (1) A namespace stores information that users, workstations, and applications must have to communicate across the network.<br><br>(2) The set of all names in a naming system.                                                                                     |
| <b>network mask</b>             | A number used by software to separate the local subnet address from the rest of a given Internet protocol address.                                                                                                                                                 |
| <b>network password</b>         | See Secure RPC password.                                                                                                                                                                                                                                           |
| <b>NIS</b>                      | A distributed network information service containing key information about the systems and the users on the network. The NIS database is stored on the <i>master server</i> and all the <i>replica</i> or <i>slave servers</i> .                                   |

|                                    |                                                                                                                                                                                                                                                                                                                                            |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NIS maps</b>                    | A file used by NIS that holds information of a particular type, for example, the password entries of all users on a network or the names of all host machines on a network. Programs that are part of the NIS service query these maps. See also <i>NIS</i> .                                                                              |
| <b>NIS+</b>                        | A distributed network information service containing hierarchical information about the systems and the users on the network. The NIS+ database is stored on the <i>master server</i> and all the <i>replica servers</i> .                                                                                                                 |
| <b>NIS-compatibility mode</b>      | A configuration of NIS+ that allows NIS clients to have access to the data stored in NIS+ tables. When in this mode, NIS+ servers can answer requests for information from both NIS and NIS+ clients.                                                                                                                                      |
| <b>parent domain</b>               | See <i>domain</i> .                                                                                                                                                                                                                                                                                                                        |
| <b>preferred server list</b>       | A <code>client_info</code> table or a <code>client_info</code> file. Preferred server lists specify the preferred servers for a client or domain.                                                                                                                                                                                          |
| <b>private key</b>                 | The private component of a pair of mathematically generated numbers, which, when combined with a private key, generates the DES key. The DES key in turn is used to encode and decode information. The private key of the sender is only available to the owner of the key. Every user or machine has its own public and private key pair. |
| <b>public key</b>                  | The public component of a pair of mathematically generated numbers, which, when combined with a private key, generates the DES key. The DES key in turn is used to encode and decode information. The public key is available to all users and machines. Every user or machine has their own public and private key pair.                  |
| <b>record</b>                      | See <i>entry</i> .                                                                                                                                                                                                                                                                                                                         |
| <b>remote procedure call (RPC)</b> | An easy and popular paradigm for implementing the client-server model of distributed computing. A request is sent to a remote system to execute a designated procedure, using arguments supplied, and the result is returned to the caller.                                                                                                |
| <b>reverse resolution</b>          | The process of converting workstation IP addresses to workstation names using the DNS software.                                                                                                                                                                                                                                            |
| <b>root domain</b>                 | See <i>domain</i> .                                                                                                                                                                                                                                                                                                                        |
| <b>RPC</b>                         | See remote procedure call (RPC).                                                                                                                                                                                                                                                                                                           |
| <b>SASL</b>                        | The simple authentication and security layer. A framework for negotiating authentication and security layer semantics in application-layer protocols.                                                                                                                                                                                      |
| <b>Secure RPC password</b>         | Password required by Secure RPC protocol. This password is used to encrypt the private key. This password should always be identical to the user's login password.                                                                                                                                                                         |

|                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>server</b>                           | <p>(1) In NIS+, NIS, DNS, and LDAP a host machine providing naming services to a network.</p> <p>(2) In the <i>client-server model</i> for file systems, the server is a machine with computing resources (and is sometimes called the compute server), and large memory capacity. Client machines can remotely access and make use of these resources. In the client-server model for window systems, the server is a process that provides windowing services to an application, or “client process.” In this model, the client and the server can run on the same machine or on separate machines.</p> <p>(3) A <i>daemon</i> that actually handles the providing of files.</p> |
| <b>server list</b>                      | See preferred server list.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>slave server</b>                     | <p>(1) A server system that maintains a copy of the NIS database. It has a disk and a complete copy of the operating environment.</p> <p>(2) Slave servers are called <i>replica servers</i> in NIS+.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>SSL</b>                              | SSL is the secure sockets layer protocol. It is a generic transport-layer security mechanism designed to make application protocols such as LDAP secure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>subnet</b>                           | A working scheme that divides a single logical network into smaller physical networks to simplify routing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>table</b>                            | In NIS+ a two-dimensional (nonrelational) database object containing NIS+ data in rows and columns. (In NIS an NIS map is analogous to a NIS+ table with two columns.) A table is the format in which NIS+ data is stored. NIS+ provides 16 predefined or system tables. Each table stores a different type of information.                                                                                                                                                                                                                                                                                                                                                        |
| <b>TCP</b>                              | See <i>Transport Control Protocol (TCP)</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>TCP/IP</b>                           | Acronym for Transport Control Protocol/Interface Program. The protocol suite originally developed for the Internet. It is also called the <i>Internet</i> protocol suite. Solaris networks run on TCP/IP by default.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Transport Control Protocol (TCP)</b> | The major transport protocol in the Internet suite of protocols providing reliable, connection-oriented, full-duplex streams. Uses IP for delivery. See TCP/IP.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Transport Layer Security (TLS)</b>   | TLS secures communication between an LDAP client and the directory server, providing both privacy and data integrity. The TLS protocol is a super set of the Secure Sockets Layer (SSL) protocol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>wide-area network (WAN)</b>          | A network that connects multiple local-area networks (LANs) or systems at different geographical sites via phone, fiber-optic, or satellite links.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>X.500</b>                            | A global-level directory service defined by an Open Systems Interconnection (OSI) standard. A precursor to LDAP.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

# Index

---

## Numbers and Symbols

- + netgroup, 75
- +/- Syntax
  - compat, 45
  - nsswitch.conf files, 45
  - passwd\_compat, 45
- +/- syntax, switch files, and, 74
- “not responding” messages (NIS), 171
- \$PWDIR/security/passwd.adjunct, 157
- \$PWDIR/shadow, 137
- “unavailable” messages (NIS), 171

## A

- adjunct files, 144
- administrative domain (DNS)**, 52
- aliases files, 143
- application-level**, 333
- .asc, 165
- Attribute map, 209
- Attributes, internet print protocol, 278
- authentication
  - digest-MD5, 217
  - simple, 217
- authentication methods, none, 217
- auto\_direct.time maps, 159
- auto\_home tables, nsswitch.conf file,
  - and, 37
- auto\_home.time maps, 159
- auto\_master tables, nsswitch.conf file,
  - and, 37
- awk, 164

## B

- Browsing indices, 234

## C

- cache manager**, 333
- Can't find messages, 76
- Can't find messages (DNS), 120
- can't initialize address messages, 76
- child domain**, 333
- CHKPIPE, 159
- client**, 333
- client-server model**, 333
- clients
  - NIS, 127
  - NIS setup, 149
- Credential Levels, LDAP client, 215
- Credential Storage, LDAP client, 216
- credentials**, 333
- crontab, 164
- crontab, NIS, problems, 177
- crontab, NIS maps propagating, 162
- crontab files, 162
  - NIS, problems, 177

## D

- daemons
  - NIS, 128
  - NIS, not running, 176
  - NIS, starting, 146

## daemons (Continued)

- nscd, 43
- rpc.yppasswdd daemon, 128
- rpc.yupdated daemon, 128
- ypbind daemon, 128
- ypserv daemon, 128
- ypupdated, 136
- ypxfr daemon, 128

## data encrypting key, 334

Database format error messages  
(DNS), 121

dbm, 165, 166

## decimal dotted notation, 334

defaultdomain files, 140

DES, 334

DIR directory, 143

directory, 334

directory cache, 334

Directory Information Tree, 206, 334

directory server, 307

distinguished name, 334

DNS, 334, 335

DNS, 30

- \$INCLUDE control entry, 109

- \$INCLUDE files, 106

- \$ORIGIN() control entry, 109

- A record, 112

- administrative domains, 50, 52

- backup files, 73

- boot files, 97

- cache-only servers, 53, 73

- Can't find messages, 76, 120

- can't initialize address  
messages, 76

- changes erratic, 118

- class fields, 107

- clients, 50

- CNAME record, 114

- control entries, 109

- data files, 97

- data files, names of, 97

- Database format error messages, 121

- default domain name, 56, 84

- domain name

  - trailing dots in, 56

- domain names, 56, 62

  - default, 56

  - fully qualified, 63

## DNS (Continued)

- domain names, registering, 62

- domain names, trailing dots, 75

- domains, 96

  - top level, 61

- domains, geographic (Internet), 61

- domains, organizational (Internet), 61

- email, and, 97

- error receiving zone transfer  
messages, 121

- example, 87

- file names, 54

- filenames, and, 97

- ftp problems, 120

- HINFO record, 112

- hosts files, 103

- hosts.rev files, 104

- illegal messages, 121

- in-addr.arpa Domain, 65

- in.named, 53

- in.named, updating, 78

- Internet, and, 61

- Internet, joining, 62

- inverse queries, 84

- IP addresses, 50

- IP registration, 71

- local loopback, 72

- LOCALDOMAIN, 84

- machines, adding, 79

- machines, removing, 80

- master server (master), 71

- master server (slave), 72

- master servers, changes on, 79

- modifying, 77

- MX record, 114

- MX records, 97

- name-address resolution, 50, 51

- name fields, 107

- named.ca files, 101

- named.conf file, 98

- named.local files, 105

- namespace, 96

- namespace, hierarchy, 96

- network, division into subdomains, 82

- NIS, and, 125, 126

- NIS and, 168

- No such... messages, 121

## DNS (Continued)

- Non-authoritative answer messages, 121
- non-authoritative messages, 76
- Non-existent domain messages, 76
- NS record, 112
- nsswitch.conf file, and, 34
- nsswitch.conf files, 44, 57
- primary servers, 53
- problem solving, 117
- PTR record, 114
- record-specific-data* fields, 108
- record-type fields, 108
- reloading data, 78
- resolver, 56
- resource records
  - formats of, 106
  - special characters, 108
  - types of, 109
- reverse domain data problems, 119
- reverse mapping, 65
- reverse resolution, 65
- RFC1535, 84
- rlogin problems, 120
- root domain servers, 51
- rsh problems, 120
- secondary servers, 53
- server cannot find machine, 117
- server failed messages, 119
- server function, specifying, 87
- server initialization, 75
- servers, 50
- servers, slave, 71
- servers, types of, 53
- setup testing, 75
- short names, client cannot use, 119
- SOA, changing number, 78
- SOA record, 110
- Solaris implementation of, 84
- subdomain setup (different zones), 95
- subdomain setup (same zone), 94
- subdomains, 96
- subdomains, creating, 81
- subdomains, names of, 82
- subdomains, planning, 81
- subdomains, set up, 83
- syntax errors, 121
- test programs, 84

## DNS (Continued)

- TTL fields, 107
- Unknown field messages, 121
- unreachable messages, 120
- utility scripts, 84
- version of, 84
- WKS record, 113
- zone
  - files, 64
  - zone expired messages, 120
  - zone reverse map, 71
  - zones, 64
- DNS client, setting up, 68
- DNS clients, resolver and, 56
- DNS data files, setup, 93
- DNS files, names of, 53
- DNS-forwarding**, 334
- DNS server setup, DNS file names, 53
- DNS zone files**, 334
- DNS zones**, 334
- DOM variable, 145, 146
- domain**, 334
- domain name**, 335
- domainname, 145, 147
- domains
  - DNS, trailing dots, 75
  - domain names (DNS), 62
  - domain names, registering, 62
  - geographic (Internet), 61
  - in-addr.arpa, 65
  - Internet, 61
  - names of (DNS), 56
  - NIS, 126, 128, 140
  - NIS, multiple, 145
  - organizational (Internet), 61

## E

- encryption key**, 335
- enterprise-level network**, 335
- entry**, 335
- error receiving zone transfer messages (DNS), 121
- /etc/defaultdomain files, 140, 172
- /etc files, 29, 45, 129
- /etc/hosts, 24, 147
- /etc/inet/ipnodes, 24

- /etc/init.d/yp, 136
- /etc/mail/aliases files, 143
- /etc/mail directory, 143
- /etc/named.conf file, 99
- /etc/named.conf files, 58
- /etc/named.pid files, 78
- /etc/nodename files, 140
- /etc/nsswitch.conf, 43
- /etc/nsswitch.files, 42
- /etc/nsswitch.nis, 42
- /etc/nsswitch.nisplus, 42
- /etc/resolv.conf files, 69, 85
  - NIS and Internet, 86

## F

- federated namespace**, 335
- files-based naming, 30
- FNS**, 335
- FQDN**, 199
- ftp, 178
  - problems, 121

## G

- gethostbyname(), 33
- getipnodebyname(), 33
- getpwnam(), 33
- getpwuid(), 33
- getXbyY(), 33
- GID**, 335
- global naming service**, 335
- group ID**, 335
- groups
  - +/- syntax, and, 74
  - netgroups (NIS), 154, 155

## H

- hosts (DNS file)
  - examples, 90, 104
  - setup, 103
  - subdomains, and, 95
  - zones, multiple, 95
- hosts (DNS files), 54

- hosts (DNS files) (Continued)
  - identified in named.boot file, 71
  - zones, multiple and, 54
- hosts (machines)
  - multihome support (NIS), 137
  - NIS clients, 127
  - NIS domains, changing, 168
  - NIS servers, 127
- hosts.byaddr, 130
- hosts.byaddr maps, YP\_INTERDOMAIN key, 85
- hosts.byname, 130
- hosts.byname maps, 130
  - YP\_INTERDOMAIN key, 85
- hosts database, 160
- hosts file (DNS), 72, 104
- hosts files, 79, 147
- hosts files (DNS), 103
- hosts.rev file, 91
  - examples, 91
  - subdomains, and, 95
  - zones, multiple, 95
- hosts.rev file (DNS), 105
- hosts.rev files, 54, 79, 104
  - examples, 105
  - setup, 105
  - subdomains (same zone), 94

## I

- illegal messages (DNS), 121
- in.named, 30, 53
- in.named file, 73, 93
- index LDAP client attributes, 233
- indexed name**, 335
- Internet
  - DNS, and, 61
  - domain names, registering, 62
  - domains
    - top level, 61
  - domains, geographic, 61
  - domains, organizational, 61
  - joining, 62
  - named.ca file (DNS), 101
  - NIS, and, 126
  - nsswitch.conf files, 44
- Internet address**, 336

**IP**, 336  
**IP address**, 336  
IPv6, `nsswitch.conf` files, 44

## K

**key (encrypting)**, 336  
**key server**, 336  
keyserver, `nsswitch.conf` file, and, 38

## L

**LAN**, 336  
**LDAP**  
    password management, 221  
    transitioning from NIS+ to, 291  
    troubleshooting, 255  
`ldap_cachemgr` daemon, 213  
LDAP schema, role based attributes, 277  
LDAP schema role based, object classes, 277  
LDAP schemas, 261  
LDAP troubleshooting  
    ldapclient cannot bind to server, 258  
    login fails, 257  
    lookup too slow, 258  
    unable to reach systems in LDAP domain  
        remotely, 257  
        unresolved hostname, 257  
`ldapaddent`, 240  
LDIF, 203  
list of, 130  
local loopback (DNS), 72  
LOCALDOMAIN, 56  
`ls`, 172

## M

**mail exchange records**, 336  
**mail hosts**, 336  
Mailgroups  
    attributes, 273  
    object class, 273  
make, 145, 155, 158, 161, 167  
make, 160

make  
    NIS maps, 133  
    NIS maps and, 132  
Make files, NIS, 130  
makedbm, 129, 130, 133, 144, 145, 159, 164, 165  
    maps, changing server of, 156, 157  
    slave servers, adding, 166  
Makefile, 157, 159  
    NIS security, 151  
    non-default maps, modifying, 164  
Makefile, propagating maps, 162  
Makefile  
    YP\_INTERDOMAIN key, 85  
Makefile files, 141, 143, 144, 145  
    4.x compatibility, 137  
    maps, supported list, 157  
    multihome support, 137  
`mapname.dir` files, 144  
`mapname.pag` files, 144  
**master server**, 336  
masters, 306  
**MIS**, 336  
`mymap.asc` files, 165

## N

**name resolution**, 336  
**name server**, 336  
name space, DNS, 30  
**name-to-address resolution**, 50  
`named.boot` file  
    examples, 88, 89  
`named.boot` files  
    backup files, 73  
    cache-only servers, 73  
    local loopback, 72  
    master server (master), 71  
    master server (slave), 72  
    zone reverse map, 71  
`named.ca` file, 103  
    example (Internet version), 101  
    examples, 92  
    Internet version of, 101  
    non-Internet version of, 102  
    setup the root servers, 101  
`named.ca` files, 54, 101  
`named.conf` file, 99

- named.conf files, 53, 58
  - DNS server function, 87
  - examples, 59
  - setup (servers), 58
- named.local file, 72, 106
  - examples, 90, 106
- named.local files, 54, 105
  - setup, 105
- named.pid files, 78
- named.root file, 101
- namespace**, 336
- naming, 23
  - DNS, 30
  - files-based, 30
  - NIS, 30
  - Solaris naming services, 29
- naming service**, 336
- naming service switch**, 336
- ndbm, 129, 143
  - slave servers, adding, 166
- ndbm files, maps, changing server of, 157
- netgroup.byhost files, 154
- netgroup.byuser files, 154
- netgroup files, 154
  - entries, example, 155
- netnames, 316
- netstat, testing, 173
- network mask**, 336
- network password**, 336
- nicknames files, 133
- NIS**, 336
- NIS, 30, 125
  - “not responding” messages, 171
  - “unavailable” messages, 171
  - 4.x compatibility, 137
  - architecture, 126
  - binding, 134
  - binding, broadcast, 134
  - binding, server-list, 134
  - broadcast binding, 135
  - C2 security, 167
  - client problems, 172
  - client setup, 149
  - clients, 127
  - commands hang, 171
  - components, 128
  - configuration files, modifying, 157
  - crontab, 162
- NIS (Continued)
  - daemons, 128
  - daemons, not running, 176
  - daemons, starting, 146
  - DNS, and, 126
  - DNS and, 168
  - domain names, 140
  - domains, 126, 128
  - domains, multiple, 145
  - earlier versions and, 136
  - halting, 170
  - Internet, and, 126
  - madedbm, 129
  - make, 133
  - Make files, 130
  - makedbm, 133
  - Makefile filtering, 158
  - makefile preparation, 143
  - master servers, 127
  - multihome support, 137
  - ndbm format, 129
  - netgroups, 154, 155
  - NSKit, 136
  - passwd maps, updatingpasswd maps, 153
  - passwd maps auto update, 162
  - password data, 141
  - passwords, user, 153
  - problems, 171
  - root entry, 151
  - rpc.yppasswdd, 128, 153
  - rpc.ypupdated, 128
  - securenets, 136
  - security, 136, 151
  - server binding not possible, 174
  - server-list binding, 135
  - servers, 127
  - servers, malfunction, 176
  - servers, maps different versions, 176
  - servers, overloaded, 175
  - servers not available, 173
  - setup, preparation for, 140, 141
  - slave server setup, 147
  - slave servers, 127
  - software installation, 136
  - source files, 141, 142
  - starting, 136, 146
  - starting, automatic, 146
  - starting, command line, 146

- NIS (Continued)
  - stopping, 136, 170
  - structure of, 126
  - SunOS 4.x compatibility, 137
  - SUNWypcr, 136
  - SUNWypu, 136
  - updates, automating, 162
  - updating via shell scripts, 162
  - user password locked, 152
  - useradd, 152
  - userdel, 153
  - users, adding, 152
  - users, administering, 152
  - utility programs, 129
  - /var/yp/, 130
  - versions, earlier, 136
  - ypbind, 128, 133, 135
  - ypbind “can’t” messages, 171
  - ypbind fails, 174
  - ypcat, 129, 133
  - ypinit, 129, 133, 144
  - ypmatch, 129, 133
  - yppoll, 129
  - yppush, 133
  - yppush, 129
  - ypserv, 128, 133
  - ypservers files, 166
  - ypset, 129, 133
  - ypstart, 136
  - ypstop, 136
  - ypupdated, 136
  - ypwhich, 129, 133, 136
  - ypwhich inconsistent displays, 174
  - ypxfr, 128, 129, 133
- NIS+**, 337
- NIS clients, not bound to server, 173
- NIS-compatibility mode**, 337
- NIS domain names
  - incorrect, 172
  - missing, 172
- NIS domains, changing, 168
- NIS hosts, changing domain of, 168
- NIS maps**, 337
- NIS maps, 130
  - administering, 155
  - CHKPIPE in Makefile, 159
  - commands related to, 133
  - configuration files, modifying, 157
- NIS maps (Continued)
  - crontab, 162
  - default, 130
  - descriptions of, 130
  - displaying contents, 155
  - displaying contents of, 132
  - format is ndbm, 129
  - locating, 132
  - Makefile, DIR variable, 160
  - Makefile, DOM variable, 160
  - Makefile, PWDIR variable, 160
  - Makefile and, 158
  - Makefile entries, updating, 161
  - Makefile filtering, 158
  - Makefile macros, changing, 160
  - Makefile variables, changing, 160
  - making, 132
  - new maps, creating from files, 165
  - new maps, creating from keyboard, 165
  - nicknames, 133
  - non-default, 161
  - NOPUSH in Makefile, 159
  - propagating, 162
  - server, changing, 156
  - updates, automating, 162
  - updating, 132
  - updating via shell scripts, 162
  - /var/yp/, 130
  - working with, 132
  - yppush in Makefile, 159
  - ypxfr, crontab file in, 162
  - ypxfr, invoking directly, 164
  - ypxfr, logging, 164
  - ypxfr, shell scripts in, 162
- NIS slave servers
  - adding, 166
  - initializing, 167
- No such... messages (DNS), 121
- nodenamefiles, 140
- Non-authoritative answer messages (DNS), 121
- non-authoritative messages, 76
- Non-existent domain messages, 76
- NOPUSH in Makefile, 159
- nscd daemon, 43
- nslookup, 76
- nsswitch.conf files, 30, 33, 38, 46, 83, 117, 140

- nsswitch.conf files (Continued)
  - +/- Syntax, 45
  - +/- syntax, compatibility, 74
  - actions, 36
  - Auto\_home table, 37
  - Auto\_master table, 37
  - choosing a file, 43
  - comments in, 38
  - compat, 45
  - continue, 36
  - default file, 42
  - default files, 42
  - default template files, 39
  - DNS, and, 34, 44, 57
  - examples, 39, 40, 41
  - format of, 34
  - incorrect syntax, 37
  - information sources, 35
  - installation of, 43
  - Internet access, 44
  - IPv6, and, 44
  - keyserver entry, 38
  - messages, status, 35
  - missing, 37
  - modifying, 37
  - NIS, 126
  - NOTFOUND=continue, 36
  - nsswitch.files files, 38, 39
  - nsswitch.nis files, 39
  - nsswitch.nisplus files, 39
  - options, 36
  - passwd\_compat, 45
  - password data, 46
  - publickey entry, 38
  - return, 36
  - search criteria, 35, 36
  - sources, 35
  - status messages, 35, 36
  - SUCCESS=return, 36
  - templates, 33, 38, 42
  - timezone table, 37
  - TRYAGAIN=continue, 36
  - UNAVAIL=continue, 36
- nsswitch.conf files, 43
- nsswitch.files files, 42
- nsswitch.ldap, 41
- nsswitch.nis, 40
- nsswitch.nis files, 42

- nsswitch.nisplus files, 42

## O

- object mappings, adding new, 320
- objectClass Map, 210

## P

- parent domain**, 337
- passwd, 153
  - NIS map auto updated, 162
- passwd.adjunct files, 144, 154, 157, 167
- passwd files
  - 4.x compatibility (NIS), 137
  - Solaris 1.x formats, 152
- passwd map, 141
- passwd maps, users, adding, 152
- password data
  - +/- syntax, and, 74
  - NIS, 141
  - NIS, and, 151
  - nsswitch.conf files, 46
  - root in NIS maps, 151
- Password Management, 221
- passwords
  - NIS, and, 153
  - rpc.yppasswdd (NIS), 153
- ping, 176
- Pluggable Authentication Methods, 219
- preferred server list**, 337
- principle names, 316
- private key**, 337
- Profiles, LDAP client, 210
- Project
  - attributes, 276
  - object class, 276
- proxy credentials, 215
- public key**, 337
- PWDIR, 142
- PWDIR/security/passwd.adjunct files, 167
- /PWDIR/shadow files, 144
- /PWDR/security/passwd.adjunct, 144

## R

`rcp`, 147, 178  
    NIS maps, transferring, 163  
`rdist`, NIS maps, transferring, 163  
**record**, 337  
Referrals, 233  
`replicas`, 306  
`resolv.conf` file, 56  
    examples, 89  
`resolv.conf` files, 68, 69, 83, 85  
    examples, 67  
    NIS and Internet, 86  
    setup, 67  
`resolve.conf` files, default domain  
    names, 56  
**resolver**, 56  
**resource record**, 107  
resource records (DNS), 93  
**reverse resolution**, 337  
RFC 2307  
    attributes, 268  
    object classes, 271  
`rlogin`, problems, 120  
**root domain**, 337  
**RPC**, 337  
`rpc.nisd` attributes, 294  
`rpc.nisd` configuration files, 292  
`rpc.yppasswdd`, 153, 154  
    4.x compatibility (NIS), 138  
    passwd updates maps, 162  
`rpc.yppasswdd` daemon, 128  
`rpc.yppupdated` daemon, 128  
`rsh`, problems, 121

## S

schema, Project, 276  
Schemas  
    directory user agent, 274  
    mail alias, 273  
    RFC 2307, 268  
**Secure RPC password**, 337  
`securenets` files, 136  
security  
    C2 security, NIS and, 167  
    NIS, 136, 141  
    NIS, and, 151

security (Continued)  
    NIS, C2 security and, 167  
    root in NIS maps, 151  
    `securenets` files, 136  
`sed`, 164  
**server**, 338  
server failed message (DNS), 119  
**server list**, 338  
servers  
    NIS, preparing, 141  
    NIS slave setup, 147  
    not available (NIS), 173  
    `yppservers` files, 166  
Service Search Descriptors, 208, 235  
setup, 53  
    DNS data files, 93  
    DNS example, 87  
    DNS server initialization, 75  
    DNS subdomains (different zones), 95  
    DNS subdomains (same zone), 94  
    DNS testing, 75  
    multiple NIS domains, 145  
    NIS, starting, 146  
    NIS clients, 149  
    NIS makefile, 143  
    NIS setup, preparation for, 140, 141  
    NIS slave servers, 147  
    switch files, 42  
shadow files, 144  
    NIS and, 137  
    Solaris 1.x formats, 152  
`sites.byname` files, maps, changing server  
    of, 157  
**slave server**, 338  
Solaris naming services, 29  
SSDs, 208  
**subnet**, 338  
Sun ONE Directory Server 5.1, setup using  
    `idsconfig`, 232  
Sun ONE server setup, load data into directory  
    server, 240  
`SUNWnskr`, 136  
`SUNWnsktu`, 136  
`SUNWypr`, 136  
`SUNWypu`, 136  
switch files  
    `nsswitch.files`, 41  
    `nsswitch.ldap`, 41

switch files (Continued)  
  nsswitch.nis, 40  
syslog, 118

## T

**table**, 338  
**TCP**, 338  
**TCP/IP**, 338  
timezone tables, 37  
/tmp/temp\_file files, 166  
**Transport Control Protocol**, 338  
Transport Layer Security, 214, 338

## U

Unknown field messages (DNS), 121  
unreachable messages (DNS), 120  
useradd, 152  
  password is locked, 152  
userdel, 153  
users  
  adding (NIS), 152  
  netgroups, 154, 155  
  NIS, 152  
  passwd maps, updating, 153  
  passwords (NIS), 153  
  useradd, 152  
  userdel (NIS), 153  
/usr/lib/netsvc/yp directories, 163  
/usr/lib/netsvc/yp/ypstart script, 142  
/usr/lib/netsvc/yp/ypstart script, 85  
  NIS security, 151  
/usr/sbin/makedbm, non-default maps,  
  modifying, 164

## V

/var/named/hosts.rev, 54  
/var/named/named.ca files, 54  
/var/spool/cron/crontabs/root files,  
  NIS, problems, 177  
/var/yp, 172  
/var/yp/, 130, 165  
/var/yp/ directory, 144

/var/yp/binding/ files, 173  
/var/yp directories, NIS security, 151  
/var/yp directory, 141, 143, 148  
/var/yp/Makefile, 145  
  maps, supported list, 157  
/var/yp/Makefile files, 4.x  
  compatibility, 137  
/var/yp/mymap.asc, 165  
/var/yp/nicknames files, 133  
/var/yp/securenets files, 136  
/var/yp/ypxfr.log files, 164

## W

**WAN**, 338

## X

**X.500**, 338

## Y

YP\_INTERDOMAIN key, 85  
ypbind, 133, 135, 146, 149, 156, 175  
  “can’t” messages, 171  
  client not bound, 173  
  fails, 174  
  slave servers, adding, 167  
ypbind “can’t” messages (NIS), 171  
ypbind daemon, 128  
ypcat, 45, 129, 132, 133  
ypinit, 129, 133, 143, 144, 145, 146, 147, 148,  
  149, 166  
  default maps, 161  
  slave servers, adding, 167  
ypmatch, 129, 133  
yppoll, 129  
yppush, 129, 133, 155, 157, 162  
  maps, changing server of, 157  
yppush in Makefile, 159  
yppush maps, NIS, problems, 178  
ypserv, 85, 133, 135, 175, 176  
  failure of, 178  
  multihome support, 137  
ypserv daemon, 128

- ypserve, 85, 146
- ypservers, 166
- ypservers files, slave server, adding, 166
- ypservers maps, NIS, problems, 178
- ypset, 129, 133
- ypstart, 136, 146
- ypstart files, 154
- ypstop, 136, 147, 167
- ypupdated daemon, 136
- ypwhich, 129, 132, 133, 136
  - display inconsistent, 174
- ypxfr, 129, 133, 165, 177
  - invoking directly, 164
  - logging, 164
  - logging output, 177
  - maps, changing server of, 156, 157
  - shell script, 177
  - shell scripts and, 163
- ypxfr\_1perday, 163
- ypxfr\_1perhour, 163
- ypxfr\_2perday, 163
- ypxfr daemon, 128
- ypxfr.log files, 164, 177

## **Z**

- zone expired messages (DNS), 120

