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

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 806–4077–06
December, 2001

011025@2471

Contents

Figures

Preface

Solaris Administration Guide: Naming and Directory Services describes the set up, configuration, and administration of the Solaris 9 operating environment naming and directory services: DNS, NIS, NIS+, FNS 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. Identify the audience.

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 and directory services:

Part I: About Naming and Directory Services

Part II: DNS Setup and Administration

Part III: NIS Setup and Configuration

Part IV: NIS Administration

Part V: NIS+ Setup and Configuration

Part VI: NIS+ Administration

Part VII: LDAP Setup and Administration

Appendix A: Transitioning from NIS to NIS+

Appendix B: Federated Naming Service

Appendix C: Error Messages (DNS, NIS, NIS+)

Related Books

- *DNS and Bind*, by Cricket Liu and Paul Albitz, (O' Reilly, 1992)
- *Managing FNS and NFS*, by Hal Stern, (O' Reilly, 1993)

Ordering Sun Documents

The Sun Software Shop stocks select manuals from Sun Microsystems, Inc. You can purchase individual printed manuals and AnswerBook2™ CDs.

For a list of documents and how to order them, visit the Software Shop at `http://www.sun.com/software/shop/`.

Accessing Sun Documentation Online

The docs.sun.com℠ 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`.

What Typographic Conventions Mean

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 `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*.
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. You must be *root* 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 of the book 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.

CHAPTER 1

Naming and Directory Services (Overview)

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

- 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.

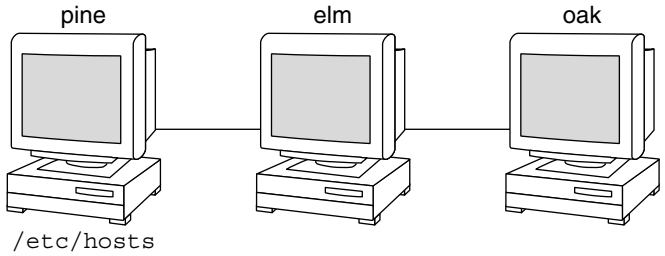
naming services are fundamental to any computing network. Among other features, a naming service provides functionality that:

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

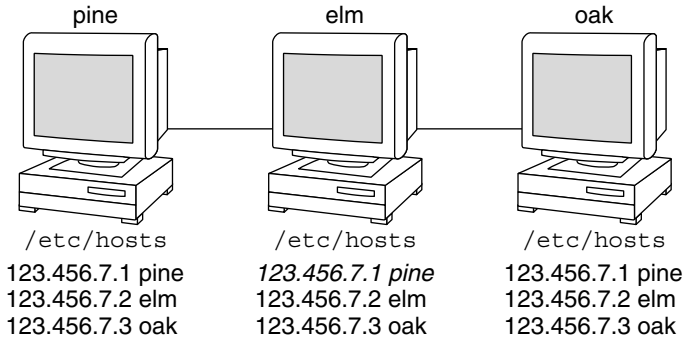
- Renames

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.



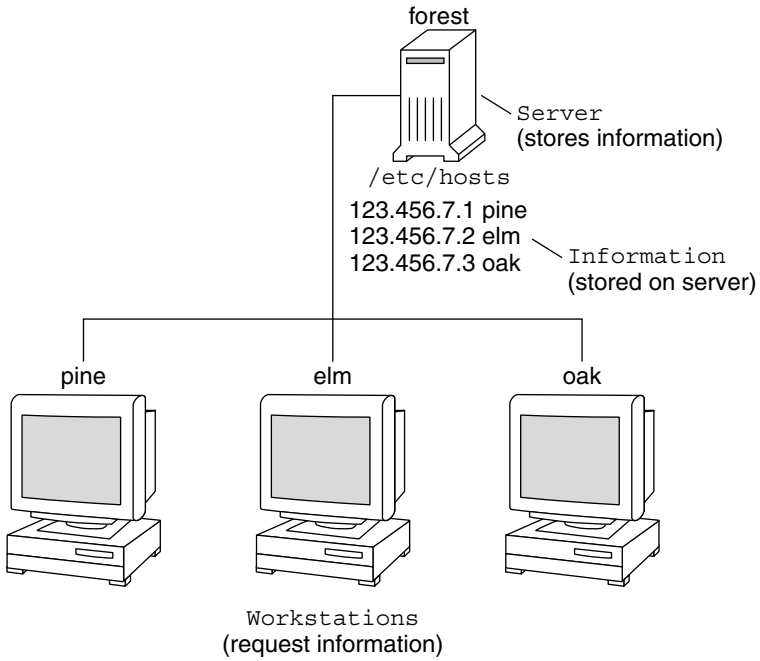
In addition to addresses, machines store security information, mail data, information about their Ethernet interfaces, network services, groups of users allowed to use the network, services offered on the network, and so on. As networks offer more services,

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.

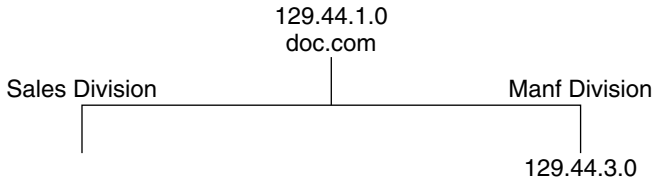
The machines are known as *clients* of the server. The following figure illustrates the client —server arrangement. Whenever information about the network changes, instead of updating each client's local file, an administrator updates only the information stored by the network information service. This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



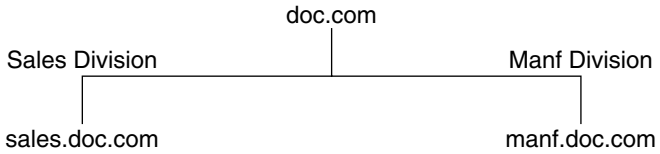
This arrangement, of a server providing centralized services to clients across a network, is known as *client-server computing*.

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.



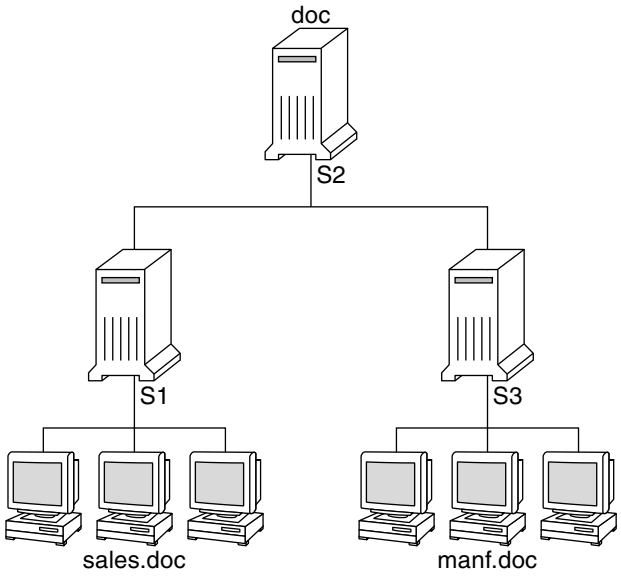
Each division could be identified by its network address, as shown above, but descriptive names made possible by naming services would be preferable.



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:

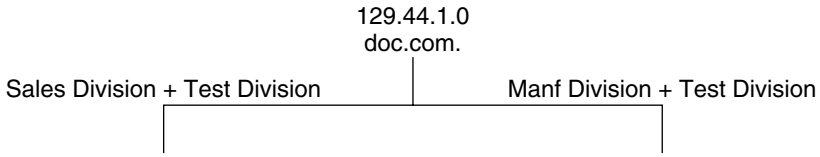


Clients C1, C2, and C3 would obtain their network information from server S1. Clients C4, C5, and C6 would obtain it from server S3. The resulting network is summarized in the following table. (The table is a generalized representation of that network but does not resemble an actual network information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

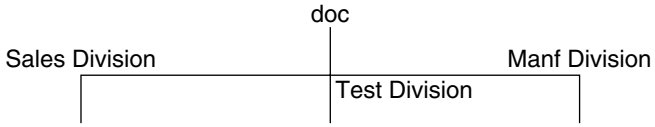
TABLE 1–1 Representation of doc.com Network

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

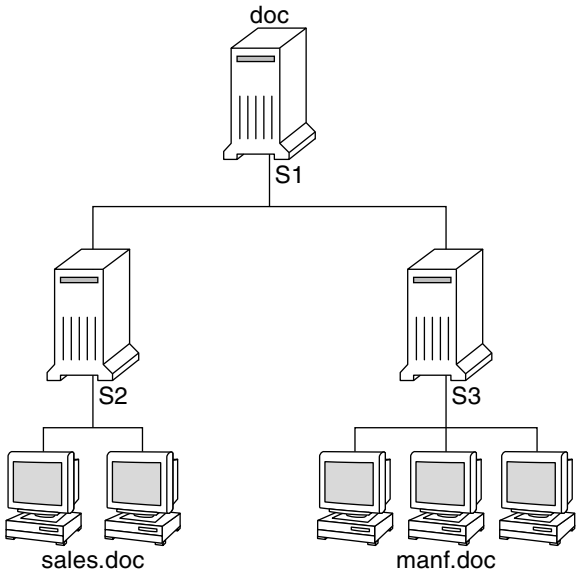
Now assume that you create a third division, Testing, which borrowed some resources from the other two divisions, but did not create a third subnet. The physical network would then no longer parallel the corporate structure:



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:



Now clients C1 and C2 would obtain their information from server S2; C3 and C4 from server S4; and C5 and C6 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 27).
- `/etc` files, the original UNIX™ naming system (see "`/etc` Files" on page 28).
- NIS, the *Network Information Service* (see "NIS" on page 28).
- 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, the *Domain Name System*, 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. See the Part II, DNS Setup and Administration for further details.

/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)NIS Maps: An Overviewfor 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 7.

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 new software program.

The NIS+ naming service is designed to conform to the shape of the organization that installs it, wrapping itself around the bulges and corners of almost any network configuration. Unlike NIS, the NIS+ name space 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.

Federated Naming Service

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

Solaris Directory Service

Solaris 9 supports LDAP (Lightweight Directory Access Protocol) in conjunction with the iPlanet Directory Server 5.x, as well as other LDAP Directory Servers.

The distinction between a Naming Service and a Directory Service lies in the differing extent of functionality. A directory service provides the same functionality of a naming service, but provides additional functionalities as well. Additional functionality includes:

- Multiple application compatibility
- Flexible [user] access control
- Incremental data pushes between master and replica servers

See Chapter 11 for more information about the above.

Naming and Directory Services: A Quick Comparison

	DNS	NIS	NIS+	FNS	LDAP
NAME-SPACE	hierarchical	flat	hierarchical	hierarchical	hierarchical
DATA STORAGE	files/resource records	bi-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
TRANS-PORT	TCP/IP	LAN	LAN	LAN	TCP/IP
Implement-ation—scale	Global	LAN	LAN	Global (w/ DNS)/LAN	Global

CHAPTER 2

The Name Service Switch

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`. 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:

- `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 36.

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 42 or "IPv6 and Solaris Naming Services" on page 42

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 36.) 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 41 .

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:

- `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 `/etc` directory. For example, `/etc/passwd`
`nisplus`	An NIS+ table. For example, the `hosts` table.
`nis`	An NIS map. For example, the `hosts` map.
`compat`	Compat can be used for password and group information to support old-style + or - syntax in `/etc/passwd`, `/etc/shadow`, and `/etc/group` 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 /etc 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:

- 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:

```
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).	# hosts: nisplus [NOTFOUND=return] files
Fully interpreted line.	hosts: nisplus [NOTFOUND=return] file
Partially interpreted line (the `files` element not interpreted)	hosts: nisplus [NOTFOUND=return] # files

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` 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:

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

Here 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.
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
```

EXAMPLE 2–1 NIS+ Switch File Template (`nsswitch.nisplus`) *(Continued)*

```
# 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 notuse
# 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
```

EXAMPLE 2–4 LDAP Switch File Template *(Continued)*

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

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 9release 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. **Log in to the client as 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

   ```
   client1# cd /etc
   client1# 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.

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 (FNS and NIS+)*

IPv6 and Solaris Naming Services

DNS and LDAP are IPv6 "compatible" in the sense that one can store IPv6 addresses in DNS. However, as of Solaris 9, one cannot use an IPv6 transport for client-server DNS or LDAP traffic.

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 "IPv6 (Overview)" in *System Administration Guide: IP Services* and "Transitioning From IPv4 to IPv6 (Reference)" in *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:

```
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.

Enabling a Client to Use IPv6

TABLE 2–5 Enabling a Machine to Use IPv6

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

▼ How to Enable a Client to Use IPv6

1. **Log in as 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.

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 this:

```
passwd: files nisplus
```

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

```
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.

CHAPTER 3

Introduction to DNS (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.

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.

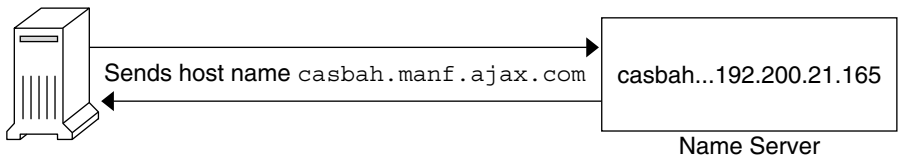


FIGURE 3–1 Name to Address Resolution

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 59. These hierarchies resemble UNIX™ file systems, in that they are organized into an upsidedown tree structure.

Each root name server maintains the host names and IP address 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.

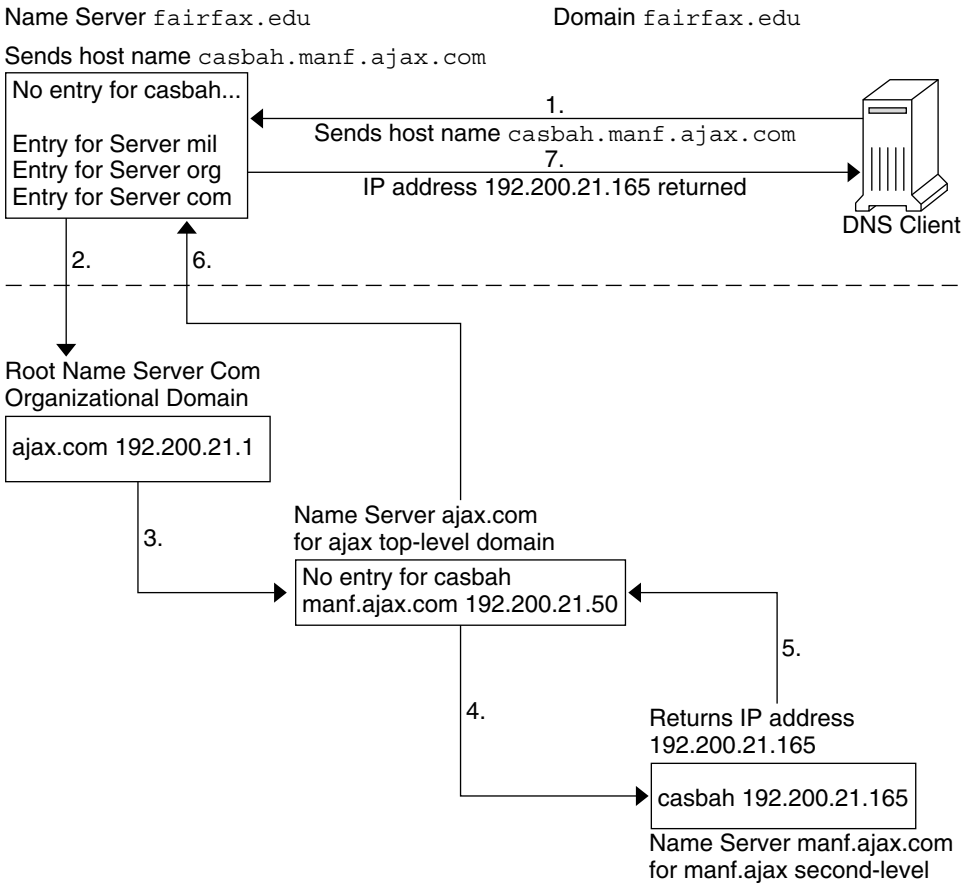


FIGURE 3–2 Name to Address Resolution for a Remote Host

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 60 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 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. The "Implementing DNS" on page 89 section 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 101for additional information on the `named.conf` file.)

Names of DNS Data Files

So long as 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* 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 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:

- `/var/named/named.ca`. (See "The `named.ca` File" on page 104 for additional information on the `named.ca` file.) So long as you are internally consistent, you can name this file anything you want.
- `/var/named/hosts`. (See "The `hosts` File" on page 106 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 108 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 109 and for additional information on the `named.local` file.) So long as you are internally consistent, you can name this file anything you want.

`$INCLUDE` *Files*

An include file is any file named in a `$INCLUDE()` statement in a DNS data file. `$INCLUDE` files can be used to separate different types of data into multiple files for your convenience. See "`$INCLUDE` Files" on page 109

For reference purposes, Table 3–1 compares BIND file names from these three 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.

TABLE 3–1 File Name Examples *(Continued)*

Solaris Names	O'Reilly Names or other names	U.C. Berkeley Names	Content and Purpose of File
`$INCLUDE` files, same convention for all three sources			Any file identified by an `$INCLUDE()` 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; rather, 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.

resolv.conf File

For a detailed description of what the`resolv.conf` file does, please refer to the `resolv.conf` man page.

The following discussion describes 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 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.

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

TABLE 3–2 `named.conf` Statements

`acl`	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 `acl` statement before it can be used elsewhere; no forward references allowed.
`include`	Inserts an include file at the point where the `include` statement is encountered. Use `include` to break up the configuration into more easily managed chunks.

TABLE 3–2 `named.conf` Statements *(Continued)*

`key`	Specifies a key ID used for authentication and authorization on a particular name server. See the `server` statement.
`logging`	Specifies what information the server logs and the destination of log messages.
`options`	Controls global server configuration options and sets default values for other statements.
`server`	Sets designated configuration options associated with a remote name server. Selectively applies options on a per-server basis, rather than to all servers.
`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;
```

EXAMPLE 3–1 Example Master Configuration File for a master Server *(Continued)*

```
        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;
        };
};
```

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 sub-domains, `sales.ajax.com`, `test.ajax.com`, and `manf.ajax.com`.

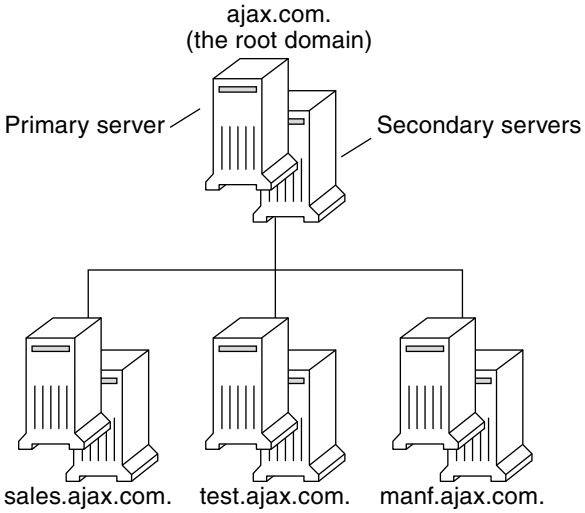


FIGURE 3–3 Hierarchy of DNS Domains in a Single Organization

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 Figure 3–3is, conceptually, 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 Figure 3–3is one of a number of top-level organizational domains in existence on the Internet.

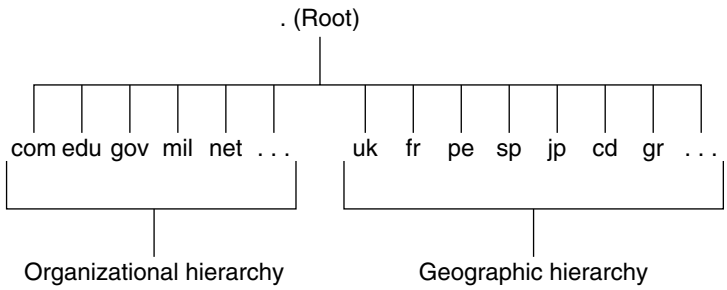


FIGURE 3–4 Hierarchy of Internet Domains

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, you have to:

- 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. In the United States, InterNIC is the company that currently handles network address and domain registration matters.
- 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`.

Figure 3–5 shows the position of the `ajax.com` domain in the DNS namespace on the Internet.

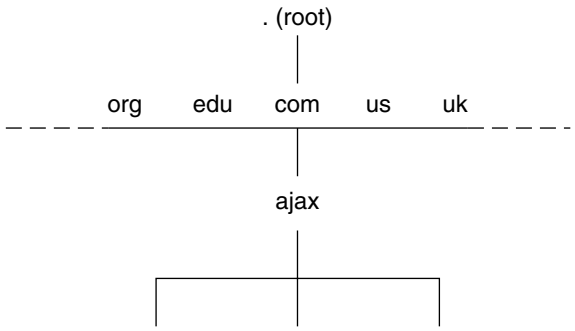


FIGURE 3–5 Ajax Domain's Position in the DNS Namespace

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

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 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 the name.

Zones

DNS service for a domain is managed on the set of name servers. Name servers can manage a single domain, 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*; thus, 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.

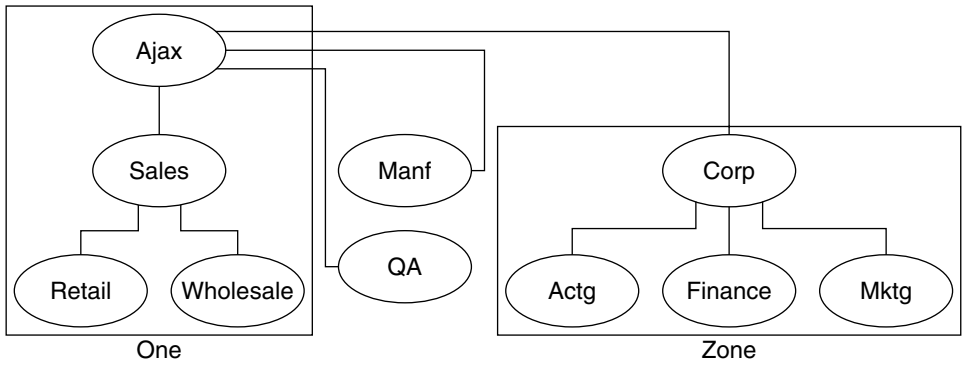


FIGURE 3–6 Domains and Zones

For example, the `Ajax` domain shown in Figure 3–6 contains a top domain (`Ajax`), four subdomains, and five sub-subdomains. It is divided into four zones shown by the thick lines. 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 include 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.

CHAPTER 4

Administering DNS (Tasks)

This chapter describes how to administer the Domain Name System (DNS). For more detailed information, see *DNS and Bind* by Cricket Liu and Paul Albitz, (O'Reilly, 1992) and "Name Server Operations Guide for BIND", University of California, Berkeley.

TABLE 4–1 DNS Administration Task Map

Task	For instructions
Set up the resolv.conf file	
Set up a DNS client.	
Set up a DNS server.	
Add +/- syntax compatibility.	
Set up trailing dot domain names.	
Test a DNS installation	
Modify DNS data files.	
Add a machine to DNS network.	
Delete a machine from a DNS network.	
Add additional DNS servers.	
Plan and set up a DNS subdomain.	
Migrate from BIND version 4.9.x to BIND version 8.2.2	
Enable DNS forwarding on a NIS client.	

Setting Up the `resolv.conf` File

A simple example `resolv.conf` 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 enter a hard carriage 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*

Where *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 — Task Map

TABLE 4–2 Configuring a network for DNS

Task	Go To
1. Set up all client machines.	See "Setting Up a DNS Client" on page 67
2. Set up the DNS server.	See "Setting Up a DNS Server" on page 69

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 in Example 4–2:

   EXAMPLE 4–2 Sample `resolv.conf` File

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

```
nameserver IP_address
```

Where *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
```

Where *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. **Log in as 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: nisplus dns [NOTFOUND=return] files
      ```

      or

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

Do *not* use the above syntax for NIS clients, since 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 the machine.

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

Setting Up a DNS Server

▼ How to Set Up a DNS Server

1. **Login as superuser.**
2. **Set the server up as a DNS client (this includes setting up the server's `resolv.conf` file). SeeExample 4–2.**
3. **Set up the boot file. See "Example Boot Files" on page 90.**
4. **Set up the data files See"The `named.ca` File" on page 104 . You need to set up four data files:**
   a. **The `named.ca` file. See"The `named.ca` File" on page 104 .**
   b. **The `hosts` file. See "The `hosts` File" on page 106.**
   c. **The `hosts.rev` file. See "The `hosts.rev` File" on page 108.**
   d. **The `named.local` file. See "The `named.local` File" on page 109.**
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 global Internet. In order 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.

Specifying a Master Server

The two types of master server are:

- *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 it 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`), 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 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
```

Specifying 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 the slave server starts `in.named`, 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.

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 multihome master server.
- *Backup file*. After the IP address of the master (and optional slave) server(s), 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

---

Specifying a Cache-Only Server

All servers are caching servers in the sense that they all maintain a cache of DNS data. A caching-only 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.

Example 4–3 is a sample boot file for a caching-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:

- 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

Specifying a Stub Server

You can set up a *stub server* that is not authoritative for any zone. A stub server is a server that is not a master server for any zone other than the `in-addr.arpa.` domain. A stub server handles the same kind of queries from clients that authoritative name servers perform. But the stub server does not maintain any authoritative data itself.

A stub server requires less memory than an authoritative server, but cannot function by itself if no master or slave servers are available.

DNS and +/- Syntax

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

Adding Compatibility With +/- Syntax-Task Map

TABLE 4–3 Adding Compatibility With +/- Syntax

Task	Description	For Instructions, Go To
Adding Compatibility With +/- Syntax	Modify the `/etc/passwd`, `/etc/shadow`, and `/etc/group` files to add DNS compatibility with +/- syntax.	"How to Add DNS Compatibility With +/- Syntax" on page 74

Security Considerations

You must perform this operation as superuser.

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 that indicate the table is empty, even if it has entries.

▼ How to Add DNS Compatibility With +/- Syntax

1. **Log in as 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** `/etc/passwd`**,** `/etc/shadow`**, and** `/etc/group` **files.**

Caution – 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 machine.**

   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.

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.
- Do not use a trailing dot in domain names in `named.conf` or `resolv.conf` files. For example, `sales.doc.com` is correct.

Initializing the Server

To initialize a server:

1. **Log in as a 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 as follows:

1. **Check your `syslog` file for error messages.**

   See "DNS Problems and Solutions" on page 121for common DNS error messages and troubleshooting information.

2. **Look up a host name in the local domain with `nslookup`.**

```
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 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.

3. **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:

```
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, your answer will be returned as "`non-authoritative`." This is normal because the answer is now coming from your cache, not the remote name server.)

4. **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: First the

name of the host you are searching for, and second, the name of the name server you are testing:

```
remotemachine9% nslookup altair remotemaster.foo.org.
 Server:  remotemaster.foo.org
 Address: 123.231.12.22
 Name:  altair.doc.com
 Addresses: 111.22.3.4
```

- 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).

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:

- 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 77.
- Inform `in.named` on the master server that it should reread the data files and update its internal database (see "How to Force `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 synch.

A typical SOA record of a sample `hosts` file looks like this:

```
; 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`.

How to Force `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, enter:

```
# 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:

▼ How to Add a Client

1. **Log in as a 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 42.)
4. **Add an address (**`A`**) record for** `rigel` **to the master server's** `hosts` **file.**
   For example:

   ```
   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:

   - 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 enter:

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

Removing a Client

To remove a client from a DNS domain:

▼ How to Remove a Client

1. **Log in as a 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.

Adding Additional DNS Servers

You can add master and slave servers to your network. To add a DNS server:

▼ How to Add an Additional Server

1. **Log in as a superuser.**
2. **Set the server up as a DNS client.**
   See "Adding a Client" on page 78.
3. **Set up the server's boot file.**
4. **Set up the server's `named.ca` file.**
5. **Set up the server's `hosts` file.**
6. **Set up the server's `hosts.rev` file.**
7. **Set up the server's `named.local` file.**
8. **Initialize the server.**
9. **Test the server.**
   These steps are explained in more detail in "Setting Up a DNS Server" on page 69

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 98 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

Here are some points to consider before dividing your network into a parent and one or more subdomains:

- *How many 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.
- *How to divide your network?* 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, etc.); 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.

- *Consider the future.* 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+ naming service.* 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. Keep in mind 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, follow these steps to set it up:

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.**

   Install the following files on each 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.**

   This requires deleting the `A` records for the machines that are now in the new subdomain from the hosts files of the old domain's servers, and also deleting 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 51.

Solaris 9 DNS BIND 8.2.2 Implementation

For your convenience, the Solaris operating environment supplies a compiled version of Berkeley Internet Name Domain (BIND) version 8.2.2. 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 since because doing so would remove implicit search lists.
- *Inverse Queries*. Enabled because SunOS 4.x `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 8.2.2

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 8.2.2 `named.conf` file.**

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

The `nsswitch.conf` file controls DNS forwarding and Interent access for clients. NIS lients have implicit forwarding capabilities. NIS+ clients do not. See the*System Administration Guide: Naming and Directory Services (FNS and NIS+)*for how to set up NIS+ forwarding.

▼ To enable DNS forwarding capabilities on an NIS+ client:

1. **Login as 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.)

   However, if your NIS server is pre-Solaris 7 or one or more of your NIS servers is lacking a /etc/resolve.conf, do the following to set up DNS forwarding.

▼ To enable DNS forwarding capabilities on an [older] NIS client:

1. **Login as 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:**

   ```
   #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. **Run make to 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 the Solaris Release 2 or higher, make sure that the `YP_INTERDOMAIN` key is present in the host maps. In addtion, 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 the SunOS operating environment or later.” The command `makedbm -b` is a reference to the “ `-B` ” variable in the `Makefile`

TABLE 4–4 NIS/DNS in Heterogeneous NIS Domains

SLAVE	MASTER		
	4.0.3+	Solaris NIS	
4.0.3+	Master: `makedbm -b` Slave: `ypxfr`	Master: `makedbm -b` Slave: `ypxfr -b`	Master: `ypserv -d` Slave: `ypxfr -b`

TABLE 4–4 NIS/DNS in Heterogeneous NIS Domains *(Continued)*

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

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

CHAPTER 5

DNS Administrtaion (Reference)

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 practica example assumes:

- 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`

TABLE 5–1 Example Network Domain and Zone Configuration *(Continued)*

Name and Zone	Number
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 `dnsmastr` Server

```
; named.boot file on the dnsmastr (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
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
```

EXAMPLE 5–2 Example Boot File for `dnssales` Server *(Continued)*

```
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 `dnsmastr` 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.   IN  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
                           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 `dnsmastr` 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. (
                           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.
```

EXAMPLE 5–12 Example `named.ca` File *(Continued)*

```
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
```

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 inTable 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
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 easiest 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.

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.

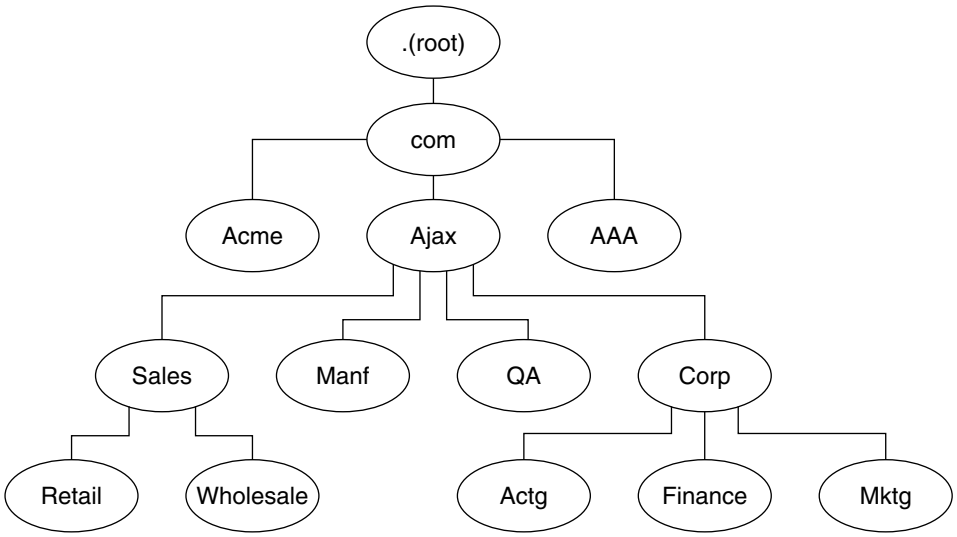


FIGURE 5–1 Domains and Subdomains

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. For example, `Mktg.Corp.Ajax.Com.` from .

How DNS Affects Mail Delivery

In addition address mapping and maps addresses to host names, as discussed in “Name-to-Address Resolution” on page 48, 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 8.2.2 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:

- 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 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

`acl`	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 `acl` statement before it can be used elsewhere; no forward references allowed.
`include`	Inserts an include file at the point where the `include` statement is encountered. Use `include` to break up the configuration into more easily managed chunks.
`key`	Specifies a key ID used for authentication and authorization on a particular name server. See the `server` statement.
`logging`	Specifies the information the server logs and the destination of log messages.
`options`	Controls global server configuration options and sets default values for other statements.
`server`	Sets designated configuration options associated with a remote name server. Selectively applies options on a per-server basis, rather than to all servers.
`zone`	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";
```

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

```
        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" {
        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:

- 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 this:

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
```

See for a more complete discussion of setting up a domain that is not connected to the Internet.

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 106.

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 these 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 95 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 *ttl* 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 *ttl* defaults to the minimum time specified in the Start-Of-Authority (SOA) resource record.

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

Most *ttl* 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 *ttl* values to reflect that frequency. Also, if you have some *ttl* 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 *ttl* 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 *ttl* 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 95.

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.

TABLE 5–6 Special Resource Record Characters *(Continued)*

Character	Definition
`..`	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

SOA— Start of Authority

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 Start-Of-Authority 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 *ttl* 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
                             7200         ;Refresh
                             3600         ;Retry
                             432000         ;Expire
                             86400)         ;Minimum             )
```

NS—Name Server

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.
```

A—Address

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 address (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
```

HINFO—Host Information

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 host-information resource record (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.

WKS—Well-Known Services

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 Well-Known Services (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.

CNAME—Canonical Name

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 Canonical-Name Resource record (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
```

PTR—Pointer Record

Example 5–31 shows the syntax for a pointer (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, PTR's 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.
```

MX—Mail Exchanger

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 mail-exchanger 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     0    Seismo.CSS.GOV.
foo.com.         IN        MX     10    RELAY.CS.NET.
*.foo.com.         IN        MX     20    RELAY.CS.NET.
```

CHAPTER 6

DNS Troubleshooting (Reference)

DNS Problems and Solutions

This section describes 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` *nnnn* `restarted` or `named` *nnn* `reloading` `nameserver` 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 "How to Force `in.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 lies 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 `129.146.168.119`, you would enter:

```
% ping 129.146.168.119 -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 `129.146.168.46`, 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 "How to Force `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 "How to Force `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 "Trailing Dots in Domain Names" on page 75.

PART III NIS Setup and Configuration

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

CHAPTER 7

Network Information Service (NIS): An Overview

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

NIS Introduction

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.

By running the service, the system administrator can distribute administrative databases, called *maps*, among a variety of servers (*master* and *slaves*), and 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.

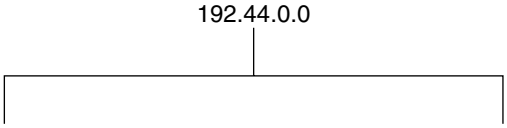
NIS was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network 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 has *host* names or *machine* names. This discussion uses *machine*, but some screen messages or NIS map names might use *host* or *machine*.

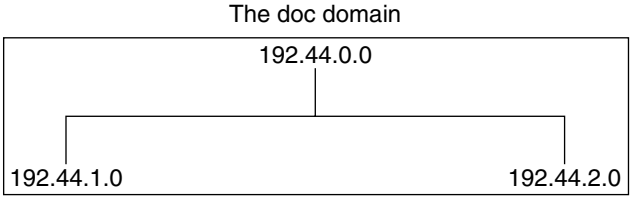
NIS Architecture

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, The Name Service Switch, 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 132)
- Maps (see "NIS Maps" on page 133)
- Daemons (see "NIS Daemons" on page 132)
- Utilities (see "NIS Utilities" on page 132)
- NIS Command Set (see "Summary of NIS-Related Commands" on page 137)

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
`ypserv`	Server process
`ypbind`	Binding process
`ypxfr`	High speed map transfer
`rpc.yppasswdd`	NIS password update daemon
`rpc.ypupdated`	Modifies other maps such as `publickey`

NIS Utilities

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

TABLE 7–2 NIS Utilities

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

NIS Maps

The information in NIS maps is stored in `ndbm` format. The `ypfiles` and `ndbm` man pages 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 value 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
`bootparams`	`bootparams`	Contains path names of files clients need during boot: root, swap, possibly others.
`ethers.byaddr`	`ethers`	Contains machine names and Ethernet addresses. The Ethernet address is the key in the map.
`ethers.byname`	`ethers`	Same as `ethers.byaddr`, except the key is machine name instead of the Ethernet address.
`group.bygid`	`group`	Contains group security information with group ID as key.
`group.byname`	`group`	Contains group security information with group name as key.
`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.

TABLE 7–3 NIS Map Descriptions *(Continued)*

Map Name	Corresponding NIS Admin File	Description
`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 the key.
`networks.byaddr`	`networks`	Contains names of networks known to your system and their IP addresses, with the address as the 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.
`protocols.bynumber`	`protocols`	Same as `protocols.byname`, except that key is protocol number.
`rpc.bynumber`	`rpc`	Contains program number and name of RPCs known to your system. Key is RPC program number.
`services.byname`	`services`	Lists Internet services known to your network. Key is port or protocol.

TABLE 7–3 NIS Map Descriptions *(Continued)*

Map Name	Corresponding NIS Admin File	Description
`services.byservice`	`services`	Lists Internet services known to your network. Key is service name.
`ypservers`	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 the `ipnodes(4)` man page. 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:

```
% 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.

Summary of NIS-Related Commands

The NIS service includes specialized daemons, system programs, and commands, which are summarized in Table 7–4. Refer to their man pages for details about how to use them.

TABLE 7–4 NIS Command Summary

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

TABLE 7–4 NIS Command Summary *(Continued)*

Command	Description
`ypxfrd`	Provides map transfers service for `ypxfr` requests (generally slave servers). It is run only on the master server.
`yppush`	Copies a new version of an NIS map from the NIS master server to its slaves. You run it on the master NIS server.
`ypset`	Tells a `ypbind` 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 ypset(1M) and ypbind(1M) man pages for information about the `ypset` and `ypsetme` options to the `ypbind` process.
`yppoll`	Tells which version of an NIS map is running on a server that you specify. It also lists the master server for the map.
`ypcat`	Displays the contents of an NIS map.
`ypmatch`	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.
`ypwhich`	Shows which NIS server a client is using at the moment for NIS services, or, if invoked with the `-m` *mapname* option, which NIS server is master of each of the maps. If only `-m` 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.

---

1. `ypbind` initiates binding to the first server that responds to the broadcast.
2. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
3. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
4. `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 Between Solaris Release 2.6 NIS and Earlier NIS Versions

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

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 Release 2.6 and there is no 2.6 Release NSKit.

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

NIS service is no longer started with the `/etc/init.d/yp` script which no longer exists. With the Solaris 2.6 Release, 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 the securenets man page.

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

```
255.255.255.0    13.13.13.255
host    13.13.14.1
host    13.13.14.2
```

where 255.255.255.0 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.

Sun Operating Environment 4.X Compatibility Mode

Solaris operating environment NIS supports password configuration files in both the Sun Operating Environment 4.x (Solaris release 1) password format and the Solaris Release 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 Release 2 mode. If this file does not exist, NIS operates in the SunOS 4.x mode.

In the SunOS 4.x mode, all password information is kept in the `passwd` file. In the Solaris Release 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 Release 2 mode because of the Solaris Release 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.x format or in the Solaris Release 2 format. The `rpc.yppasswdd` daemon understands both password formats. The Solaris Release 2 format is recommended.

CHAPTER 8

Setting Up and Configuring NIS Service

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

Before You Begin Configuring NIS

Before configuring your NIS namespace, you must:

- 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(s). 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(s).

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 strictly necessary.

NIS Configuration Steps

After your Solaris operating environment software and `nsswitch.conf` files are installed, and your domain planned, you must perform the following steps to configure NIS:

1. Prepare the master server (see "Preparing the Master Server" on page 145).
2. Configure the NIS master server (see "How to Set Up the Master Server With `ypinit`" on page 148).
3. Start the NIS daemons on the master server (see "Starting NIS Service on the Master Server" on page 150).
4. Configure your slave servers (see "Setting Up NIS Slave Servers" on page 151).
5. Configure NIS client machines (see "Setting Up NIS Clients" on page 153).

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*.

The following sections explain these steps in detail.

Preparing the Master Server

Setting up the master server involves converting the source (input) text files on the master into NIS master server maps. Before doing this, however, you need to take several precautions.

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` might be 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 would 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 145 for additional information.

However, if you choose to locate 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` from *=/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.x `passwd` file format, this implementation of NIS accepts the Solaris 7 release `/etc/passwd` and `/etc/shadow` file format 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`.

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.

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

Preparing the Master Server — Task Map

TABLE 8–1 Preparing the Master Server

Task	Description	For Instructions, Go To
Preparing the Master Server	Convert files to NIS maps	"How To Prepare Source Files for Conversion to NIS Maps" on page 146
Preparing the Master Server	Set up with `ypinit`	"How to Set Up the Master Server With `ypinit`" on page 148

▼ How To Prepare Source Files for Conversion to NIS Maps

Prepare the source files for conversion to NIS maps.

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

2. **Copy all of these source files, except** `passwd`**, to the** `DIR` **directory that you have selected.**

3. **Copy the** `passwd` **file to the** `PWDIR` **directory that you have selected.**

4. **Copy** `audit_user`**,** `auth_attr`**,** `exec_attr` **and** `prof_attr` **to the selected** `RBACDIR` **directory**

5. **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 the `/etc/mail/aliases` source file is complete by verifying that it contains all the mail aliases that you want to have available throughout the domain. Refer to the aliases man page for more information.

6. **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.)

7. **Check to 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 148.

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 145, 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 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 /usr/sbin/ypinit shell 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, follow these steps:

1. **Log in as a 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. **To build new maps on the master server, type:**

   ```
   # /usr/sbin/ypinit -m
   ```

5. `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. `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 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. You must answer `yes` to install the new version of NIS.

8. **After** `ypinit` **has constructed the list of servers, it invokes** `make`**.**

   ```
   # 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 and runs `makedbm` on them, 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:**

```
# 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 modify the steps slightly, as described in the section above, 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 section above 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
```

Where *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 `password` 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 148.)

Starting and Stopping NIS From the Command Line

To begin NIS service from the command line, run the `/usr/lib/netsvc/yp/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` command:

```
#/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 NIS Slave Servers — Task Map

TABLE 8–2 Setting Up NIS Slave Servers

Task	Description	For Instructions, Go To
Setting Up NIS Slave Servers	Set up NIS slave servers	"Setting Up a Slave Server" on page 152

▼ Setting Up a Slave Server

Now you are ready to create a slave server, as follows:

1. **Log in as a 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.**
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).

   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 153 for details.)

5. **To determine if** `ypbind` **is running, type:**

   ```
   # ps -ef | grep ypbind
   ```

   If a listing is displayed, `ypbind` is running.

6. **If `ypbind` is running, stop it by typing:**

   ```
   # /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.

Starting NIS Service on a Slave Server

Now you can start daemons on the slave server and begin NIS service. All existing `yp` processes must be stopped, by typing:

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

To start `ypserv` on the slave server and run `ypbind`, type:

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

Alternatively, you can reboot the slave server and daemons will be started automatically.

Setting Up NIS Clients

TABLE 8–3 NIS Client Set Up

Task	Go To
Select the correct `nsswitch.conf` file.	See Chapter 2
Configure the Client to use NIS	See below.

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 net 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 log in 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 one, it binds to it. This search is referred to as *broadcasting*. If there is no NIS server on the client's local subnet, it fails to bind and the client machine is not able to obtain namespace data from the NIS service.

PART IV NIS Administration

This part describes the administration and troubleshooting of NIS naming service in the Solaris operating environment

CHAPTER 9

Administering NIS

This chapter describes how to administer NIS.

Password Files and Namespace Security

For security reasons:

- 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.x version `passwd` file format, this implementation of NIS accepts the Solaris Release 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

To add a new NIS user:

1. **Log in as a superuser on the master NIS server.**
2. **Create the new user's login ID with the `useradd` command.**

   For Solaris Release 2 systems, type the following:

   ```
   # useradd userID
   ```

   Where *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 in the form:

   ```
   # 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 157 for additional information on this matter.)

   For example, if you added the new user `baruch`, the line from `/etc/passwd` that you would copy to your passwd input file would look like:

   ```
   baruch:x:123:10:User baruch:/home/baruch:/bin/csh:
   ```

   The line for `baruch` that you would copy from `/etc/shadow` would look like:

   ```
   baruch:W12345GkHic:6445::::::
   ```

Note – If you are using a Solaris Release 1 `passwd` file format as input for your NIS maps, you must use a text editor to add the new user to your `passwd` file, manually.

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, it is not good practice to maintain 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 `baruch` from the master server's `/etc` files, you would enter:

   ```
   # userdel baruch
   ```

   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 baruch
   # 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.

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.yppasswd`, 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.yppasswd` 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:

- 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 the `netgroup` man page 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 and/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 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 with each element separated by a comma. 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.

For example, below 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 the netgroup man page for detailed information on using netgroups and netgroup input file syntax.

Working With NIS Maps

The following sections describe how to administer 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:

```
% ypcat mapname
```

To list both the keys and the values (if any) in a map, type:

```
% 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:

```
% ypwhich -m
```

To list the master server for a particular map, type:

```
% ypwhich -m mapname
```

To match a key with an entry in a map, type:

```
% ypmatch key mapname
```

If the item you are looking for is not a key in a map, type:

```
% ypcat mapname | grep item
```

Where *item* is the information you are searching for. 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, it causes `ypbind` to consult the `/var/yp/binding/`*domainname*`/ypservers` file for a list of servers for that domain. If this file doesn't 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.

Here are instructions for remaking a sample NIS map called `sites.byname`.

1. **Log in to the new master as superuser and type:**

   ```
   newmaster# cd /var/yp
   ```

2. `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.**

3. **To update or remake the map, type:**

   ```
   newmaster# make sites.byname
   ```

4. **If the old master remains an NIS server, remote log in (**`rlogin`**) to the old master and edit** `Makefile`**. Comment out the section of the** `Makefile` **that made** `sites.byname` **so that it is no longer made there.**

5. **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. However, 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:

   ```
   oldmaster# /usr/lib/netsvc/yp/ypxfr -h newmaster sites.byname
   ```

   Now it is safe to run `yppush`. The 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 try this cumbersome but sure-fire option: log in as root on each NIS server and execute the `ypxfr` 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
- Add or delete `/etc/resolv.conf` to allow or deny DNS forwarding
- Add or delete `$PWDIR/security/passwd.adjunct` to allow or deny C2 security. (`$PWDIR` is defined in `/var/yp/Makefile`.)

To modify any of the listed files:

1. **Stop the NIS server by typing;**

   ```
   # /etc/init.d/yp stop
   ```

2. **Make the necessary changes to your files.**

3. **Restart the NIS server by typing:**

   ```
   # /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 points:

   - 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, and it is strongly recommended that you 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` man page 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, and a missing tab can invalidate an entry that is otherwise well formed.

To add an entry to the `Makefile`, do the following:

- Add the name of the database to the `all` rule

- Write the `time` rule
- Add 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:

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) - $(YPDBDIR)/$(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, and feeds the output to

The same procedure should be followed for all other automounter maps such as `auto_home`, or any other nondefault maps.

5. **Run** `make`**.**

   ```
   # make name
   ```

   Where *name* is the name of the map you want to make. For example, `auto_direct`.

   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 the value of `DIR` from `DIR=/etc` to `DIR=/var/etc/`*domainname*. You should also change the value of `PWDIR` from `PWDIR=/etc` to `PWDIR=/var/etc/`*domainname*.

The variables are:

- *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. Remember that most NIS commands use the current machine's domain which is set in the machine's `/etc/defaultdomain` file.

Updating 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. On the other hand, the `auto_master` map changes little, if at all.

As mentioned in "Default NIS Maps" on page 134, 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 it is a default map or not.

- 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.

Use the following procedure for updating 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.

Modifying Nondefault Maps

To update a nondefault map, you must:

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 164.
   - Use the `/usr/sbin/makedbm` program. (The `makedbm` man page 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

To create new maps, you can use one of two possible procedures: the first method uses an existing text file as input; the second uses standard input.

Creating 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# makedbm - homedomain/mymapkey1 value1  key2 value2  key3 value3
ypmaster#
```

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:

```
# 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.

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.yppasswdd` 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 n 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:

```
# 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.

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 new NIS server:

1. **Log in to the master server as a superuser.**
2. **Change to the NIS domain directory by typing:**

   ```
   # cd /var/yp/domainname
   ```

3. **Disassemble the `ypservers` file, as follows:**

   ```
   # 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:**

   ```
   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, log in to the new NIS slave as superuser and run the `ypinit` and `ypbind` commands:

   ```
   slave3# cd /var/yp
   slave3# ypinit -c list of servers
   slave3# /usr/lib/netsvc/yp/ypbind
   ```

8. **To initialize this machine as a slave, type the following:**

   ```
   # /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/up/ypstop
   ```

10. **Run `ypstart` to start NIS slave service.**

    ```
    #/usr/lib/netsvc/up/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:

```
# 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:

1. **Log into the machine as a 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 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 automactically 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 Release 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: `makedbm –b`	Master: `makedbm –b`	Master: `ypserv –d`
	Slave: `ypxfr`	Slave: `ypxfr`	`ypxrf –b`
Solaris NIS	Master: `makedbm –b`	Master: `makedbm –b`	Master: `ypserv –d`
	Slave: `ypxfr`	Slave: `ypxfr`	Slave: ypxfr with `resolv.conf` or `ypxfr –b`

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:

```
% /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.

CHAPTER 10

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:

- Messages saying that `ypbind` can't find or communicate with a server.
- Messages saying server not responding.
- Messages saying 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 182).

`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.

```
Client#7 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.

```
Client#7 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, `ypbind` 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 `ypbind`, 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, `ypbind` 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 `yppinit -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 ties up 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 `yppinit -c` or `ypinit -s` as described "Working With NIS Maps" on page 161.
- 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 stabilizes 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:

```
% 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:

```
% 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 178). If many NIS clients are failing to bind properly, the problem probably exists on one or more of the NIS servers.

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 `ypserv`, 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:

```
# 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 `ypserv` daemons are not running, kill them and then restart them by entering:

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

If both the `ypserv` and `ypbind` processes are running on the NIS server, type:

```
# ypwhich
```

If ypwhich does not respond, ypserv has probably hung and should be restarted. While logged in as root on the server, kill ypserv and restart it by typing:

```
# /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 add 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 183).
- Check the control files (see "Check the crontab File and ypxfr Shell Script" on page 184).
- Check the ypservers map on the master (see "Check the ypservers Map" on page 184).

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:

```
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. For instance, here is how you might transfer the problem map:

```
ypslave# rcp ypmaster:/var/yp/mydomain/map.\* /var/yp/mydomain
```

Here 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 181. 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 ypserver
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:

```
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, deregister the service from `rpcbind` before restarting `ypserv`. To deregister the service from `rpcbind`, on the server type:

```
# rpcinfo -d number 1
# rpcinfo -d number 2
```

Where *number* is the ID number reported by rpcinfo (`100004`, in the example above).

PART V LDAP Setup and Administration

This part provides an overview of the LDAP directory service. Additionally, it covers the setup, configuration, administration and troubleshooting of LDAP in the Solaris operating environment, with a focus on the use of the iPlanet Directory Server 5.1.

CHAPTER 11

Introduction to LDAP (Overview/Reference)

The LDAP chapters describe how to set up a Solaris client network to support the LDAP Directory Service.

- Introducing LDAP as a directory service (See "LDAP Model" on page 189
- LDAP Servers
  1. Directory Information Tree (DIT) . See "The Directory Information Tree" on page 192
  2. Virtual List Views (VLVs) . See "Browsing Indexes (Virtual List Views)" on page 194
  3. Schemas. See "Schemas" on page 192
  4. Client Profile Attributes. See "Client Profile Attributes" on page 194
  5. Indexing. See "Browsing Indexes (Virtual List Views)" on page 194
  6. Command Line Tools. See "Command Line Tools" on page 196
  7. LDAP Data Interchange Format. See "LDAP Data Interchange Format" on page 197
- LDAP Clients. See "LDAP Clients" on page 198
- The `ldap_cachemgr` daemon. See "The ldap_cachemgr Daemon" on page 199
- Making the Transition from NIS or NIS+ to LDAP. See "Making the Transition from NIS/NIS+ to LDAP" on page 199

For information on LDAP Security, refer to Chapter 12

LDAP Model

LDAP is the emerging industry standard protocol for accessing directory servers. It is a *lightweight* protocol. It uses a simplified set of system-independent encoding methods and runs directly on top of TCP/IP.

LDAP directories provide a way to name, manage, and access collections of directory entries. A directory *entry* is composed of attributes that have a type and one or more values. The syntax for each attribute defines the values allowed (such as ASCII characters or a jpeg photograph) and how those values are interpreted during a directory operation (such as whether a search or compare is case sensitive) .

Directory entries are organized into a tree structure, which may be based on geographic (country), organizational (company) boundaries, or domain components (dc).

Entries are named according to their position in this tree structure by a distinguished name (DN). Each component of the distinguished name is called a relative distinguished name (RDN). An RDN is composed of one or more attributes from the entry. (See RFC 2253 for a formal definition of a distinguished name.)

The hierarchy of the directory tree structure is analogous to that of the UNIX file system. An RDN is analogous to the name of a file, and the DN is analogous to the absolute pathname to the file. As in the UNIX file system, sibling directory entries must have unique RDNs. However, in the directory tree, both leaf nodes and nonleaf nodes can contain content or attributes.

Like the DNS namespace, LDAP directory entries are accessed in a "little-endian" manner. This means that LDAP names start with the least significant component and proceed to the most significant, that just below `root`. The DN is constructed by concatenating the sequence of RDNs up to the root of the tree. For example, if the person named Joe Qwerty works for the company named Ultra Keyboards in the United States, the commonName (CN) attribute for the person Joe Qwerty contains the value "Joseph Qwerty". The DN would contain "cn=Joseph Qwerty, o=Ultra Keyboards, c=US".

Why use LDAP as a Directory Service?

LDAP has the potential to replace existing application-specific directories and consolidate information. This means that changes made on an LDAP server will take effect for every directory-enabled application that uses this information. For example, you can add a variety of information about a new user through a single interface only once, and immediately the user has a UNIX account, a mail address and aliases, membership in departmental mailing lists, etc. When a user leaves, you can remove the user's access to all of these services with a single operation.

A directory is distinguished from a general-purpose database in that a directory contains information that is often searched but *rarely modified*. Host names or user names would be good data types to store in an LDAP directory as they are assigned

once and then rarely changed. Relational databases, however, are much more geared toward maintaining data that is constantly changing and would not be well-suited for an LDAP environment.

A directory can be replicated to protect from unfortunate situations like equipment failure by making the directory data available on multiple servers, known as replica servers. Replicas also improve performance by making more copies of directory data available and by placing the data close to the users and applications that use them.

Reducing load on the authoritative server is not the only reason for using replica servers. Many UNIX networks use Network Information Service (NIS), also known as YP, which uses slave servers on each subnet. As with NIS, putting replicas on subnets can avoid network traffic through routers and reduce latency. However, unlike NIS, the LDAP synchronization scheme features incremental updates that can be pushed immediately to the replicas rather than periodically transferring all of the data.

In order for authoritative information to be maintained, access control needs to be imposed for privileges to read, write, search, or compare. Access control can be done on a subtree, entry, or attribute type and granted to individuals, groups, or "self" (which allows an authenticated user to access his or her own entry). This scheme provides a great deal of flexibility. For example, you might want to only allow people in a personnel department to change the title or manager attributes, allow administrative assistants to change office location and pager number information for just their department, and allow individuals to modify their own home phone number, car license plate, and so on. For more information, refer to your directory server documents.

Consider UNIX login information as an example. Once attributes for users are stored in a directory server, you can synchronize user names and passwords for multiple operating system platforms when updated through a directory server interface. This not only simplifies the change for users but can reduce the chance of having infrequently used accounts with forgotten passwords.

LDAP as a Naming Service in the Solaris Operating Environment

The predominant protocol-independent interfaces to naming services within Solaris are the standard `getXbyY` APIs. An application using `getXbyY`() calls (e.g., `gethostbyname`(3NSL)) goes through the naming service switch which in turn calls the appropriate source protocol. In the case of LDAP, it calls LDAP Application Program Interfaces (APIs) . An LDAP API functions similarly to a subroutine within a program as it retrieves information from a LDAP server. See `nsswitch.conf`(4) for more information about the naming service switch.

In addition to all the features of LDAP previously mentioned, the Solaris client configuration and maintenance is greatly simplified by storing client profiles in the

directory. Each client runs a daemon which is responsible for refreshing the configuration by downloading the latest profile from the directory. See `ldap_cachemgr`(1M) for more information.

LDAP Servers

Schemas

Schemas are definitions describing what types of information can be stored as entries in a server's directory. When information that does not match the schema that is stored in the directory, clients attempting to access the directory may be unable to display the proper results.

In order for a directory server to support Solaris 9 LDAP Naming clients, schemas defined by IETF and some Solaris 9 specific schemas are required.

There are three 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. To support Naming Information Service, the definition of these schemas must be added to the directory server. Detailed information about IETF and Solaris specific schemas is included in Chapter 16. The various RFCs can also be accessed on the IETF website `http://www.ietf.org`.

The Directory Information Tree

Solaris LDAP clients access the information in a default Directory Information Tree (DIT) . This default DIT, however, can be overridden by specifying the modified DIT in the profile. The DIT is divided into containers that are subtrees containing entries for a specific information type.

The search `baseDN` specifies the location in the DIT where all information for the client is found. In the node designated as the search base, the `NisDomainObject` objectclass must exist. The search base node subtrees designate all the containers for the various information types.

Table 11–1 lists the container and information type stored in the DIT.

TABLE 11–1 Default Locations on the Directory Information Tree

Container	Information Type
ou=Ethers	bootparams(4), ethers(4)
ou=Group	group(4)
ou=Hosts	hosts(4), ipnodes(4),publickey(4)
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
nismapname=auto_*	auto_*

LDAP Directory Information Tree (DIT) Terms

Container	A “non leaf” ENTRY
Attribute	A TYPE [of data] with VALUES
Entry	An OBJECT in the DIT that has ATTRIBUTES
ObjectClass	Defines an ENTRY type in the DIT by its ATTRIBUTES
Schema	The set of specific ATTRIBUTES that applies to a given ENTRY

Browsing Indexes (Virtual List Views)

The browsing index, (or Virtual List View) functionality, provides a way in which a client can view a select group and/or number of entries from very long list, thus making the search process less time consuming for each client. You can think of the index functionality as an optimized, pre-defined search. The browsing index output appears in a scrollable graphic user interface, much like a small web browser. When a client uses a indexl for searches, is does not ask the server waste time by returning with the entire list of entries. Rather, it asks the server to return only a select portion of entries to be displayed according to the index used. Indexes are configured on the directory server. Refer to Chapter 10 of the iPlanetDirectory Server Administrator's Guide 5.1 for information on configuring indexes on the iDS as well as the performance cost associated with using them.

Client Profile Attributes

The following lists the amended attributes the make up a client profile for Solaris 9, LDAPv3. You will set these attributes when you run `idsconfig`.

- `preferredServerList`

  the host addresses of the preferred servers

- `defaultServerList`

  host addresses of the default servers

- `defaultSearchBase`

  the base parameter within which a client searches a [server's] database. There is no default for this value and the parameter can be overridden or changed by the `serviceSearchDescriptor` attribute.

- `defaultSearchScope`

  defines the scope of a database search by a client. It can be overridden by the `serviceSearchDescriptor` attribute

- `authenticationMethod`

  identifies the method of authentication used by the client

- `credentialLevel`

  identifies the type of credentials a cleint should use to authenticate. The choices are anonymous or proxy.

- `serviceSearchDescriptor`

  defines how and where a client should search for a naming database, for example, if the client should look in one or multiple points in the DIT and/or if the client should refer to [yet] another entry which would contain additional `serviceSearchDescriptor` attributes.

- `serviceAuthenticationMethod`

Authentication method used by a client's naming database to connect to a server's.

- `attributeMap`

  Attribute mappings used by a Naming client

- `objectclassMap`

  The objectclass mappings used by a client.

- `searchTimeLimit`

  Maximum time [in seconds] a client should allow for a search to complete before timing out.

- `bindTimeLimit`

  Maximum time [in seconds] a client should allow to bind with a server before timing out.

- `followReferrals`

  Specifies whether a client should follow a server's referral to [another LDAP server]

- `profileTTL`

  Lifespan of the a client profile, specifying the time after which the client should pull its refreshed profile off the server.

In addition to the above, there are important attributes that are not part of the client profile.

- `certificatePath`

  The certificate path for the certificate database. The value can be a path to where 'cert7.db' resides, or a full path including the file name.

- `domainName`

  Specifies the DNS domain name (which becomes the default domain for the client machine).

- `proxyDN`

  If the client machine is to be configured to use a proxy credential Level, proxy credentials must be provided using a proxy, defined by`proxyDN=` (the domain name of the proxy) and a `proxyPassword`.

- `proxyPassword`

  As above, if the client machines is to be configured to use a proxy credentialLevel, proxy credentials must be provided using a proxy and a password , defined by `proxyPassword=`.

Command Line Tools

LDAP command line tools support a common set of options, including authentication and bind parameters.

The following are general LDAP utilities, which are not naming service specific:

- `ldapsearch`

  Use to search for directory entries in the namespace. Displays attributes and values found.
- `ldapmodify`

  Use to modify, add, delete, or rename directory entry.
- `ldapadd`

  Use to add new directory
- `ldapdelete`

  Use to delete existing directory entry.
- `ldapmodrdn`

  Rename existing directory entry.

The `ldapsearch`, `ldapadd`, and `ldapmodify` tools support a common text-based format for representing directory information called the LDAP Data Interchange Format (LDIF).

The following are commands used in server setup.

- `idsconfig`

  Use to configure the iPlanet Directory Server 5.1. See "Running `idsconfig`" on page 222 for instructions.
- `ldapclient/ldap_gen_profile`

  Use to [initialize client machines and] create an LDIF of an LDAP client profile.
- `ldapaddent` (actually run on the client machine

  use to create entries in ldap containers from their corresponding `/etc` files.

The following are commands are related to ldap clients.

- `ldapclient`

  Use to initialize client machines.
- `ldap_cachemgr`

  LDAP daemon used to cache server and client information for Network Information Service (NIS) lookups.
- `ldaplist`

  Use to list contents of various services/maps from the directory.

For detailed descriptions of the above commands, refer to the cooresponding man (1) pages.

LDAP Data Interchange Format

The LDAP Data Interchange Format, as defined in RFC 2849, facilitates the transfer of information across difference platforms, applications and naming services, as it provides a "generic" test-based format for entries. (You can think of LDIF as being similar to .txt format for word processing files) For example, the process of transitioning from NIS to LDAP [as the naming service for your network] will include translating the NIS map data into LDIF-formatted entries for LDAP directories.

LDIF is the format produced by the `ldapsearch` tool, the format accepted by the `ldapadd` tool, and is the basis for the change information format that the `ldapmodify` tool uses.

An LDIF file contains one or more entries. Each entry is separated by an empty line. The basic form on an LDIF file entry is:

```
dn: entryDN
attrtype: attrvalue
...
```

where:

- `entryDN`

  is the LDAP Distinguished Name (DN) of the directory entry.

- `attrtype`

  is an LDAP attribute type, such as `cn` or `telephoneNumber`.

- `attrvalue`

  is a value for `attrtype`.

The `attrtype`: `attrvalue` line can be repeated as many times as necessary to list all of the attribute values present in an entry. The line can be continued by inserting a single space or horizontal tab character at the start of the next line.

For example, an LDIF file that contains Joe Qwerty's entry includes five attributes (`cn` and `objectclass` have two values)

```
dn: cn=Joseph Qwerty, o=Ultra Keyboards Inc., c=US
cn: Joseph Qwerty
cn: Joe Qwerty
sn: Qwerty
mail: jqwerty@ultra.com
seeAlso: cn=Joe Qwerty, ou=Engineering Division, o=Peo
 ple, o=IEEE, c=US
objectClass: top
```

```
objectClass: person
```

Note – The value of `seeAlso` is split across two lines by inserting a single space character at the start of the line that begins with " `ple, ...`"

LDAP Clients

When a Solaris client machine is made an LDAP client, it operates similarly to a Solaris client using NIS/NIS+ or DNS. The client does hard lookups, which means the `getXXbyYY()` calls wait until they get a response. Normally NIS(YP) has its servers on the local subnet (as they are normally bound to using a broadcast). Since Solaris 2.0 it has been possible (but not often used) to enable the use of NIS(YP) servers off the local subnet (see the `ypinit`(1M) command) and of course NIS+ is routinely setup without local servers. LDAP is more like NIS+ in it's tendency to deploy non-local servers.

This means that the routers become essential in making your machine work.

You must make sure your clients can always reach at least one of your LDAP servers, either by making sure your network is properly reliable (most are unless someone cuts the wire or turns off the power to the router) or by making sure a server is on the local subnet.

The best method to keep your clients operational is to make sure you have multiple servers, keep those servers up to date (so they have the same data) and make sure your clients can reach all of them.

Using Fully Qualified Domain Names

One significant difference between an LDAP client and a NIS or NIS+ client is that an LDAP client always returns a FQDN (fully qualified domain name) for a hostname (similar to those returned by DNS). For example, if your domain name is `engineering.example.net` and you lookup the hostname `server` with `getipnodebyname()` (as they should in preparation for the conversion to IPv6 even though LDAP in this release only runs over IPv4). Both `gethostbyname()` and `getipnodebyname()` return the FQDN version `server.engineering.example.net`. Also if you use interface specific aliases like `server-#` you will see a long list of fully-qualified host names returned.

If you are using hostnames to share file systems or have other such checks you need to realize this key difference and account for it. Especially if you *assume* non-FQDN for

local hosts and FQDN only for remote (DNS resolved) hosts. If you setup LDAP with a different domain name from DNS you might be surprised when the same host has two different FQDNs, depending on the lookup source.

The ldap_cachemgr Daemon

The `ldap_cachemgr`(1M) is a daemon that runs on LDAP client machines. It performs the following key functions:

- It accesses the configuration files, running as root.
- It refreshes the client configuration information contained in the /var/ldap/ldap_client_file on the server and "pushes" this data to the clients in real time.
- It maintains a real-time, sorted list of servers to "check in with".
- It improves lookup efficiency by caching common look-up requests submitted by various clients.
- It improves the efficiency of RDN to DNS domain name mapping.

Note – The `ldap_cachemgr` *must be running at all times* in order for the constant refresh functionality to work.

Making the Transition from NIS/NIS+ to LDAP

For detailed information on transitioning Solaris 9 client machines from NIS+ to LDAP, refer to "Transitioning from NIS+ to LDAP" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*

New Features included with the Solaris 9 LDAP Implementation

The following outline the newest LDAP features for Solaris 9.

- Simplified LDAP directory services setup using `idsconfig`. See Chapter 15.
- Real-time refresh rate of data transfer between client and server, via the `ldap_cachemgr`.
- A more robust security model, which supports strong authentication, TLS encrypted sessions, and provides the basis for password management support. For example, a client's proxy credentials are *NO LONGER* stored in a client's profile [on the server]. They must be entered manually each time to setup or amend a client's profile.

CHAPTER 12

Solaris LDAP Security Model (Overview)

Security Basics

To access the information stored in the directory, clients can first establish identity with the directory server. Next, they are allowed access to part or all of the information available in the directory, either by making directory queries (lookups) or directory updates, based on the Access Control Information (ACI). (For more information on ACI, please consult the iPlanet Directory Server 5.1 Administrator's Guide.) Thus, the LDAP security model is essentially a two-step process. First, if the client is connecting as anything other than `anonymous` for any given request, the client must prove its identity to the server to be able to establish any connection at all. Second, if the server 'acknowleges' the client's identity, for any given request, the client must authenticate to the server using a specific credential level which determines to what part of the server's Directory Information Tree (DIT) it has access.

This chapter discusses the concepts of client identity, authentication methods, and `pam_ldap` and `pam_unix` modules.

Assigning Client Credential Levels

LDAP clients can be assigned three possible credential levels with which to authenticate to a directory server: anonymous, proxy, and proxy-anonymous.

- `anonymous`

  Anonymous does not provide any level of security. Any client which assumes the `anonymous` identity can browse and update (read and write to) all Network Information records, (including password and shadow information).

Caution – Allowing anonymous write to a directory should never be done, as anyone coud change any piece of information in the DIT, including another user's password.

- `proxy`

  The client authenticates to the directory using a proxy account in the directory. This proxy account can be any entry that is allowed to bind to the directory (in the iPlanet Directory Server, this translates to any entry which has a `userPassword` attribute). If you choose to use the `proxy` method of authentication, you will need to configure the `proxyDN` and `proxyPassword` on every client. In addition, if the `proxyPassword` changes, you will need to change it on every client as well.

  You can setup different proxies for different groups [of clients] within an organization. For example, you can configure a proxy for all the sales client servers to access the 'company-wide-accessible' directories and directories that store sales information, while preventing sales client servers from accessing human resource directories with payroll information. Or, in the most extreme cases, you can assign different proxy agents to each individual client server and/or assign just one proxy to all client servers within the corporate network. A typical LDAP deployment would probably lie between the two extremes. Consider the choices carefully. Too few proxy agents might limit the your ability to control user access to resources. However, if you set up too many agents complicates the setup and maintenance of the system and require a large number of profiles as well.

  ACIs define the type of access assigned to any given proxy. ACIs are defined on the server, so you need to consult your directory server documentation to learn how to set them.

Note – As client configuration information is stored in client profiles, which reside on the directory server, the number of proxy agents you create will effect the number of client profiles to be defined. See "Creating a Client Profile" on page 233 for more information.

- `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 that fails, the client will then connect as `anonymous`.

Credential Levels and Client Profiles

If you configure a client to use a proxy identity, the client saves its`proxyDN` and password in its `/var/ldap/ldap_client_cred` file. For the sake of increased

security, this file is restricted to root-access only and the value of `proxyPassword` is encrypted. Note that the `/var/ldap/ldap_client` file is separate from the client profile. While past LDAP implementations have stored proxy credentials in a client's profile, in the Solaris 9 LDAP implementation, they are not. The proxy credentials are set manually using the `ldapclient` command (with the `genprofile` option) . This results in improved security surrounding a client's name and password information.

See "Creating a Client Profile" on page 233 and your directory server documentation 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. The authentication method might have a *transport security option* associated with it. Solaris and the 5.1 iPlanet directory server support the following authentication methods:

- `anonymous`
- `SIMPLE`
- `sasl DIGEST-MD5` (without privacy or integrity options)
-

The authentication method, like the credential level, might be multi-valued. For example, in the client profile you could specify that the client first try to bind with the directory server [over a TLS secured line] using the `SIMPLE` method and then try to bind with the `DIGEST-MD5` method if the `SIMPLE` method fails.

Note – Currently, TLS support is limited to LDAP servers (like the iDS 5.1) using the default ports 389 and 636 for LDAP and SSL, respectively.

The line in the client profile would appear as:

```
tls:simple;sasl/DIGEST-MD5
```

Note – `idsconfig` allows you to configure the above line manually.

- `SIMPLE`

  If the client machine uses the `SIMPLE` method, it authenticates to the server by sending a simple bind request. Please note that the client's password is transmitted in the clear and is subject to snooping when you use this method. The primary advantage of using `SIMPLE` (LDAP standard) method is that all directory servers support it.

However, if `tls:simple` is used, the LDAP session is encrypted so that the password is protected.

- `CRAM-MD5`

  See RFC 2195 for information on the `CRAM-MD5` method which is supported by some, but not all directory servers.

Note – The iPlanet 5.1 Directory Server *does support* `CRAM-MD5` .

- `DIGEST-MD5`

  Some directory servers, including the iDS 5.1, also support (`DIGEST-MD5`) through Simple Authentication and Security Layer (`SASL`). 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 `SIMPLE`. See RFC 2831 for information on `DIGEST-MD5`. `DIGEST-MD5` is considered an improvement over CRAM-MD5 for its improved security and multi-application compatibility. See RFC 2222 for information on `SASL`

Caution – iDS 5.1 (and 5.0) require passwords to be stored in the clear in order to use `DIGEST-MD5`. Passwords for all proxy users will need to be stored in the clear, which you can set using the `clear` tag with `ldapadd (1)`. If you wish to use `pam_ldap`, set the `passwordstoragescheme` attribute of `cn=config` to `clear`. In theis case, you would not want to allow for read access of the`userPassword`.

Using the Pluggable Authentication Module (PAM) for Authentication by Multiple Applications

The use of PAM allows for a way in which a non-Solaris specific application, such as ftp, can authenticate to a Solaris server [running LDAP] without having to be SIMPLE or DIGEST-MD5 compatible. By using the PAM layer, applications authenticate to the server without worrying about what authentication method is defined by the system administrator for any given client. You can use either `pam_unix` or `pam_ldap` in conjunction with LDAP. Because of its increased flexibility and support of stronger authentication methods, the use of `pam_ldap` is recommended. Generally, `pam_ldap` does not store passwords in crypt format. Thus, if you required backwards compatibility with NIS or NIS+, which require passwords to be stored in crypt format, you must use`pam_unix`. However, with iDS 5.1, you can use `pam_ldap` as long as the `passwordtoragescheme` attribute of `cn=config` is set to `crypt` and you are *not* using `digest-MD5`.

To use PAM in conjunction with LDAP, you must configure one of two `pam` modules in `pamconf`: `pam_unix`(5) and `pam_ldap`(5)(the preferred method).

PAM allows you to choose any combination of authentication methods per application, (ftp, telnet, or login, for example). In addition it allows for the setting of multiple passwords on high-security systems.

- `pam_unix`

  When `pam_unix` follows the traditional model of UNIX authentication, which means:

  - the client retrieves the user's encrypted password from the server's directory.
  - the user is prompted for his password.
  - that user's password is encrypted.
  - the two encrypted passwords are compared to decide if the user should be authenticated or not. If LDAP clients are configured with this module, the `userPassword attribute must be readable by the identity that the client is using (anonymous or the configured proxy agent). Additionally, there are two more restrictions when using pam_unix:`

  1. The password must be stored in an attribute called `userPassword`.
  2. The password must be stored in `UNIX crypt` format (not clear text or encrypted by other encryption methods). As mentioned above, use of the `pam_unix` format is *required* with NIS and NIS+.

  Also note that when using `pam_unix`, the `userPassword` attribute muct be readable by the client. For example, if the credential level is `anonymous`, then anonymous must be able to read the `userPassword` attribute. Or, if the credential level is `proxy`, then proxy must be able to read the `userPassword` attribute.

- `pam_ldap`

  Since having to store passwords in `crypt` format is limiting, `pam_ldap()`, is recommended if you do not have to allow for backwards compatibility with NIS or NIS+.

  Because `pam_ldap` authenticates users directly to the directory server, you can specify user level access controls to control an individuals' authentication using ACIs. ACIs are set automatically when you use `idsconfig` to configure the server. See "Running `idsconfig`" on page 222 for more information.

  Use the `passwd` command to change a password.

An Example `pam.conf` file

```
# Authentication management
#
login   auth sufficient /usr/lib/security/$ISA/pam_unix.so.1
login   auth required   /usr/lib/security/pam_ldap.so.1 try_first_pass
login   auth required   /usr/lib/security/$ISA/pam_dial_auth.so.1
#
```

```
rlogin  auth sufficient /usr/lib/security/$ISA/pam_rhosts_auth.so.1
rlogin  auth sufficient /usr/lib/security/$ISA/pam_unix.so.1
rlogin  auth required   /usr/lib/security/pam_ldap.so.1 try_first_pass
#
dtlogin auth sufficient /usr/lib/security/$ISA/pam_unix.so.1
dtlogin auth required   /usr/lib/security/pam_ldap.so.1 try_first_pass
#
rsh     auth required   /usr/lib/security/$ISA/pam_rhosts_auth.so.1
other   auth sufficient /usr/lib/security/$ISA/pam_unix.so.1
other   auth required   /usr/lib/security/$ISA/pam_dial_auth.so.1
```

CHAPTER 13

Post-Installation Configuration of iPlanet Directory Server 5.1

This chapter discusses how to configure the iPlanet Directory Server 5.1 (iDS 5.1) to get it up and running. You must complete the procedures contained in this chapter before you can go on to configure the iDS 5.1 for use with Solaris LDAP clients.

Preparing for Configuration

Before you begin configuring the iDS 5.1, you should have an understanding of the various Directory Server components and the design and configuration decisions you need to make.

To help you configure iDS 5.1, you should be familiar with the concepts contained in the following sections:

- Components
- Configuration Decisions
- Configuration Process Overview
- Configuration Privileges

The *iPlanet Directory Server Deployment Guide* contains basic directory concepts as well as guidelines to help you design and successfully deploy your directory service.

Configuration Components

iDS 5.1 contains the following software components:

- iPlanet Console iPlanet Console

  provides the common user interface for all iPlanet server products. From it you can perform common server administration functions such as stopping and starting servers, installing new server instances, and managing user and group information. iPlanet 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 Administration Server

  is a common front-end to all iPlanet servers. It receives communications from iPlanet Console and passes those communications on to the appropriate iPlanet server. Your site will have at least one Administration Server for each server root in which you have installed an iPlanet server.

- Directory Server

  is iPlanet's LDAP implementation. The Directory Server runs as the `ns-slapd` process on Solaris. This is the server that manages the directory databases and responds to client requests. Directory Server is a required component.

Note – If you install the entire Solaris disk suite, all of the above components are installed by default.

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:

- Port number
- Server root
- 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 Directory Server:

- The standard Directory Server (LDAP) port number is 389.
- Port 636 is reserved from 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.
- Directory Server must be run as root if will listen on 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.

For information on how to set up LDAP over SSL (LDAPS) for Directory Server, see the *iPlanet 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 Directory Server.

You must therefore decide what user accounts you will use for the following purposes:

- The user and group under which you will run Directory Server.

  If you will not be running the directory server as root, it is strongly recommended that you create a user account for all iPlanet 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 directory server 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 iPlanet 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 iPlanet servers, such as `gid iPlanet`, to ensure that files can be shared between servers when necessary.

Before you can install Directory Server 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 iDS 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 former releases of 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 Directory Server. Therefore, you must not manually create an actual Directory Server 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.

- Configuration Directory Administrator ID and password.

  The configuration directory administrator is the person responsible for managing all the iPlanet servers accessible through iPlanet Console. If you log in with this user ID, then you can administer any iPlanet server that you can see in the server topology area of iPlanet Console.

  For security, the configuration directory administrator should not be the same as the directory manager. The default configuration directory administrator ID is `admin`.

- 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 the iPlanet servers stored in the local server root.

  Administration Server user ID and password is used only when the Directory Server 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 Directory Server, 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 `myco.com`, then select a suffix of `dc=myco,dc=com`.

For more information on planning the suffixes for your directory service, see the *iPlanet Directory Server 5.1 Deployment Guide*.

Choosing the Location of the Configuration Directory

Many iPlanet servers including Directory Server 5.1 use an instance of Directory Server to store configuration information. This information is stored in the `o=NetscapeRoot` directory tree. It does not need to be held on the same Directory Server as your directory data. Your configuration directory is the Directory Server that contains the `o=NetscapeRoot`

If you are installing Directory Server only to support other iPlanet servers, then that Directory Server is your configuration directory. If you are installing Directory Server to use as part of a general directory service, then you will have multiple Directory Servers 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 iPlanet servers (including iDS 5.1).

For ease of upgrades, you should use a Directory Server 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 Directory Server 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 Directory Server instance. However, for very large sites that are installing a large number of iPlanet 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. iPlanet 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 *iPlanet Directory Server 5.1 Deployment Guide* for information on using replication and DNS round robins to increase directory availability.

Caution – Corrupting the configuration directory tree can result in the necessity of reinstalling all other iPlanet 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 iPlanet 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 iPlanet servers should ever modify the configuration.

Choosing the Location of the User Directory

Just as the configuration directory is the Directory Server that is used for iPlanet server administration, the user directory is the Directory Server 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 iPlanet 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 iPlanet 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 they belong to.
- 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 iPlanet servers, regardless of the domain that they belong to.
- 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 iPlanet servers, create three administration domains each named after a different customer.

Configuration Process Overview

You can use one of several configuration processes to install Directory Server. 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 Directory Server 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 iDS 5.1. See "Using Express Configuration" on page 215.

- *Typical configuration*

  Use this if you are performing a normal install of Directory Server. See "Using Typical Configuration" on page 216.

- *Custom configuration*

  In iDS 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 iDS 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 *iPlanet Directory Server 5.1 Deployment Guide*.
2. Configure your Directory Server 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 *iPlanet Directory Server 5.1 Administrator's Guide*.
4. Create additional Directory Server instances and set up replication agreements between your directory servers to ensure availability of your data.

Using Express and Typical Configuration

Using Express Configuration

Use express configuration if you are installing Directory Server 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 iDS 5.1 using Express Configuration

1. **Log in as superuser.**
2. **Run the Directory Server program by typing:**

   ```
   # /usr/sbin/directoryserver setup
   ```

3. **When you are prompted for the type of configuration, choose Express.**
4. **For the user and group to run the servers as, enter the identity that you want this server to run as.**
5. **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 iPlanet 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 Directory Server configuration:

   - The Directory Server 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.myco.com`, then you will have the suffix `dc=myco, 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 Directory Server, see the *iPlanet Directory Server 5.1 Administrator's Guide*.

Using Typical Configuration

Most first time configurations of Directory Server 5.1 can be performed using the Typical option of the setup program.

▼ How to Configure iDS using Typical Configuration

1. **Log in as Superuser.**
2. **Run the Directory Server program.**

   ```
   # /usr/sbin/directoryserver setup
   ```

3. **When you are prompted for what you want to install, hit enter for [the default] iPlanet 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 in 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 Directory Server port, select the default (389) unless you already have another application using that port.**

12. **For the Directory Server 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 Directory Server 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 Directory Server instance to be installed into a directory labeled `slapd-phonebook`.

Caution – The directory server identifier must not contain a period. For example, myco.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 myco.com, then enter `dc=myco,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 Directory Server, 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 Directory Server). 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 Directory Server, see the *iPlanet Directory Server 5.1 Administrator's Guide*.

CHAPTER 14

iPlanet Directory Server Setup (Tasks)

This chapter describes how to configure the iPlanet Directory Server 5.1 (iDS 5.1) to support a network of Solaris LDAP clients. The information is specific to version 5.1 of the iPlanet Directory Server.

Note – You must have already performed all the post-installation configuration procedures described in the previous chapter before you can configure the iDS 5.1 to work with Solaris LDAP clients.

Refer to the following iPlanet manuals for in-depth information regarding the iDS 5.1.

- *iPlanet Directory Server Schema Reference Guide*
- *iPlanet Directory Server Deployment Manual*
- *iPlanet Directory Server Configuration, Command, and File Reference*
- *iPlanet Directory Server Administrator's Guide*

Configuring iDS 5.1using `idsconfig`

Creating a Checklist Based on Your Server Installation

During the server installation process, you will have entered defined crucial variables about both the iDS server itself and its clients' profiles. We suggest you create a checklist similar to the one below before launching `idsconfig`.

Note – The information included below will serve as the basis for all examples that follow in this chapter. The example domain is of an widget company, MyCo, Inc. with stores nationwide. The examples will deal with the West Coast Division, with the domain west.myco.com

TABLE 14–1 Server Variables Defined

Variable	Definition for MyCo Network
Port number at which an instance of the directory server is installed (DEFAULT=389)	**`default`**
Name of Master Server [Distinguished Name (DN) format]	**`ipdserver`** [from the FQDN ipdserver.west.myco.com or 192.168.0.0]
Replica Server(s) [IPnumber:port number]	**`192.168.0.1 :390`**[for ipdrep.west.myco.com]
Directory Manager [dn: cn=directory manager]	**`default`**
Domain name to be served [dc=subdomain dc=maindomain dc=com OR dc=maindomain, dc=com (NOTE: the comma after main indicates including all entries below main domain.)	**`west.myco.sun`**
Time (in seconds) to process client requests before timing out	2
Size of data transfer (number of entries returned per request)	[default]

TABLE 14–2 Client Profile Variables Defined

Variable	Deifinition for MyCo Network
Profile Name	WestUserProfile
Server List (defaults to the local subnet)	west.myco.com
Preferred Server List (listed in order of which server to try first, second, etc.)	**`192.168.0.1`** [for idsrep.west.myco.com]
Search Scope (number of levels down through the directory tree. 'One' or 'Sub')	**`One`** [default]
Authentication Method used to gain access to server	**`anonymous`**

TABLE 14–2 Client Profile Variables Defined *(Continued)*

Variable	Deifinition for MyCo Network
Referrals (a pointer to another server if the main server is unavailable)	`Y`[would be specified in the ipdserver directory as, for example refserver.west.myco.com or 192.168.0.3]
Search Time Limit (in seconds, default 30) for waiting for server to return information	`default`
Bind Time Limit (in seconds, default 10) for contacting server. The default is seconds.	`2`

Note – Client profiles are defined per domain.

Solaris ObjectClass Definitions

`idsconfig` automatically adds the necessary Solaris ObjectClass Definitions. Unless you are very experienced in ldap administration, do not manually modify these definitions.

- NisKeyObject
- nisDomainObject
- SolarisNamingProfile
- mailGroup
- nisMailAlias
- nisNetId
- SolarisAuditUser
- SolarisExecAttr
- SolarisProject
- IPP printer
- Sun printer

Running idsconfig

▼ How to Configure the iPlanet Directory Server Using idsconfig

1. **Call up** idsconfig**.**

   ```
   /usr/lib/ldap/idsconfig
   ```

2. **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 219. 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 Service Search Descriptors (ssd). Refer to "Using Service Search Descriptors to Modify Client Access to Various Services" on page 224 for a detailed discussion.

   A carriage return sign after the prompt means that you are accepting the [default] by hitting enter.

   EXAMPLE 14–1 Running idsconfig for the YourCo Network

```
It is strongly recommended that you BACKUP the directory server
before running idsconfig.

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 iPlanet Directory Server's (iDS) name to setup: ipdserver
Enter the port number for iDS (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.myco.com]
Enter LDAP BaseDN (h=help): [dc=west,dc=myco,dc=com]
Enter the profile name (h=help): [default]
Are you sure you want to overwrite profile cn=default? y
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:
```

EXAMPLE 14–1 Running `idsconfig` for the YourCo Network *(Continued)*

```
  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:sals/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] n
Do you want to modify the server timelimit value (y/n/h)? [n] n
Do you want to modify the server sizelimit value (y/n/h)? [n] n
Do you want to store passwd's in "crypt" format (y/n/h)? [n] y
Do you want to setup a Service Authentication Methods (y/n/h)? [n] n
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] n

              Summary of Configuration

  1  Domain to serve               : eng.sun.com
  2  BaseDN to setup               : dc=eng,dc=sun,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  Authenication Method          : simple
  9  Enable Follow Referrals       : FALSE
 10  iDS Time Limit                :
 11  iDS Size Limit                :
 12  Enable crypt passwd storage   : 1
 13  Service Auth Method pam_ldap  :
 14  Service Auth Method keyserv   :
 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] 0
Enter DN for proxy agent: [cn=proxyagent,ou=profile,dc=west,dc=myco,dc=com]
Enter passwd for proxyagent:
Re-enter passwd:
```

EXAMPLE 14–1 Running `idsconfig` for the YourCo Network *(Continued)*

```
WARNING: About to start committing changes. (y=continue, n=EXIT) y

  1. Changed passwordstoragescheme to "crypt" in cn=config.
  2. Schema attributes have been updated.
  3. Schema objectclass definitions have been added.
  4. NisDomainObject for dc=west,dc=myco,dc=com was already set.
  5. Top level "ou" containers complete.
  6. Nis maps: auto_home auto_direct auto_master auto_shared processed.
  7. Top level ACI LDAP_Naming_Services_deny_write_access
already exists for dc=west,dc=myco,dc=com.
  8. Add of VLV Access Control Information (ACI).
  9. Proxy Agent cn=proxyagent,ou=profile,dc=west,dc=myco,dc=com added.
  10. Give cn=proxyagent,ou=profile,dc=west,dc=myco,dc=com read premission for passwd.
  11. Generated client profile and loaded on server.
  12. Processing eq,pres indexes:
      ipHostNumber (eq,pres) skipped already exists
      uidNumber (eq,pres) skipped already exists
      ipNetworkNumber (eq,pres) skipped already exists
      gidnumber (eq,pres) skipped already exists
      oncrpcnumber (eq,pres) skipped already exists
  13. Processing eq,pres,sub indexes:
      membernisnetgroup (eq,pres,sub) skipped already exists
      nisnetgrouptriple (eq,pres,sub) skipped already exists
  14. Processing VLV indexes:
      getgrent vlv_index skipped already exists
      gethostent vlv_index skipped already exists
      getnetent vlv_index skipped already exists
      getpwent vlv_index skipped already exists
      getrpcent vlv_index skipped already exists
      getspent vlv_index skipped already exists

idsconfig: Setup of iDS server mirage is complete.
```

Using Service Search Descriptors to Modify Client Access to Various Services

A service search descriptor 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 without having to change your existing LDAP database and data.

EXAMPLE 14–2 A Smart Way to Use Service Search Descriptors

Assume your predecessor at MyCo 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, which is the ldap standard. 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 MyCo's existing database and to rewrite all the exiting applications on MyCo's network so that they are compatible with the new ldap. 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 iPlanet Directory Server 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: serviceou=user
*/OR, the longhand version*/ou=user, dc=west, dc=myco, dc=comEnter 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=myco, 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
```

EXAMPLE 14–2 A Smart Way to Use Service Search Descriptors *(Continued)*

```
  Q  Exit menu
Enter menu choice: [Quit] q
```

For information on the other services for which the search method can be modified using SSDs, refer to `ldap`(1)

Populating the iDS Server using `ldapaddent`

Use `ldapaddent`. You will be prompted for the Directory Manager password.

Note – Before populating the directory server with data, you must configure the server to store passwords in `Unix Crypt` format For more information on setting the password `Unix Crypt` format, see the iPlanet Directory Server documents.

`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. (the —b option allows you to overrride the default)

See the `ldapaddent (1)` man page for detailed information on `ldapaddent` and see Chapter 12 for information regarding LDAP security and write-access to the Directory Server.

Supporting Virtual List Views (VLVs)

▼ How to Verify that the Directory Supports Virtual List Views.

1. **Use `ldapsearch` to determine if the directory supports Virtual List Views as identified by their OIDs: 1.2.840.113556.1.4.473 VLV control type and 2.16.840.1.113730.3.4.9 VLV control value.**

   ```
   # ldapsearch -b "" -s base objectclass=\*
   ```

   For our example configuration, `ldapsearch` returns:

```
objectclass=top
namingcontexts=dc=sun,dc=com
namingcontexts=o=NetscapeRoot
subschemasubentry=cn=schema
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=2.16.840.1.113730.3.4.3
supportedcontrol=2.16.840.1.113730.3.4.4
supportedcontrol=2.16.840.1.113730.3.4.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=2.16.840.1.113730.3.4.9
supportedcontrol=2.16.840.1.113730.3.4.12
supportedsaslmechanisms=EXTERNAL
supportedldapversion=2
supportedldapversion=3
dataversion=atitrain2.east.sun.com:389 020000605172910
netscapemdsuffix=cn=ldap://:389,dc=atitrain2,dc=east,dc=sun,dc=com
```

   Note – For more information on `ldapsearch` see `ldapsearch`(1).

2. **Index the following list of Virtual List View attributes.**

```
getpwent:      vlvFilter: (objectclass=posixAccount),   vlvScope: 1
getspent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getgrent:      vlvFilter: (objectclass=posixGroup),       vlvScope: 1
gethostent:    vlvFilter: (objectclass=ipHost),           vlvScope: 1
getnetent:     vlvFilter: (objectclass=ipNetwork),        vlvScope: 1
getprotoent:   vlvFilter: (objectclass=ipProtocol),       vlvScope: 1
getrpcent:     vlvFilter: (objectclass=oncRpc),           vlvScope: 1
getaliasent:   vlvFilter: (objectclass=rfc822MailGroup),  vlvScope: 1
getserviceent: vlvFilter: (objectclass=ipService),        vlvScope: 1
```

   Create these indexes for any `ou` in the tree that contains a large number of objects or for those that are heavily accessed.

3. **Use `ldapmodify` to give "anyone" read, search, compare permission for VLV feature. This ensures anonymous searches do not fail when trying to use VLV control.**

```
#ldapmodify -D "cn=Directory Manager" -w nssecret -f vlvcntrl.ldif
```

the contents of `vlvcntrl.ldif` are:

```
dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
changetype: modify
replace: aci
aci: (targetattr !="aci")(version 3.0; acl "VLV Request Control";
 allow (compare,read,search) userdn = "ldap:///anyone"; )
```

4. **Use `ldapsearch` to show the changes to the VLV control ACI.**

```
#ldapsearch -L -b "cn=features,cn=config" -s one \
oid=2.16.840.1.113730.3.4.9
```

The ACI returned by `ldapsearch` would look like:

```
dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
objectclass: top
objectclass: directoryServerFeature
oid: 2.16.840.1.113730.3.4.9
cn: VLV Request Control
aci: (targetattr !="aci")(version 3.0; acl "VLV Request Control";
 allow (compare,read,search) userdn = "ldap:///anyone"; )
```

5. **Create the VLV index for `getpwent`.**

```
# cd /usr/netscape/server4/slapd*
# ./vlvindex getpwent
OK# ./vlvindex getgrent
OK# ./vlvindex gethostent
OK# ./vlvindex getspent
OK#
# ./vlvindex
[05/Jun/2000:15:34:31 -0400] - ldbm2index: Unknown VLV Index named ''
[05/Jun/2000:15:34:31 -0400] - ldbm2index: Known VLV Indexes are:
'getgrent', 'gethostent', 'getnetent', 'getpwent', 'getspent',
```

6. **Repeat steps 4 an 5 for the rest of the Virtual List View attributes.**

▼ How to Give “anyone” Read, Search, and Compare Permission on VLV Request Control

- **Use `ldapsearch` to show the VLV control ACI.**

```
#ldapsearch -D "cn=Directory Manager" -w nssecret -b cn=features, \
cn=config objectclass=\*
```

The result of the search is:

```
cn=features,cn=config
objectclass=top
cn=features

cn=options,cn=features,cn=config
objectclass=top
cn=options

oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
objectclass=top
objectclass=directoryServerFeature
oid=2.16.840.1.113730.3.4.9
cn=VLV Request Control
aci=(targetattr != "aci")(version 3.0; acl "VLV Request \
 Control"; allow( read,
search, compare ) userdn = "ldap:///all";)
```

Additional iPlanet Directory Server Administration Tasks

▼ How to Create Browsing Indexes to Improve Search Performance

Note – For information about how to create an index, see "Managing Indexes" in the iPlanet *Directory Server Administrator's Guide*.

1. **Index the following list of Solaris client attributes.**

```
membernisnetgroup    pres,eq,sub
nisnetgrouptriple    pres,eq,sub
memberuid            pres,eq
macAddress           pres,eq
uid                  pres,eq
uidNumber            pres,eq
gidNumber            pres,eq
ipHostNumber         pres,eq
ipNetworkNumber      pres,eq
ipProtocolNumber     pres,eq
oncRpcNumber         pres,eq
ipServiceProtocol    pres,eq
ipServicePort        pres,eq
```

```
nisDomain            pres,eq
nisMapName           pres,eq
mail                 pres,eq
```

2. **For the password entry (getpwent), add the following entries to the directory.**

```
dn: cn=getpwent,cn=config,cn=ldbm
objectclass: top
objectclass: vlvSearch
cn: getpwent
vlvBase: ou=people,dc=eng,dc=sun,dc=com
vlvScope: 1
vlvFilter: (objectclass=posixAccount)
aci: (target="ldap:///cn=getpwent,cn=config,cn=ldbm")(targetattr="*")
 (version 3.0; acl "Config";allow(read,search,compare)
userdn="ldap:///anyone";)
dn: cn=getpwent,cn=getpwent,cn=config,cn=ldbm
cn: getpwent
vlvSort: cn uid
objectclass: top
objectclass: vlvIndex
```

▼ How to Generate a Client Profile Manually

1. **Generate the client profile and then add it to the LDAP server.**

```
ldap_gen_profile -P profile -b baseDN -D bindDN \
  -w bindDNpasswd ldapServer_IP_address(es)[:port#]
```

The `bindDN` is the bind DN of the proxy agent. You can specify more than one LDAP server's IP address if you want to allow fail over to another LDAP server. Capture the above result in a file, such as `profile.ldif`.

A typical command looks like:

```
ldap_gen_profile -P myProfile -b "dc=mkt,dc=mainstore,dc=com" \
-D "cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com" \
-w proxy_agent_pswd -a simple 100.100.100.100 > profile.ldif
```

2. **Add this client profile into LDAP server so that clients can download it.**

Related Books

Understanding and Deploying LDAP Directory Services, by Timothy A. Howes, PhD, Mark C. Smith, and Gordon S. Good, (Macmillan Technical Publishing, 1999)

CHAPTER 15

General Server Setup

This chapter describes the following basic components of setting up an LDAP server, including the specific procedures.

General Requirements

To support Solaris 9 LDAP clients, the server, regardless of what brand, must support the LDAP v3 protocol, as Solaris naming clients use controls that are available only in v3.

At least one of the following controls must be supported:

- Simple paged-mode (RFC 2696)
- Virtual List View controls

The server must support one of the following authentication methods:

- `anonymous`
- `simple`.
- `SASL CRAM-MD5`

In addition, the server must support command naming and auxiliary object classes DEFINE]. The iPlanet Directory Server 5.1 meets all of the above requirements. Refer to Chapter 14for its specific set up procedure. Otherwise if you are using another brand of server that meets the above requirements, please follow the setup procedures described in this chapter.

Please refer to Chapter 12 for a detailed discussion of the LDAP security model.

Simple Page Mode Control

Sun servers use this instead of VLV. Briefly define what it means. For information on Virtual List View, please refer to "Supporting Virtual List Views (VLVs)" on page 227.

▼ Verify that Directory Supports Simple Page Mode Control.

- **Use `ldapsearch` to determine if the directory supports simple page mode control as identified by their OIDs: 1.2.840.113556.1.4.319 simple page mode control type and 2.16.840.1.113730.3.4.2 simple page mode control value.**

```
# ldapsearch -b "" -s base objectclass=\*
```

For our example configuration, `ldapsearch` returns:

```
objectclass=top
namingcontexts=dc=sun,dc=com,o=internet
subschemasubentry=cn=schema
supportedsaslmechanisms=CRAM-MD5
supportedextension=1.3.6.1.4.1.1466.20037
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=2.16.840.1.113730.3.4.2
supportedldapversion=2
supportedldapversion=3
```

Directory Information Trees

Overriding the Default Containers in the DIT

If a particular LDAP deployment requires the default containers be modified in some way, specify the modified container in the profile. You can then define an alternate search `baseDN` for each of the databases

For example, assume that an organization wants to replace the `ou=People` container with `ou=fulltime` and `ou=contractor` containers. For this profile entry (which can exist anywhere in the DIT), an alternate search DN needs to be specified. Generate

the LDAP client profile using the -B option to specify an alternate search DN. See ldapclient(1M) (using the genprofile option for details. The attribute looks like:

```
SolarisDataSearchDN="passwd:(ou=fulltime,dc=mkt,dc=mystore,dc=com),
                     (ou=contractor,dc=mkt,dc=mystore,dc=com)"
```

Defining the NIS Domain Attribute

In order for the Solaris clients to find a server for a specific domain, the nisDomain attribute of the nisDomainObject objectclass must be defined in the root DN entry of the DIT representing the desired domain. This information is used by the client when initializing the system and refreshing the client profile. During the initialization, the client searches for an entry on the LDAP server that has the nisDomain matching the desired domain. The DN for the entry found will be used as the BaseDN for the naming information.

For illustrative purposes, this document uses the following nisDomain:

```
dn: dc=mkt,dc=mainstore,dc=com
dc: mkt
objectclass: top
objectclass: domain
objectclass: nisDomainObject
nisdomain: mkt.mainstore.com
```

Creating a Client Profile

To simplify Solaris client setup, and avoid having to reenter the same information for each and every client binding to your server, create a “onetime” client profile on the server which can be used for clusters of, or for all of the clients. This single client profile will define the LDAP environment to be used by Solaris clients.

The benefit of having set up profiles extend beyond the original setup. When it comes time to modify, delete or add client information, the profiles facilitate a less time-consuming process. See ldapclient(1M) (using the genprofile option) for details.

▼ How to Create a Client Profile

The `ldapclient`(1M) command (using the `genprofile` option) is provided as part of the Solaris client tools to create client profiles. This tool generates an `LDIF` file which can be stored in the LDAP server using the `ldapadd`(1) command. The following example shows how to create a client profile in LDIF format.

1. **Use `ldapclient`(1M) with the `genprofile` option to create a client profile with all values changedfrom the default.**

```
# ldapclient genprofile -a -v profileName=eng \
        -a credentialLevel=proxy -a authenticationMethod=sasl/DIGEST-MD5 \
        -a bindTimeLimit=20 -a proxyPassword=foo \
        -a defaultSearchBase=dc=eng,dc=ge-uk,dc=com \
        -a serviceSearchDescriptor=passwd:ou=people,dc=lab,dc=ge-uk,dc=com?one \
        -a defaultSearchScope=sub \
        -a attributeMap=passwd:uid=employeeNumber \
        -a objectclassMap=passwd:posixAccount=unixAccount \
        -a proxyDN=cn=proxyagent,ou=profile,cd=eng,dc=ge-uk,dc=com \
        -a followReferrals=false -a profileTTL=6000 \
        -a preferredServerList=129.100.100.30 -a searchTimeLimit=30 \
        -a "defaultServerList=129.100.200.1 129.100.100.1 204.34.5.6" > eng.ldif
```

The following example shows the profile generated:

```
dn: cn=eng,ou=profile,dc=ge-uk,dc=com
SolarisBindDN: cn=proxyagent,ou=profile,dc=eng,dc=ge-uk,dc=com
SolarisBindPassword: {NS1}xxxxxxxxxxxxxx
SolarisLDAPServers: 129.100.100.30
SolarisSearchBaseDN: dc=eng,dc=ge-uk,dc=com
SolarisAuthMethod:sasl/DIGEST-MD5
SolarisTransportSecurity: ????
SolarisSearchReferral: false
SolarisSearchScope: sub
SolarisSearchTimeLimit: 30
cn: eng
ObjectClass: ?????
ObjectClass: ???????
```

2. **Save the generated profile to a file (such as profile.ldif) and use `ldapadd`(1) to store the client profile file in the LDAP server.**

```
# ldapadd -h ldap_server_hostname -D "cn=eng" \
-w nssecret -f profile.ldif
```

The `ldap_cachemgr`(1M) on every client machine automatically updates the content of the LDAP configuration files. This means changes need to be made only on the server and those changes automatically propagate to every client in the namespace in real time.

CHAPTER 16

LDAP Client Setup (Task)

This chapter describes how to set up a Solaris client to use the LDAP directory service. The instructions are not specfic to one type of directory server.

Setting Up an LDAP Client

`ldapclient` is a setup utility used to setup LDAP clients in a Solaris environment. `ldapclient`assumes the server has already been configured with the appropriate client profiles. In otherwords, *you must set up the server before you can set up any clients*. There are two ways to set up a client using `ldapclient()`:

- Static:

  You configure the profile on the client itself, which means defining all parameters form the command line. Thus, the profile information is stored on cache files and is never refreshed by the server.

- Profile:

  You only define the server address and [client] profile name on the command line. The server will provide the rest of the required information, except for proxy infomration. If a client's credential is defined a proxy, you must enter the proxy bind DN and password for each client. See Chapter 12 for more information.

Refer to the `ldapclient` (1M) man page for more detailed information on the LDAP utility.

▼ Create an LDAP Client using the 'profile' method with a credential level of 'anonymous'

1. **Become superuser.**
2. **Run** `ldapclient`**(1M).**

```
# ldapclient genprofile -a profileName=myprofile authenticationMethod=anonymous
```

`ldapclient` creates the configuration files and configures the client to use LDAP for name service lookups by modifying the `/etc/nsswitch.conf` file.

3. **Reboot the client.**

`ldaplist` Command

`ldaplist` is an LDAP utility to list the naming information from the LDAP servers. See `ldaplist`(1) for more info.

▼ List the Naming Information from the LDAP Servers

- **List the containers for the baseDN.**

```
# ldaplist hosts myhost
dn: cn=myhost+ipHostNumber=100.100.100.100,ou=Hosts,
dc=mkt,dc=mainstore,dc=com
```

Without any argument, `ldaplist` returns all the containers in the current search `baseDN`.

Adding a Network Printer

To add printer entries into the LDAP directory, use either the "printmgr" configuration tool or the `lpset -n ldap` command-line utility (See `lpset`(1M)). Keep in mind that when you add printer objects to the directory, they define the printer's connection parameters required by the print system clients. Local print server configuration data still resides in files. The following is an example of a typical printer entry:

```
printer-uri=myprinter,ou=printers,dc=mkg,dc=mainstore,dc=com
objectclass=top
objectclass=printerService
objectclass=printerAbstract
objectclass=sunPrinter
printer-name=myprinter
sun-printer-bsdaddr=printsvr.mainstore.com,myprinter,Solaris
sun-printer-kvp=description=HP LaserJet (PS)
printer-uri=myprinter
```

Listing Printer Entries Using `lpget`

Use the `lpget` command to list all printer entries contianed in the ldapclient's LDAP directory. Remember, if the particular client's LDAP server is a *replica* server, then list of printers in the replica's directory might or might not be up to to date (i.e. the master server might have yet to push the most recent printer inforamation to its slaves.) See`lpget`(1M) for more information on the update replication agreement.

The following illustrates how to list all printers for the base DN:

```
lpget -n ldap list
myprinter:
    dn=myprinter,ou=printers,dc=mkt,dc=mainstore,dc=com
    bsdaddr=printsvr.mainstore.com,myprinter,Solaris
    description=HP LaserJet (PS)
```

CHAPTER 17

Troubleshooting

This chapter describes configuration problems and suggested solutions.

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 backend is designed to return fully qualified hostnames for host lookups, such as hostnames returned by `gethostbyname`(3N) and `getipnodebyname`(3N). If the name stored is fully 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 fully qualified (it does not contain any dot), the client backend 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 domainname is different from the LDAP domainname, change the `nsswitch.conf` file. In the host entry, specify `dns`or put `dns` before `ldap`.

Sendmail Fails to Deliver/Receive Mail To/From Remote Users

If your mail domain (commonly the DNS domain) is different from the LDAP domain, you might run into a mail delivering problem. `sendmail`(1M) derives the mail domain from the domain portion of the hostname returned by `gethostname`(3N). This means the return address will be in the LDAP domain. Because the mail/DNS domain is different from the LDAP domain, external users cannot respond to the email To fix this problem, change the host entry in the `nsswitch.conf` file to `dns` or put `dns` before `ldap`.

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` does not exists as a source in the `/etc/nsswitch.conf` file
2. Password on the server 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
3. Incorrectly configured proxy agent causes authentication to fail.
4. The entry does not have the shadowAccount objectclass.

Lookup Too Slow

The LDAP database relies on indexes to improve the performance. A major performance degradation occurs when indexes are not configured properly. As part of the documentation, we have provided 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. There are several possible reasons for this failure

1. Check that the `ldap_cachemgr` is running (`ps -ef |grep ldap`) should show it running.
2. Try running `ldapclient list` to check out the contents of the LDAP client cached files.

Note – Do not try to read the configuration and credential files directly as there is no guarantee they are in ASCII readable format.

3. `nisDomain` attribute is not set in the DIT to represent the entry point for the specified client domain.
4. Virtual list view indexing is not set up properly on the server.
5. Access control information is not set up properly on the server; thus disallowing anonymous search in the LDAP database.
6. Incorrect server address passed to the `ldapclient` command. Use `ldapsearch`(1) to verify the server address
7. Incorrect profile name passed to the `ldapclient` command. Use `ldapsearch`(1) to verify the profile name in the DIT.
8. 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. Note that you do not need to become superuser to execute this command.

CHAPTER 18

LDAP Schemas (Reference)

To support Solaris operating environment naming clients, some Solaris specific schemas and some schemas defined by IETF are required.

IETF Schemas

LDAP requires two schemas defined by IETF: the revised RFC 2307 Network Information Service schema and the LDAP mailgroups Internet draft.

RFC 2307 Network Information Service Schema

The LDAP servers must be configured to support the revised RFC 2307.

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

The nisSchema OID is 1.3.6.1.1. The RFC 2307 Attributes are:

```
( 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' )
```

The nisSchema OID is 1.3.6.1.1. The RFC 2307 Objectclasses are:

```
( 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 ( l $ 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 $ l $ 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 )
```

Mail Alias Schema

The LDAP servers must be configured to support mail alias information. 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.

Note – Internet-Drafts are draft documents valid for a maximum of six months and might be updated, replaced, or obsoleted by other documents at any time

---

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:

```
( 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:

```
( 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 )
)
```

Solaris Schemas

The schemas required for the Solaris operating environment are the:

Solaris Projects schema.
Role based access control schema.
Solaris client naming profile schema.

Solaris Projects Schema

`/etc/project` is a local source of attributes associated with projects. For more information see `project`(4).

The Project Attributes are:

```
( 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:

```
( 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 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:

```
( 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 Objectclassses are:

```
( 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 ) )
```

Solaris Client Naming Profile Schema

`/etc/user_attr` is a local source of extended attributes associated with users, roles, and profiles. For more information see `user_attr`(4).

`/etc/security/prof_attr` is a local source for execution profile names, descriptions, and other attributes of execution profiles. For more information see `prof_attr`(4).

The Solaris client naming profile Attributes are:

```
( 1.3.6.1.4.1.42.2.27.5.1.15 NAME 'SolarisLDAPServers'
  DESC 'LDAP Server address eg. 76.234.3.1:389'
  EQUALITY caseIgnoreIA5Match
  SYNTAX SolarisLDAPServerSyntax)

( 1.3.6.1.4.1.42.2.27.5.1.16
  NAME 'SolarisSearchBaseDN'
  DESC 'Search Base Distinguished Name'
  EQUALITY distinguishedNameMatch
  SYNTAX DN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.17
  NAME 'SolarisCacheTTL'
  DESC 'TTL value for the Domain information eg. 1w, 2d, 3h, 10m, or 5s'
```

```
  EQUALITY caseIgnoreMatch
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.18
  NAME 'SolarisBindDN'
  DESC 'DN to be used to bind to the directory as proxy'
  EQUALITY distinguishedNameMatch
  SYNTAX DN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.19
  NAME 'SolarisBindPassword'
  DESC 'Password for bindDN to authenticate to the directory'
  EQUALITY caseExactIA5Match
  SYNTAX OctetString SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.20
  NAME 'SolarisAuthMethod'
  DESC 'Authentication method to be used eg. "NS_LDAP_AUTH_NONE",
       "NS_LDAP_AUTH_SIMPLE" or "NS_LDAP_AUTH_SASL_CRAM_MD5"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String)

( 1.3.6.1.4.1.42.2.27.5.1.21
  NAME 'SolarisTransportSecurity'
  DESC 'Transport Level Security method to be used eg.
        "NS_LDAP_SEC_NONE" or "NS_LDAP_SEC_SASL_TLS"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.22
  NAME 'SolarisCertificatePath'
  DESC 'Path to certificate file/device'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.23
  NAME 'SolarisCertificatePassword'
  DESC 'Password or PIN that grants access to certificate.'
  EQUALITY caseExactIA5Match
  SYNTAX OctetString SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.24
  NAME 'SolarisDataSearchDN'
  DESC 'Search DN for data lookup in "<database>:(DN0),(DN1),..." format'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String)

( 1.3.6.1.4.1.42.2.27.5.1.25
  NAME 'SolarisSearchScope'
  DESC 'Scope to be used for search operations eg.
           "NS_LDAP_SCOPE_BASE", "NS_LDAP_SCOPE_ONELEVEL" or
       "NS_LDAP_SCOPE_SUBTREE"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)
```

```
( 1.3.6.1.4.1.42.2.27.5.1.26
  NAME 'SolarisSearchTimeLimit'
  DESC 'Time Limit in seconds for search operations'
  EQUALITY integerMatch
  SYNTAX INTEGER SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.27
  NAME 'SolarisPreferredServer'
  DESC 'Preferred LDAP Server address or network number'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IAString)

( 1.3.6.1.4.1.42.2.27.5.1.28
  NAME 'SolarisPreferredServerOnly'
  DESC 'Boolean flag for use of preferredServer or not'
  EQUALITY booleanMatch
  SYNTAX BOOLEAN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.29
  NAME 'SolarisSearchReferral'
  DESC 'referral chasing option eg.
       "NS_LDAP_NOREF" or "NS_LDAP_FOLLOWREF"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)
```

The Solaris client naming profile Objectclasss is:

```
( 1.3.6.1.4.1.42.2.27.5.2.7 NAME 'SolarisNamingProfile'
  SUP top STRUCTURAL
  DESC 'Solaris LDAP Naming client profile objectClass'
  MUST ( cn $ SolarisLDAPServers $ SolarisSearchBaseDN )
  MAY ( SolarisBindDN $ SolarisBindPassword $ SolarisAuthMethod $
        SolarisTransportSecurity $ SolarisCertificatePath $
        SolarisCertificatePassword $ SolarisDataSearchDN $
        SolarisSearchScope $ SolarisSearchTimeLimit $
        SolarisPreferredServer $ SolarisPreferredServerOnly $
        SolarisCacheTTL $ SolarisSearchReferral )
    )
```

Internet Print Protocol (IPP) Attributes

```
ATTRIBUTE ( 1.3.18.0.2.4.1140
NAME printer-uri
DESC 'The URI supported by this printer.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
)
```

```
ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE
)

ATTRIBUTE ( 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 [RFC 1766] "Tags for
      the Identification of Languages".'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE
)


ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE
)


ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 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
      (subsequently).'
EQUALITY caseIgnoreMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{63}
SINGLE-VALUE
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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 [RFC 1766], Tags for the
      Identification of Languages.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{63}
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 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"
      (public domain ZIP), "gzip" (GNU ZIP), "compress" (UNIX).'
EQUALITY caseIgnoreMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{255}
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{255}
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{255}
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{255}
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 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
)

ATTRIBUTE ( 1.3.18.0.2.4.1112
NAME printer-current-operator
DESC 'The name of the current human operator responsible for
      operating this printer.  It is suggested that this string
      include information that would enable other humans to reach the
      operator, such as a phone number.'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
SINGLE-VALUE
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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}
)

ATTRIBUTE ( 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 caseIgnoreSubstringMatch
SYNTAX  1.3.6.1.4.1.1466.115.121.1.15{127}
)
```

Internet Print Protocol (IPP) ObjectClasses

```
OBJECTCLASS ( 1.3.18.0.2.6.2549
NAME  slpService
DESC  'DUMMY definition'
STRUCTURAL
SUP   top
)


OBJECTCLASS ( 1.3.18.0.2.6.254
NAME slpServicePrinter
DESC  'Service Location Protocol (SLP) information.'
AUXILIARY
SUP   slpService
)


OBJECTCLASS ( 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 )
)


OBJECTCLASS ( 1.3.18.0.2.6.255
NAME  printerService
DESC  'Printer information.'
STRUCTURAL
SUP   printerAbstract
MAY   ( printer-uri $ printer-xri-supported )
)


OBJECTCLASS ( 1.3.18.0.2.6.257
NAME  printerServiceAuxClass
DESC  'Printer information.'
AUXILIARY
SUP   printerAbstract
MAY   ( printer-uri $ printer-xri-supported )
)


OBJECTCLASS ( 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 )
)


OBJECTCLASS ( 1.3.18.0.2.6.253
NAME  printerLPR
DESC  'LPR information.'
AUXILIARY
SUP   top
MUST ( printer-name )
MAY ( printer-aliases)
)
```

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 ObjectClasses

```
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 )
)
```

Glossary

access rights The permissions assigned to classes of NIS+ principals that determine what operations they can perform on NIS+ objects: read, modify, create, or destroy.

application-level name service Application-level name services are incorporated in applications offering services such as files, mail, and printing. Application-level name services are bound below enterprise-level name services. The enterprise-level name services provide contexts in which contexts of application-level name services can be bound.

atomic name An FNS (XFN) term referring to the smallest indivisible component of a name as defined by the naming convention.

attribute In FNS (XFN), each named object is associated with a set of zero or more attributes. Each attribute in the set has a unique attribute identifier, an attribute syntax, and a set of zero or more distinct attribute values.

authentication The determination of whether an NIS+ server can identify the sender of a request for access to the NIS+ namespace. Authenticated requests are divided into the authorization categories of owner, group, and world. Unauthenticated requests—the sender is unidentified, are placed in the Nobody category.

binding In FNS (XFN), the association of an atomic name with an object reference. For simplicity, an object reference and the object it refers to are used interchangeably in this guide.

BNF An FNS (XFN) acronym referring to a Backus-Naur Form.

cache manager The program that manages the local caches of NIS+ clients (`NIS_SHARED_DIRCACHE`), 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.

child domain See *domain.*

checkpointing The process of writing changes to NIS+ data that are stored in server memory and recorded in the transaction log to the NIS+ tables stored on disk. In other words, updating the NIS+ tables with recent changes to the NIS+ data set.

client (1) In NIS+, the client is a principal (machine or user) requesting an NIS+ service from an NIS+ server.

(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.

(3) In the client-server model, the client is an *application* 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.

client-server model 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 *Domain Name System (DNS)* and file-server/file-client relationships such as *NFS* and diskless hosts. See also *client*.

cold-start file The NIS+ file given to a client when it is initialized that contains sufficient information so that the client can begin to contact the master server in its home domain.

composite name In FNS (XFN), a name that spans multiple naming systems. It consists of an ordered list of zero or more components. Each component is a name from the namespace of a single naming system. Composite name resolution is the process of resolving a name that spans multiple naming systems.

compound name In FNS (XFN), a sequence of atomic names composed according to the naming convention of a naming system.

context In FNS (XFN), an object whose state is a set of bindings with distinct atomic names. Every context has an associated naming convention. A context provides a lookup (resolution) operation, which returns the reference, and may provide operations such as binding names, unbinding names, and listing bound names.

credentials The authentication information about an NIS+ principal that the client software sends along with each request to an NIS+ server. This information verifies the identity of a user or machine.

data encrypting key A key used to encipher and decipher data intended for programs that perform encryption. Contrast with *key encrypting key*.

data encryption standard (DES) 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.

decimal dotted notation 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.

DES See *data encryption standard (DES)*.

directory (1) An NIS+ directory is a container for NIS+ objects such as NIS+ tables, groups, or subdirectories

(2) In UNIX, a container for files and subdirectories.

directory cache A local file used to store data associated with directory objects.

distinguished name 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.

DNS See *Domain Name System*.

DNS-forwarding An NIS server or an NIS+ server with NIS compatibility set forwards requests it cannot answer to DNS servers.

DNS zones Administrative boundaries within a network domain, often made up of one or more subdomains.

DNS zone files A set of files wherein the DNS software stores the names and IP addresses of all the workstations in a domain.

domain (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.

- *Parent domain*. Relative term for the domain immediately above the current domain in the hierarchy.
- *Child domain*. Relative term for the domain immediately below the current domain in the hierarchy.
- *Root domain*. Highest domain within the current NIS+ hierarchy.

(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).
domain name	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 *domain*.
Domain Name Service (DNS)	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.
encryption key	See *data encrypting key*.
enterprise-level name service	An enterprise-level naming service identifies (names) machines (hosts), users and files within an enterprise-level network. FNS also allows naming of organizational units, geographic sites, and application services.
enterprise-level network	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.
enterprise root	In FNS (XFN), the root context of an enterprise. A context for naming objects found at the root of the enterprise namespace.
entry	A single row of data in a database table.
federated naming service	The service offered by a federated naming system.
federated naming system	An aggregation of autonomous naming systems that cooperate to support name resolution of composite names through a standard interface. Each member of a federation has autonomy in its choice of operations other than name resolution.
federated namespace	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.
FNS	See *Federated naming service*.
generic context	In FNS (XFN), a context for binding names used in applications.
GID	See *group ID*.
global context	In FNS (XFN), a context for naming objects that have global names (currently, DNS and X.500 are the only global naming systems specified by XFN).

global name service 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.

group (1) A collection of users who are referred to by a common name.

(2) In NIS+ a collection of users who are collectively given specified access rights to NIS+ objects. NIS+ group information is stored in the NIS+ group table.

(3) In UNIX, groups determine a user's access to files. There are two types of groups: default user group and standard user group.

group ID A number that identifies the default *group* for a user.

host context In FNS (XFN), a context for naming objects related to a computer.

implicit naming system pointer An FNS (XFN) term referring to an unnamed reference that points to a context in another naming system.

indexed name A naming format used to identify an entry in a table.

initial context In FNS (XFN), every XFN name is interpreted relative to some context, and every XFN naming operation is performed on a context object. The XFN interface provides a function that allows the client to obtain an initial context object that provides a starting point for resolution of composite names.

initial context function An FNS function, `fn_ctx_handle_from_initial()`, that obtains the initial context which allows a client to obtain an initial starting point for name resolution.

Internet The world-wide collection of networks interconnected by a set of routers that enable them to function and communicate with each other as a single virtual network.

Internet address A 32-bit address assigned to hosts using *TCP/IP*. See *decimal dotted notation*.

IP Internet Protocol. The *network layer* protocol for the Internet protocol suite.

IP address A unique number that identifies each host in a network.

junction An FNS (XFN) term referring to a name in one namespace bound to a context in the next naming system.

key (column) An NIS+ table entry's data can be accessed from any column, regardless of that table's key.

key (encrypting) A key used to encipher and decipher other keys, as part of a key management and distribution system. Contrast with *data encrypting key*.

key server A Solaris operating environment process that stores private keys.

local-area network (LAN) Multiple systems at a single geographical site connected together for the purpose of sharing and exchanging data and software.

mail exchange records Files that contain a list of DNS domain names and their corresponding mail hosts.

mail hosts A workstation that functions as an email router and receiver for a site.

master server The server that maintains the master copy of the network information service database for a particular domain. Namespace changes are always made to the name service database kept by the domain's master server. Each domain has only *one* master server.

MIS Management information systems (or services)

naming convention In FNS (XFN), every name is generated by a set of syntactic rules called a naming convention.

name resolution The process of translating workstation or user names to addresses.

name server Servers that run one or more network name services.

name service switch A configuration file (`/etc/nsswitch.conf`) that defines the sources from which an NIS+ client can obtain its network information.

name service A network service that handles machine, user, printer, domain, router, an other network names and addresses.

namespace (1) A namespace stores information that users, workstations, and applications must have to communicate across the network.

(2) The set of all names in a naming system.

(3) NIS+ namespace, A collection of hierarchical network information used by the NIS+ software.

(4) NIS namespace. A collection of *non*-hierarchical network information used by the NIS software.

(5) DNS namespace. A collection of networked workstations that use the DNS software.

namespace identifier An FNS (XFN) term referring to a special atomic name used to refer to the root of a namespace.

naming system In FNS (XFN), a connected set of contexts of the same type (having the same naming convention) and providing the same set of operations with identical semantics. In the UNIX operating environment, for

example, the set of directories in a given file system (and the naming operations on directories) constitutes a naming system.

network mask A number used by software to separate the local subnet address from the rest of a given Internet protocol address.

next naming system pointer (NNSP) In FNS (XFN), a reference to a context in which composite names from subordinate naming systems are resolved.

network password See Secure RPC password.

NIS A distributed network information service containing key information about the systems and the users on the network. The NIS database is stored on the *master server* and all the *replica* or *slave servers*.

NIS maps 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 *NIS*.

NIS+ A distributed network information service containing hierarchical information about the systems and the users on the network. The NIS+ database is stored on the *master server* and all the *replica server*s.

NIS-compatibility mode 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.

NIS+ environment For administrative purposes, an NIS+ environment refers to any situation in which the applicable `nsswitch.conf` file points to `nisplus`. Or any time a command is run with an option that forces it to operate on objects in an NIS+ namespace (for example, `passwd -r nisplus`).

NIS+ object An NIS+ domain, directory, table, or group. See *domain*, *directory*, *group*, and *table*.

NIS+ principal See *principal*.

NIS+ transaction log A file that contains data updates destined for the NIS+ tables about objects in the namespace. Changes in the namespace are stored in the transaction log until they are propagated to replicas. The transaction log is cleared only after all of a master server's replicas have been updated.

NNSP See *next naming system pointer*.

organizational units In FNS (XFN), an enterprise is organized into organizational units such as centers, laboratories, departments, divisions, and so on. An organizational unit is a subunit of an enterprise.

organizational unit context In FNS (XFN), a context for naming objects related to an organizational unit within an enterprise.

parent context In FNS (XFN), a context in which this context and its siblings are bound.

parent domain See *domain*.

pinging The process by which an NIS+ master server transfers a change a NIS+ data to the domain's replica servers.

preference rank number A number which a machine uses to rank the order in which it tries to obtain namespace information from NIS+ servers. A machine will first try all servers with a given rank number before trying any server with the next highest rank number. For example, a machine will query NIS+ servers with a rank number of 0 before it queries any server with a rank number of 1.

preferred server From the point of view of a client machine, a preferred NIS+ server is a server that the client should try to use for namespace information ahead of non-preferred servers. Servers that are listed in a client or domain's preferred server list are considered preferred servers for that client or domain.

preferred server list A `client_info` table or a `client_info` file. Preferred server lists specify the preferred servers for a client or domain.

principal Any user of NIS+ information whose credentials have been stored in the namespace. Any user or machine that can generate a request to a NIS+ server. There are two kinds of NIS+ principal: client users and client machines:

- *Root principal*. A machine root user (user ID = 0). Requires only a DES credential.
- *User principal*. Any nonroot user (user ID > 0). Requires local and DES credentials.

private key 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.

public key 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.

populate tables Entering data into NIS+ tables either from files or from NIS maps.

record See *entry*.

reference An FNS (XFN) term referring to the thing bound to a name. It contains addresses identifying the communication endpoints of the object.

remote procedure call (RPC) 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.

replica server NIS+ server that maintains a duplicate copy of the domain's master NIS+ server database. Replicas run NIS+ server software and maintain copies of NIS+ tables. A replica server increases the availability of NIS+ services. Each NIS+ domain should have at least one, and perhaps more, replicas. (In an NIS namespace, a replica server was known as a *slave* server.)

reverse resolution The process of converting workstation IP addresses to workstation names using the DNS software.

root context In FNS (XFN), a context for naming the objects found in the root of the namespace.

root domain See *domain*.

root master server The master server for a NIS+ root domain.

root replica server NIS+ server that maintains a duplicate copy of the root domain's master NIS+ server database.

RPC See remote procedure call (RPC).

Secure RPC password 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.

server (1) In NIS+, NIS, DNS, and FNS (XFN) a host machine providing naming services to a network.

(2) In the *client-server model* 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.

(3) A *daemon* that actually handles the providing of files.

server list *See* preferred server list.

service context In FNS (XFN), a context for naming objects that provide services.

site context In FNS (XFN), a context for naming objects related to a physical site.

slave server	(1) A server system that maintains a copy of the NIS database. It has a disk and a complete copy of the operating environment. (2) Slave servers are called *replica servers* in NIS+.
strong separation	An FNS (XFN) term referring to cases where the XFN context treats the XFN component separator as the naming system boundary.
subcontext	In FNS (XFN), a context bound within another context.
subnet	A working scheme that divides a single logical network into smaller physical networks to simplify routing.
table	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.
TCP	See *Transport Control Protocol (TCP)*.
TCP/IP	Acronym for Transport Control Protocol/Interface Program. The protocol suite originally developed for the Internet. It is also called the *Internet* protocol suite. Solaris networks run on TCP/IP by default.
Transport Control Protocol (TCP)	The major transport protocol in the Internet suite of protocols providing reliable, connection-oriented, full-duplex streams. Uses IP for delivery. See TCP/IP.
user context	In FNS (XFN), a context for naming objects related to a human user.
weak separation	An FNS (XFN) term referring to cases where the XFN context does not treat the XFN component separator as the naming system boundary.
wide-area network (WAN)	A network that connects multiple local-area networks (LANs) or systems at different geographical sites via phone, fiber-optic, or satellite links.
XFN link	In FNS (XFN), a special form of reference that has a composite name as an address. Like any other type of reference, an XFN link is bound to an atomic name in a context.
X.500	A global-level directory service defined by an Open Systems Interconnection (OSI) standard.

Index

D

E

F

G

H

I

J

K

N

O

P

R

S

T

U

W

X

Y

Z