NIS Administrator’s Guide

Document Number 007-2161-006

CONTRIBUTORS
Written by Pam Sogard and Susan Ellis

Updated by Julie Boney

Edited by Susan Wilkening

Production by Diane Ciardelli

Engineering contributions by Chris Duffy, Ron Short, Ron Susztar, Barry Kirkpatrick

New Features in This Guide

This update of the NIS Administrator's Guide supports the 6.5.12 release of IRIX. Changes include the following:

- Added `yp` and `ypserv` flags to list of flags under "Daemons" on page 10
- Added an alternative to the method of obtaining the IP address and port number for the NIS server process under "Binding" on page 11
- Added `shadow` and `jlimits` to the list of NIS maps under "NIS Database" on page 13
- Added the location for the `nsswitch.conf` file for clients under "Global Files" on page 18
- Added the `ypmaster.options` file to the example of files that support the NIS password file under "Selecting the NIS Master Server" on page 30
- Added an alternative method for specifying an NIS server under "Specify an NIS Server at Client Startup" on page 31
- Added two more steps to the procedure for setting up NIS slave servers under "Setting Up NIS Slave Servers" on page 37
- Added a method for updating the password more quickly under "Changing NIS Passwords" on page 46
- Added a note about size limits under "Using Netgroups" on page 47

Record of Revision

Version	Description
004	July 1999 Incorporates information for the IRIX 6.5.5 release
005	July 2000 Updates information for the IRIX 6.5.9 release
006	March 2001 Updates information for the IRIX 6.5.12 release

Contents

List of Figures

List of Tables

About This Guide

The *NIS Administrator's Guide* documents the SGI implementation of the network information service (NIS).

NIS is a database service that provides location information about network entities to other network services, such as the Network File System (NFS). Systems with heterogeneous architectures and operating systems can participate in the same NIS. The service can also include systems connected to different types of networks.

This guide was formerly published under the title *NFS and NIS Administration Guide and Man Pages*, and documented NFS as well as NIS. You can now find information about NFS in a separate volume titled *ONC3/NFS Administrator's Guide*.

Using This Guide

This guide provides information you need to set up and maintain NIS. It explains the software fundamentals of the product and provides procedures to help you install, test, and troubleshoot NIS on your network. It also contains recommendations for planning and administering NIS.

Summary of Contents

Table i contains a summary of each chapter in this guide and suggests how to use the chapter.

Table i Contents of Each Chapter

Chapter	**Summary**	**When to Read**
Chapter 1, "Understanding NIS"	Introduces the vocabulary of NIS, describes the relationship of NIS to other network software, and explains how NIS domains are organized.	Read this chapter to learn NIS basics. If you are already experienced with NIS, you can skip Chapter 1.
Chapter 2, "Preparing to Manage NIS"	Describes the fundamental operation of NIS and its database.	Read this chapter for the background required to do the procedures in Chapter 4, "Setting Up and Testing NIS."
Chapter 3, "Planning Your NIS Service"	Presents the issues you need to consider before you implement NIS for your site and offers planning recommendations.	Review this chapter before setting up NIS on your network.
Chapter 4, "Setting Up and Testing NIS"	Contains procedures for implementing NIS on server and client systems and procedures for verifying their operation.	Use this chapter as a guide through NIS setup tasks.
Chapter 5, "Maintaining NIS"	Explains how to change NIS and its database when conditions in your network change. It also contains information on managing security with NIS.	Refer to this chapter when you need to update NIS maps, implement security, or add new users to NIS.
Chapter 6, "Troubleshooting NIS"	Describes problems that can arise when maps are propagated and when NIS server or client software is malfunctioning. Recommends corrective action for each type of problem.	Use this chapter to identify the source of NIS problems and take corrective action. Read the information in the final section before phoning the Silicon Graphics Technical Assistance Center.

Audience for This Guide

To use NIS setup and maintenance information, you should have experience in these areas:

- Setting up network services
- Assessing the needs of network users
- Maintaining hosts databases
- Understanding the UNIX filesystem structure
- Using UNIX editors

To troubleshoot NIS, you should be familiar with these concepts:

- Theory of network services
- SGI network implementation

Related Publications

You can find supplementary information in these documents and books:

- *IRIX Admin: Networking and Mail* (SGI publication) explains the fundamentals of system and network administration for SGI systems on a local area network.
- *ONC3/NFS Administrator's Guide* (SGI publication) explains how to set up and maintain the SGI implementation of NFS.
- *IRIX Admin: System Configuration and Operation* (SGI publication) explains how to set up configuration files.
- *Personal System Administration Guide* explains how to add entries to the NIS password file.

  To obtain SGI documentation, go to the SGI Technical Publications Library at `http://techpubs.sgi.com.`
- Stern, Hal, *Managing NFS and NIS*, O'Reilly & Associates, Inc. 1991. This book contains detailed, but not SGI-specific, information about NIS and how to administer and use it.

Typographic Conventions

This guide uses the following font conventions:

italics	Italic font is used for variables.
`Courier`	Courier font is used for commands, text that you are to type literally, examples of system output, and the contents of files.

Product Support

SGI offers a comprehensive product support and maintenance program for IRIX products. For information about using support services for this product, refer to the release notes that accompany it.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. Be sure to include the title and part number of the document with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, the document number can be found on the back cover.)

You can contact us in any of the following ways:

- Send e-mail to the following address:

  `techpubs@sgi.com`

- Use the Feedback option on the Technical Publications Library World Wide Web page:

  `http://techpubs.sgi.com`

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

  Technical Publications
  SGI
  1600 Amphitheatre Pkwy.
  Mountain View, California, 94043-1351

- Send a fax to the attention of “Technical Publications” at:

  +1 650 932 0801

We value your comments and will respond promptly.

Chapter 1

Understanding NIS

This chapter contains a general description of the SGI implementation of the Sun Microsystems network information service NIS. It provides an overview of NIS, an explanation of the NIS client-server model, and an introduction to NIS domains and NIS maps.

This chapter contains these sections:

- “About NIS” on page 2
- “NIS Portability” on page 2
- “Client-Server Model” on page 4
- “Server Hierarchy” on page 4
- “NIS Maps” on page 4
- “NIS Domains” on page 5

About NIS

NIS is a network lookup service that provides a centralized database of information about the network to systems participating in the service. The NIS database is fully replicated on selected systems and can be queried by participating systems on an as-needed basis. Maintenance of the database is performed on a central system.

The purpose of NIS is to make network administration more efficient by reducing the risk of error and the time required to perform redundant file management tasks. For example, maintaining the `/etc/hosts` database on a large network might require creating a script to automatically copy the `/etc/hosts` file from a central system to all systems on the network. It also requires setting up the appropriate access permissions on each system to enable this file transfer; this is a redundant and time-consuming process. By contrast, on networks using NIS, maintaining the `/etc/hosts` database requires modifying a single file, typically `/etc/hosts`, on a single system.

On many networks, a number of information sources are available to provide information to network applications. For this reason, most applications have a standard lookup rule for finding the information they need. Starting with IRIX 6.5, the default lookup order is specified in the `/etc/nsswitch.conf` file.

NIS can service networks with approximately 1000 systems. Larger networks can be organized into multiple NIS service areas, or domains.

NIS Portability

NIS is an application layer service that can be used on any network using the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) for transport layer services. NIS also relies on remote procedure call (RPC) for session layer services and external data representation (XDR) for presentation layer services. Because the NIS application adheres to these standard network protocols, it is portable and works with a variety of vendors’ platforms.

The network protocols TCP and UDP provide the services required to transport messages on the same system or between remote systems. TCP provides reliable, connection-oriented transport. UDP provides unreliable, connectionless transport. TCP and UDP are protocols that are widely accepted and used in the network environment, making them the logical choices for NIS and several other network applications.

RPC and XDR are services that bridge the gap between the transport layer services and the network application. They provide the functionality required to build distributed applications and resolve operating system and hardware architectural differences.

RPC provides a message passing scheme. It allows information to be passed between procedure calls that do not reside in the same address space. The address space can be located on the same system or it may reside on a remote system. In the NIS application, RPC enables client and server processes on local or remote systems to access the NIS database. NIS users are not aware that the procedure calls are occurring between two different systems.

XDR translates differences that can occur between machine architectures. It allows remote procedure calls and/or an application to interpret an RPC message independent of machine architecture. In the NIS application, XDR services allow systems from multiple vendors access to an NIS database located on any vendor's system. RPC and XDR are not exclusive to NIS. RPC and XDR are industry standards and are used in a variety of distributed network applications.

Figure 1-1 illustrates the NIS software implementation in the context of the Open Systems Interconnect (OSI) model.

application	**NIS**
presentation	**XDR**
session	**RPC**
transport	**UDP/TCP**
network	**IP**
data link	**network interface**
physical	

Figure 1-1 NIS Software Implementation

Client-Server Model

An NIS client is a process running on a system that requests data from an NIS database. An NIS server is a process running on a system that provides data from the NIS database. The terms client and server designate both processes and systems: a system is considered a client when requesting NIS data, and it is considered a server when providing NIS data. A system can function as a client and a server simultaneously.

Sometimes client requests are handled by NIS servers running on the same system, and sometimes they are serviced by NIS servers running on a different system. If one NIS server system fails, client processes obtain NIS services from another. In this way, the NIS service remains available even when an NIS server system goes down.

Server Hierarchy

NIS servers, each of which contains a copy of the NIS database, are divided into two groups: master servers and slave servers. A master server is the system on which NIS databases are originally created and maintained. A slave server is a system that holds a duplicate copy of the database. A server may be a master server with respect to one database and a slave server with respect to another.

The master server makes updated database information available to slave servers by a process known as propagation. Propagation ensures the consistency of database information between the master server and its slave servers.

NIS Maps

The NIS database is composed of a group of files known as maps. Maps are created with NIS tools that convert input files (usually standard ASCII files) to files in database record format (see the `mdbm`(3B) man page). Since data in `mdbm` format is faster to find than ASCII data, using `mdbm` files increases NIS performance.

Each NIS map has a map name that programs use to access it. Any program using an NIS map must recognize the format of the data it contains.

Maps are composed of keys and values. A key is a particular field in the map that the client must specify whenever it queries the map; a value is an attribute of the key that is returned from the query. For example, in the map called `hosts.byname`, the keys are the names in individual systems, and the values are the `ip-address,hostname` lines, similar to those in `/etc/hosts`.

At steady state, maps throughout the network contain consistent information. In this state, a client query receives the same answer to the query, regardless of which server responds to it. This consistency of information allows multiple servers to operate on a network, increasing the availability and reliability of the NIS service.

NIS Domains

An NIS domain is a collection of systems using the same NIS database. To participate in the NIS service, a system must belong to an NIS domain.

Figure 1-2 shows the basic layout for the systems in Building 1 and a domain called `eng`.

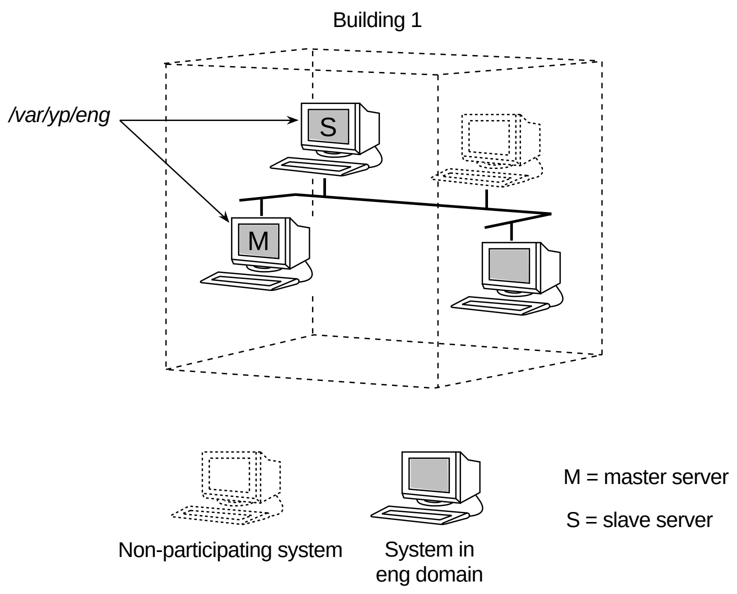

Figure 1-2 Basic NIS Domain

The domain `eng` consists of the master server, one slave server, and three clients. One system on the network does not participate in NIS at this time but may be included in the domain at a later date.

NIS Domains and Server Directories

The NIS databases are contained in subdirectories of the NIS home directory `/var/ns/domains`. These subdirectories are named for the domain whose database they contain. For example, in Figure 1-2, both servers contain the database for the `eng` domain in a subdirectory named `/var/ns/domains/eng`.

NIS and Internet Domains

The Internet is a registered organization of wide area networks originally funded by DARPA (U.S. Defense Advanced Research Projects Agency). It is organized into domains, machines grouped into networks that are given names to identify them clearly. In the Internet naming scheme, commercial businesses in the United States are given names that end in `.com` (`sgi.com` is one such), educational institutions use `.edu`, and governmental organizations use domain names ending in `.gov`.

Domains within an organization are organized on the same grouping principle. For example, in a business with the Internet domain name of `dender.com`, two separate NIS domains might be finance and engineering, which would use the domain names `.finance` and `.eng`, respectively. If you subdivided `dender.com` this way, the NIS domain name of engineering would be `eng.dender.com`, and the NIS domain name of `finance` would be `finance.dender.com`.

Often the Internet domain name is used as the basis for the NIS domain, a useful practice, but not required. You can use some other NIS domain name, but you should understand fully about domain names and their interaction with other aspects of name service before doing so. For further details on domains, refer to *IRIX Admin: Networking and Mail*.

Multiple NIS Domains

Complex networks that require large NIS databases, approximately 1000 systems or more, should be evaluated for multiple NIS domains. Factors that should be considered when determining whether to have multiple domains are network complexity and server availability. In addition, on networks where dynamic conditions make database synchronization difficult, multiple domains can make NIS more reliable and easier to maintain. NIS domains are not constrained by the topology of the network. Systems in the same NIS domain are not necessarily on the same local area network. For administrative or organizational reasons, it may make sense to configure large networks as separate NIS domains such as eng and finance.

Figure 1-3 illustrates the organization of Building 1 and Building 2 local area networks into two domains, eng and finance.

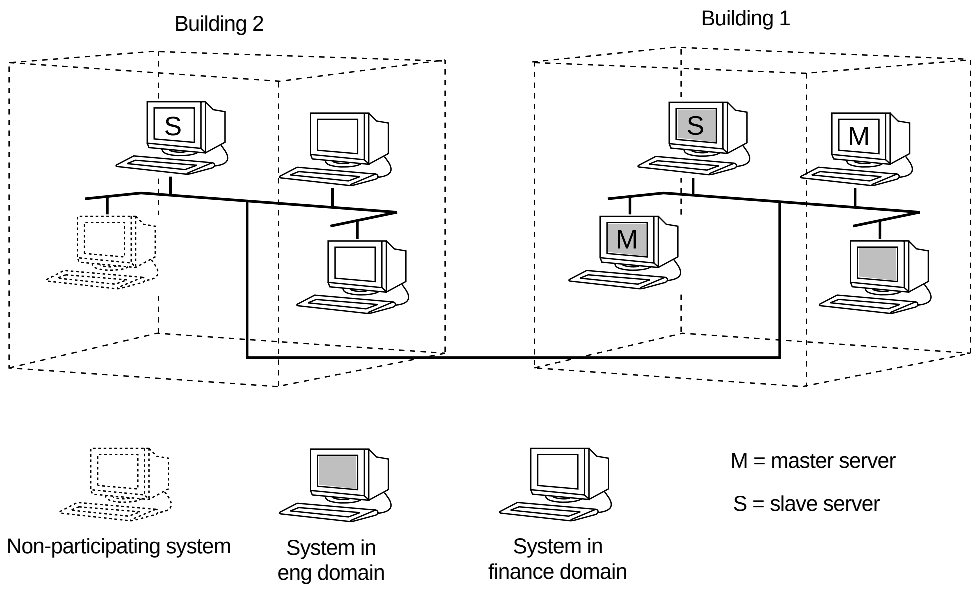

Figure 1-3 Multiple NIS Domains

The master and slave server for the `eng` domain both contain a database of information for the `eng` domain in `/var/ns/domains/eng`, and the master and slave server for the `finance` domain both contain a copy of the database for the `finance` domain in `/var/ns/domains/finance`. Notice that one system in the Building 1 local area network belongs to the `finance` domain and is the master server for the `finance` domain. (Chapter 2, “Preparing to Manage NIS,” contains detailed information on multiple NIS domains.)

Chapter 2

Preparing to Manage NIS

To be prepared for managing NIS, you should understand NIS software elements and the tools available for controlling its operation. This chapter contains the prerequisite information. It identifies NIS client and server daemons and their interactions, and describes a special daemon interaction called binding. It also explains how the NIS database is created and maintained, and how local client files and global files are used when NIS is in effect. Finally, this chapter provides a quick reference guide to NIS software and NIS management tools.

This chapter contains these sections:

Daemons

Which NIS daemons are running on a system depends on the system's function in the NIS environment: clients, master servers, and slave servers each run a particular set of daemons.

Table 2-1 lists the daemons required for each type of system for NIS to function correctly.

Table 2-1 NIS Daemons by System Type

Daemon	Client	Slave	Master
`nsd`	X	X	X
`rpc.passwd`			X

The binder daemon, `nsd`, runs on all NIS clients. In this instance, the daemon is responsible for remembering information necessary for communicating with the NIS server process. See the `nsd`(1M) man page for more information.

The `nsd` daemon also acts as the server daemon and runs on all NIS servers. It acts as the database server and is responsible for answering client inquiries and managing database updates. Most NIS servers are also NIS clients; they use the NIS database information.

On the NIS master server, the server process daemon, `nsd`, runs to answer client inquiries and to solicit information from the NIS database. The master server also runs a second daemon, `/usr/etc/rpc.passwd`, which allows NIS users to remotely modify their NIS password by using `yppasswd` and to modify some other password file fields by using `ypchpass`. For more information, see the `yppasswd`(1) and `ypchpass`(1) man pages.

On IRIX, NIS daemons are started by the master network script, `/etc/init.d/network`, if the NIS daemon flags are set on (flags can be set by using the `chkconfig` command). The `chkconfig` flags for NIS are `nsd`, `yp`, `ypserv`, and `ypmaster` (see Chapter 4, "Setting Up and Testing NIS," for more details).

Binding

In binding, a process remembers the address at which the NSD server process is listening for requests. In the NIS environment, when an application on a client needs information that is normally derived from certain local files, the application solicits the information from the NIS database on a selected NIS server. The relationship between the binder daemon, and the server daemon, determines whether or not an NIS connection is bound or unbound. A brief summary of the binding process is given below.

To obtain the IP address and port number for the NIS server process, NSD broadcasts for any NIS server within its domain. The first NIS server process to respond with its IP address and port number, whether local or remote, is the process that is used to service the request. The IP address for the physical NIS server and the port number for the NIS server process are remembered by the NSD process and used to obtain NIS database information. The alternative is to supply a list of host names or IP addresses for `nsd` to bind to. For more information, see the `/etc/config/nsd.options` and `/etc/nsswitch.conf` files.

Figure 2-1 illustrates the binding process initiated for an `ls` command. Before the `ls` command can list the contents of a directory, it needs to translate the file's user ID into a user's name. `ls` uses the library routine `getpwuid`, which accesses the local `/etc/passwd` file and the NIS password file as appropriate. In an NIS environment, this entails accessing the password map in the NIS database. Note that the general process is the same whether binding occurs on the local system or between remote systems. For more information, see the `ls`(1) and `getpwuid`(1) man pages.

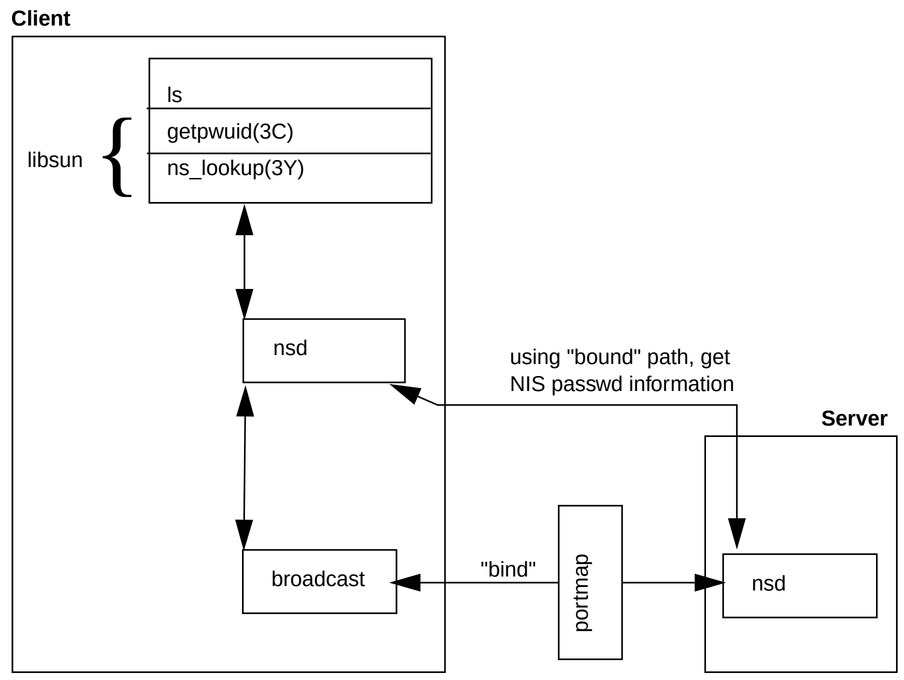

Figure 2-1 NIS Binding Process

When a client boots, NSD broadcasts or multicasts, by default, to the portmap port number for the NIS service. The portmapper forwards the packet to the NIS server, if there is one running on the machine, which then determines whether or not it services the domain requested. Similarly, NSD broadcasts, asking for a new NIS server if the old server fails to respond, or it selects the next one on the list, determined by the contents of the `/etc/config/NSD.options` or the `/etc/nsswitch.conf` files. An `nsd` daemon runs on both the client and the server. The `ypwhich`(1) command gives the name of the server to which NSD is currently bound.

NIS Database

The NIS database is a collection of files in `mdbm` format. To create the database, the NIS tool `makemdbm` converts input files (usually ASCII) to output files. The output files have `.m` extensions. Each is a map. For example, the `aliases` map is composed of the file `aliases.m`.

A typical listing of NIS database files looks like this:

```
bootparams.m          mac.byname.m          passwd.byuid.m
capability.byname.m   mac.byvalue.m         protocols.byname.m
clearance.byname.m    mail.aliases.m        protocols.bynumber.m
ethers.byaddr.m       mail.byaddr.m         rpc.byname.m
ethers.byname.m       netgroup.byhost.m     rpc.bynumber.m
group.bygid.m         netgroup.byuser.m
group.byname.m        netid.byname.m
group.bymember.m      networks.byaddr.m
hosts.byaddr.m        networks.byname.m
hosts.byname.m        passwd.byname.m
```

Standard and Nonstandard Maps

The NIS application is capable of making and updating a particular set of maps automatically. These are known as standard maps and are derived from regular ASCII files. The maps included in a standard set vary with each NIS release. Nonstandard maps are maps that have no ASCII form or maps that are created for vendor- or site-specific applications; NIS does not automatically know how to make or update nonstandard maps. NIS can serve any number of standard (default) and nonstandard maps. Following is a list of standard MIS maps:

```
bootparams         hosts.byaddr   netgroup.byhost   protocols.bynumber
capability.byname  hosts.byname   netgroup.byuser   rpc.byname
clearance.byname   jlimits        netid.byname      rpc.bynumber
ethers.byaddr      mac.byname     networks.byaddr   services
ethers.byname      mac.byvalue    networks.byname   services.byname
group.bygid        mail.aliases   passwd.byname     shadow
group.byname       mail.byaddr    passwd.byuid      ypservers
group.bymember     netgroup       protocols.byname
```

In most cases, the format of the data in NIS default maps is identical to the format within the ASCII files.

Some maps have default nicknames to make administration easier. The `ypcat`(1) command, a general NIS database print program, with the `–x` option prints a list of default map nicknames and their corresponding full names. Table 2-2 shows the list of default nicknames and full names for maps supplied in the NIS release.

Table 2-2 Default Nicknames for Maps

Map Nickname	Map Full Name
`aliases`	`mail.aliases`
`ethers`	`ethers.byname`
`group`	`group.byname`
`hosts`	`hosts.byaddr`
`networks`	`networks.byaddr`
`passwd`	`passwd.byname`
`protocols`	`protocols.bynumber`
`rpc`	`rpc.bynumber`
`services`	`services.byname`

For example, the command `ypcat hosts` is translated into `ypcat hosts.byaddr` because there is no map called `hosts`.

Map Propagation

Propagating an updated database from master server to slave servers ensures database consistency between all NIS clients. Databases can be updated in two ways: periodically using the `crontab` command and interactively from the command line (see Chapter 5, “Maintaining NIS,” for details on map propagation methods).

The propagation process varies depending on the propagation method. For example, when a map is updated and propagated using the `ypmake` command, `ypmake` looks at `mdbm_parse` to determine which maps to make. The `mdbm_parse command` updates the maps and calls `yppush`. The `yppush` command reads the `ypservers` map to determine which slave servers to contact; `yppush` contacts NSD on the selected slave servers and requests `ypxfr` service. The slave server can now transfer the maps using `ypxfr`. For more information on map propagation methods, see the `cron`(1), `ypmake`(1M), `yppush`(1M), and `ypxfr`(1M) man pages.

Figure 2-2 illustrates the propagation process between a master server and a slave server using `ypmake`.

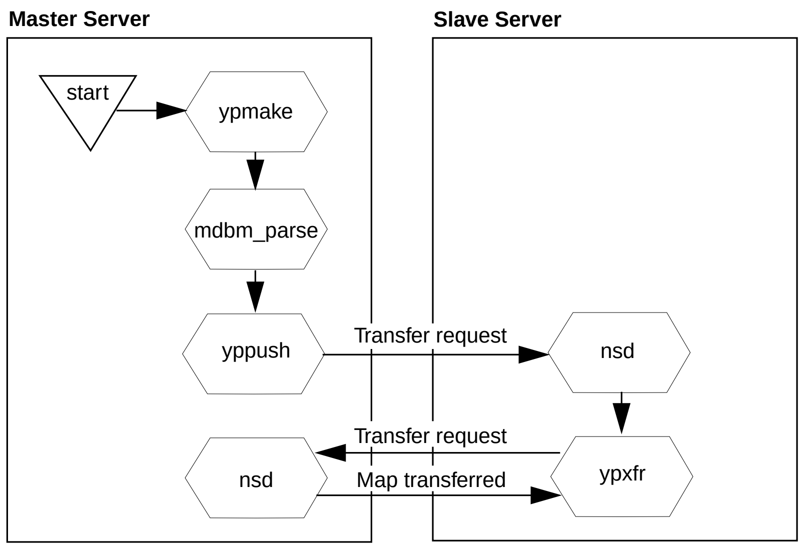

Figure 2-2 Map Propagation Between Servers

NIS and Other Network Files

Network files under system control can be divided into two groups: local files and global files. Local files are those that NIS first checks on the local system and may continue checking in the NIS database. Global files reside in the NIS database and are always consulted by programs using NIS. The level of system control over some files depends on the NIS syntax used within those files.

The next two sections discuss the local and global files consulted by NIS. More information on these configuration files is included in *IRIX Admin: System Configuration and Operation*.

Local Files

Table 2-3 shows the local files that NIS consults, the use of which is determined by how they are ordered in `/etc/nsswitch.conf`. These files can be controlled at different levels. Two special cases, however, are `/etc/aliases` and `/etc/passwd`.

The `/etc/aliases` and `/etc/passwd` files may contain a special cookie starting with a plus sign (+). This directs the files parser to insert data from subsequent libraries at that point. This replacement is done in the files protocol library, but only if the NSD attribute `compat` is set in the `nsswitch.conf` file.

For example, a program that calls `getpw ent` to access `/etc/passwd` (a local file) first looks in the password file on your system (if there is a password entry in the `nsswitch.conf` file); the NIS password file is consulted only if your system's password file contains this plus sign (`+`) entry (see the `passwd` man page).

Table 2-3 Local Files Consulted by NIS

Local File
`/etc/hosts.equiv`
`/etc/passwd`
`/etc/aliases`
`.rhosts`

Table 2-4 shows some examples of plus (+) and minus (–) entries for the local `/etc/group` and `/etc/passwd` files. Note that the position of + and – entries in the files affects processing. The first entry (+, –, or regular) is the one that is used.

Table 2-4 Local File Entries to Control Access

Local File	Example Entry	Meaning of the Entry
`/etc/passwd`	`+:`	Get all password information from the NIS password database.
	`+gw:`	Get all user account information for `gw` from NIS.
	`+@marketing:`	Allow anyone in the `marketing` netgroup (see "Using Netgroups" in Chapter 5 for details) to log in using NIS account information.
	`+nb::::Nancy Doe:/usr2/nb:/bin/csh` (shown wrapped; entry is one line)	Get the user password, user ID, and group ID from NIS. Get the user's name, home directory, and default shell from the local entry.
	`-fran:`	Get all user account information from NIS and disallow any subsequent entries (local or NIS) for `fran`.
	`-@engineering:`	Disallow any subsequent entries (local or NIS) for all members in the netgroup `engineering`.

In the `/etc/hosts.equiv` file, if there are + or `–` entries whose arguments are @ symbols and netgroups, the NIS `netgroup` map is consulted; otherwise NIS is not consulted. This rule also applies to the `.rhosts` file.

In `/etc/aliases`, if there is a +:+ entry, the NIS `aliases` map is consulted. Otherwise, NIS is not consulted.

Global Files

All global files are controlled by the `/etc/nsswitch.conf` file, which determines the maps, the methods, and the order in which they are looked up. The compatibility attribute to override local control of a file is set in the following manner in the `nsswitch.conf` file:

```
passwd:                 files(compat) [notfound=return] nis
```

This line compels files to be searched in the historical manner: the files are parsed and if a +/- entry is found, the next element is called. If the requested item is not found in the file, either as a regular entry or as one of the + or - entries, then control is returned immediately, without notification, to the next name service.

For example, previously `ypserv` had a flag `-i` pertaining to the hosts map, which meant if a requested item was not found in the `dbm` files (NIS maps), then the request was forwarded to DNS. An IRIX server has an `nsswitch.conf` file just like the client, which gives a resolve order for each map. Now the line for hosts in the `/var/ns/domains/domainname/nsswitch.conf` file for the server and the `/var/ns/nsswitch.conf.sisserv` file for clients shows an entry `nisserv`, referring to the library for serving NIS. If you put `dns` after that, the name server will use DNS if a requested key is not found in the maps:

```
hosts:                  nisserv dns
```

If the `-i` flag was previously used, the entry should exist as described. Note that `ypserv` no longer exists.

NIS Software Quick Reference Guide

This section provides a quick reference to NIS daemons, files, and tools and suggests the man pages you should consult for complete information. The man pages at the end of this guide contain detailed information on the structure of the NIS system and NIS commands.

NIS Daemons

NIS daemons are as follows:

`rpc.passwd`	A server process that allows users with NIS accounts to change their NIS password and other NIS password-related fields. This process runs only on the master server.
`nsd`	The daemon acts as both server and client depending on how it is configured.

NIS Configuration Files

NIS configuration files are as follows:

`/var/ns/domains`
: The default location of NIS database files. For more information, see the `ypfiles` man page.

`/etc/config/rpc.passwd.options`
: Specifies an alternate NIS password filename. Default password file is `/etc/passwd`. Must be used in conjunction with the `/etc/config/ypmaster.options` PWFILE variable. For more information, see the `rpc.passwd` man page.

`/etc/config/nsd.options`
: Specifies default options to use with `nsd`. Options that can be included in this file are a secure mode and a cache timeout specifier. For more information, see the `nsd` man page.

NIS Tools

NIS tools are as follows:

`makemdbm`	A low-level tool for building an `mdbm` file that is a valid NIS map. You can use `makemdbm` to build or rebuild databases not built from `/var/yp/mdbm_parse`. You can also use `mdbm_dump` to disassemble a map so that you can see the key-value pairs that comprise it. In addition, you can modify the disassembled form with standard tools (such as editors, `awk`, `grep`, and `cat`). The disassembled form is in the form required for input back into `makemdbm`. See the `makedbm`(1M) man page for more information.

`ypcat`	Lists the contents of an NIS map. Use it when you do not care which server's map version you see. If you need to see a particular server's map, use the `rlogin` or `rsh` commands to gain access to that server, and use `makemdbm`. For details on `ypcat`, see the `ypcat`(1) man page.
`ypchpass`	Changes select NIS password fields. As the NIS user, you can change your full name, your home directory, and your default shell environment. Use `yppasswd` to change your NIS password. See the `ypchpass`(1) man page for details on the distinction between the two.
`ypinit`	Constructs many maps from files located in `/etc`, such as `/etc/hosts`, `/etc/passwd`, and others. The database initialization tool `ypinit` does all such construction automatically. Also, it constructs initial versions of maps required by the system but not built from files in `/etc`; an example is the map `ypservers`. Use this tool to set up the master NIS server and the slave NIS servers for the first time. Use `ypinit` to construct initial versions of maps rather than as an administrative tool for running systems. See the `ypinit`(1M) man page for details.
`ypmake`	Builds several commonly changed components of the NIS database from several ASCII files normally found in `/etc`: `bootparams`, `passwd`, `hosts`, `group`, `netgroup`, `networks`, `protocols`, `rpc`, and `services`, and the file `/etc/aliases`. The `/var/yp/ypmake.log` file is the log file for all `ypmake` activity. For more information, see the `ypmake`(1M) man page.
`ypmatch`	Prints the value for one or more specified keys in an NIS map. Again, you have no control over which server's version of the map you are seeing. See the `ypmatch`(1) man page for details on using `ypmatch`.
`yppasswd`	Allows NIS users to remotely change their NIS passwords. For details see the `yppasswd`(1M) man page.
`yppoll`	Asks `nsd` for the information it holds internally about a single map. See the `yppoll`(1M) man page for details on using `yppoll`.
`yppush`	Runs on the master NIS server. It requests each of the `nsd` processes within a domain to transfer a particular map, waits for a summary response from the transfer agent, and prints out the results for each server. For more information, see the `yppush`(1M) man page.
`ypset`	Tells an `nsd` process (the local one, by default) to get NIS services for a domain from a named NIS server. By default, `nsd` disallows the use of `ypset`. See the `ypset`(1M) man page for details on enabling `ypset`.

`ypwhich`	Tells you which NIS server a node is using at the moment for NIS services, or which NIS server is master of some named map. For more information, see the `ypwhich`(1M) man page.
`ypxfr`	Moves an NIS map from one NIS server to another, using NIS itself as the transport medium. You can run it interactively, or periodically from `crontab`. Also, `nsd` uses `ypxfr` as its transfer agent when it is asked to transfer a map. You can create the file `/var/yp/ypxfr.log` to log all `ypxfr` activity. See the `ypxfr`(1M) man page for details.

In addition to these NIS tools, the `rpcinfo` and `crontab` tools are also useful for administering NIS. For further information, please see the man page for each tool.

Chapter 3

Planning Your NIS Service

This chapter presents information to consider before you set up the NIS service on your network. It explains how to set up multiple NIS domains (if you decide they are needed) and identifies the files that should be up-to-date before NIS setup begins. It suggests how to name a domain and how to select master and slave servers. Finally, this chapter provides general recommendations to help you make planning decisions.

This chapter contains these main sections:

Establishing Multiple NIS Domains

Before you set up NIS, determine the number of domains you need. Establishing more than one domain is advisable if your network is very complex or requires a very large database. You might also consider using multiple domains if your network contains a large number of systems (say, in excess of 1000 systems).

If you decide to establish multiple domains and require interdomain communication, your planning involves additional network considerations. Those considerations are addressed in the remainder of this section.

Note: If you plan to establish a single domain or multiple isolated domains, you can skip ahead to "Verifying ASCII File Contents" to proceed with your planning.

Domain Boundaries

NIS is not hierarchical in nature; it cannot resolve issues that extend beyond domain boundaries. For example, suppose you set up two domains as shown in Figure 3-1: `shapes`, which includes system client1; and `colors`, which includes system client2. Without NIS, communication between client1 and client2 relies on entries in their local `/etc/hosts` file that provide a host name-to-address mapping.

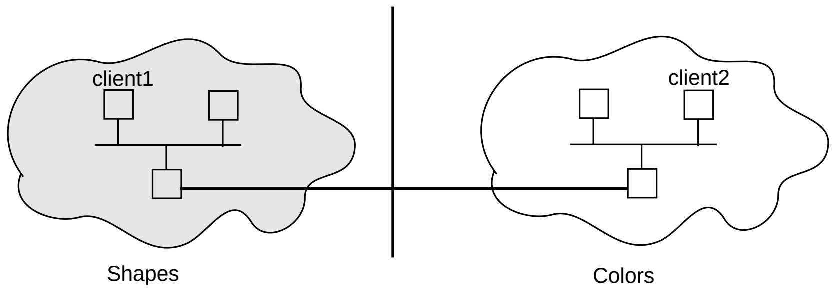

Figure 3-1 Boundary Problem With Multiple Domains

With NIS, host name and address information is in the `hosts` database on the NIS servers for a domain. However, this name and address information is limited to domain members. The `colors` database has no entry for client1 in the `shapes` domain, and the `shapes` database has no entry for client2 in the colors domain. Consequently, when client1 tries to contact client2, host name resolution fails and a connection cannot be established. Although there may be a physical connection between client1 and client2, there is no logical connection to support the communication process.

Bridging Domain Boundaries

When multiple NIS domains are used, you must form a logical bridge between domains to allow systems in different domains to communicate as shown in Figure 3-2. The logical bridge must contain or be able to access system information for all systems on a given network, regardless of domain. There are two ways to achieve this logical bridge: using the Domain Name System (DNS) or using a customized update procedure.

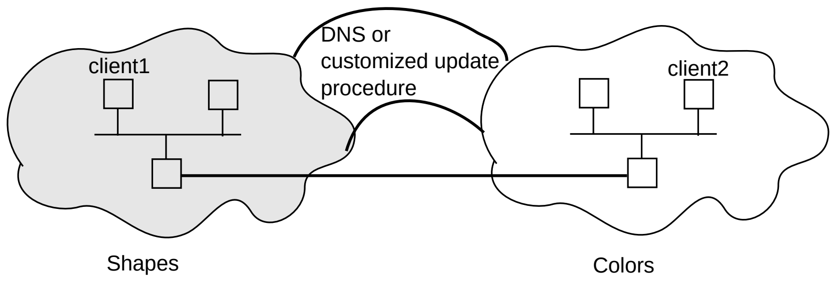

Figure 3-2 Boundary Solutions for Multiple Domains

Using the Domain Name System (DNS)

DNS, sometimes referred to as BIND (Berkeley Internet Name Daemon) or `named`, is a service that maps host names to IP addresses and vice-versa. DNS is concerned mainly with host name-address and address-host name resolution. It was developed to support very large scale environments and provides an accurate network depiction; it is hierarchical in nature. When correctly set up, DNS resolves host names and addresses throughout an entire network. For NIS to use DNS, DNS must be set up to know about all systems. The *IRIX Admin: Networking and Mail* book provides detailed information on setting up DNS.

By default, host name resolution is done by first checking NIS. If NIS is not running, then DNS is checked. If DNS is not running, then the local `/etc/hosts` file is checked. To redefine the host resolution order, change the `/etc/nsswitch.conf` file as described in the next section.

Using `/etc/nsswitch.conf`

The lookup order for resolving a system's identity can be configured in a variety of ways using `/etc/nsswitch.conf`. For example, a network application could resolve host name lookup by accessing files or databases in this order: NIS, DNS, and finally the local file. It can be configured to check only the first service running, or to check services until a match is found. Whatever order is specified, it becomes the default lookup order used by routines in the standard C library, such as `gethostbyname`(3N), for resolving host names.

If you want applications to resolve host names via the DNS database only, put this line in the `/etc/nsswitch.conf` file:

```
hosts: dns
```

If applications are to search only DNS and `/etc/hosts`, put this line in the `/etc/nsswitch.conf` file:

```
hosts: dns files
```

To specify that NIS should be checked first, then if no match is found check DNS, and if no match is found, check `/etc/hosts` and put this line in the `/etc/nsswitch.conf` file:

```
hosts: nis dns files
```

See the `nsswitch.conf`(1) man page for more detailed information.

Establishing a Customized Update Procedure

An alternative to using DNS is to establish a procedure for updating the hosts file on all master servers. For example, designate one system at your site to be the repository for new system addresses and limit administration of this system to a few select people. Set up a script and `crontab` entry on the designated system to copy its `/etc/hosts` file to the NIS master servers on each domain at regular intervals. When each NIS master server performs a `ypmake`, the host database is updated with the names and addresses for all systems on the network, regardless of the domain. This scheme distributes an updated list of all network systems to NIS servers, allowing clients in different domains to communicate successfully.

While DNS is mainly for host name resolution, NIS supports multiple database maps in addition to the `hosts` map. This method of setting up your own customized update procedure is also useful if you need the same information for other maps distributed between domains (for example, `/etc/aliases`).

Verifying ASCII File Contents

NIS databases are built on the NIS master server from a set of ASCII files the master server contains. A key preparation step is to ensure that the information contained in the ASCII files is correct and up-to-date.

Table 3-1 lists the maps that make up the NIS database, the input files that create these maps, and the purpose of each map in the NIS environment.

Table 3-1 Maps, ASCII Files, and Descriptions

Map Name	ASCII File	Description
`bootparams`	`/etc/bootparams`	Contains pathnames of files diskless clients need during booting: `root`, `swap`, `share`, possibly others.
`capability.byname`	`/etc/capability`	Contains information about the capability each user may have when logging onto the system.
`clearance.byname`	`/etc/clearance`	Contains mandatory access control labels to allow user access to the system.
`ethers.byaddr`	`/etc/ethers`	Contains host names and Ethernet addresses. The Ethernet address is the key in the map.
`ethers.byname`	`/etc/ethers`	Same as `ethers.byaddr`, except the key is the host name instead of Ethernet address.
`group.bygid`	`/etc/group`	Contains group security information with the group ID as key.
`group.byname`	`/etc/group`	Contains group security information with the group name as key.
`group.bymember`	`/etc/group`	Contains all groups of which a login is a member.

Table 3-1 (continued) Maps, ASCII Files, and Descriptions

Map Name	ASCII File	Description
`hosts.byaddr`	`/etc/hosts`	Contains host names and IP addresses, with the IP address as key.
`hosts.byname`	`/etc/hosts`	Contains host names and IP addresses, with the host name as key.
`mac.byname`	`/etc/mac`	Contains mandatory access control information, with the name as key.
`mac.byvalue`	`/etc/mac`	Same as `mac.byname,` except that the key is numeric value.
`mail.aliases`	`/etc/aliases`	Contains aliases and mail addresses, with aliases as key.
`mail.byaddr`	`/etc/aliases`	Contains mail addresses and aliases, with the mail address as key.
`netgroup.byhost`	`/etc/netgroup`	Contains group names, user names, and host names, with the host name as key.
`netgroup.byuser`	`/etc/netgroup`	Same as `netgroup.byhost`, except that the key is the user name.
`netgroup`	`/etc/netgroup`	Same as `netgroup.byhost`, except that the key is the group name.
`netid.byname`	`/etc/group`, `/etc/hosts`, `/etc/netid`	Contains user, group, and host information, with the user name as the key.
`networks.byaddr`	`/etc/networks`	Contains names of networks known to your system and their IP addresses, with the address as the key.
`networks.byname`	`/etc/networks`	Same as `networks.byaddr`, except the key is the name of network.
`passwd.byname`	`/etc/passwd`	Contains password information with the user name as the key.
`passwd.byuid`	`/etc/passwd`	Same as `passwd.byname`, except that the key is the user ID.

Table 3-1 (continued) Maps, ASCII Files, and Descriptions

Map Name	ASCII File	Description
`protocols.byname`	`/etc/protocols`	Contains network protocols known to your network, with the protocol name as the key.
`protocols.bynumber`	`/etc/protocols`	Same as `protocols.byname`, except that the key is the protocol number.
`rpc.byname`	`/etc/rpc`	Contains the program number and name of RPCs known to your system. The key is the RPC program name.
`rpc.bynumber`	`/etc/rpc`	Same as `rpc.byname` except that the key is the RPC program number.
`services`	`/etc/services`	Same as `services.byname`, but key is the service name and protocol.
`services.byname`	`/etc/services`	Lists Internet services known to your network. Key is the port or protocol.
`shadow`	`/etc/shadow`	An access-restricted shadow password file.
`ypservers`	`/var/yp/ypservers`	Lists NIS servers known to your network. Initially created by `ypinit` when the master server was built.

Selecting a Domain Name

The name you choose for your NIS domain is at your discretion; however, it should reflect some characteristics of the network it is serving, such as its location, function, or types of systems it contains. You can use a simple domain name, such as marketing; or, if you are a member of the Internet and you choose to do so, you can use your Internet domain name (such as finance.company.com) as your NIS domain name.

The `domainname` command sets a domain name on an NIS system. The NIS domain name is assigned at system startup. Enter it in the domain file, `/var/yp/ypdomain`. Be aware that domain names are case sensitive: marketing and Marketing are different domains. See Chapter 4, “Setting Up and Testing NIS,” for complete instructions on setting domain names, and the `domainname` (1) man page for details of the `domainname` command.

Selecting the NIS Master Server

Determine the system to be the NIS master server for the domain; there is only one NIS master server per domain. The NIS master server houses the original NIS database maps for the domain and is the only server on which changes are made to the NIS database. For this reason, the master server should be a very reliable and stable system. It must be accessible via the network to both NIS clients and NIS slave servers. The master server need not be a dedicated system; it can be responsible for other functions as well.

This is also a good time to determine the name of the NIS password file to be used. By default, NIS derives the database file from the ASCII version of `/etc/passwd`. This can be a security hole as all system password files require a root account.

To ensure security, create a separate NIS password file that contains no root or superuser-equivalent accounts (no UID=0). A good generic NIS password filename is `/etc/passwd.nis`. If you plan to use a password file other than the default `/etc/passwd`, you must tell NIS about the new filename. To do so, you must create the `/etc/config/rpc.passwd.options` and `/etc/config/ypmaster.options` files to support `/etc/passwd.nis`, the NIS password file. The contents of these files should look like the following examples:

```
# cat /etc/config/rpc.passwd.options
PWFILE=/etc/passwd.nis
```

```
# cat /etc/config/ypmaster.options
PWFILE=/etc/passwd.nis
```

Selecting the NIS Slave Servers

Slave servers contain copies of the NIS database. The number of NIS slave servers you assign per domain depends upon the size of the domain and the number of networks over which your domain extends. NIS slave servers must be accessible to both NIS clients and the NIS master server by means of the network. NIS slave servers should be reliable systems; the degree of reliability of these systems depends on the availability of backup slave servers.

By default, NIS clients broadcast an NIS bind request when they boot. Since broadcast requests cannot go through gateways, you `must` have at least one NIS slave server on any network where there are NIS clients. For reliability, there should be more than one NIS slave server on any network where there are NIS clients.

Broadcasting bind requests is the default setting, but clients can specify the server they wish to bind to at boot time. For instance, say you have a domain that encompasses many subnets, one of which contains only one client. To avoid making that client a server, you can specify a list of servers the client can bind to at boot time.

Specify an NIS Server at Client Startup

To specify an NIS server at client startup, modify the `/etc/config/NSD.options` file or the `/etc/nsswitch.conf` file. These files contain the resolve order and identify the address of any NIS server. Add the following line to the `/etc/config/NSD.options` file:

`nis_servers=`*nnn.nn.nnn*

As an alternative, add the following line to the `/etc/nsswitch.conf` file:

`ypservers:` *xx.xx.xx.xx. yy.yy.yy.yy*

where *nnn.nn.nnn* and *xx.xx.xx.xx. yy.yy.yy.yy* are the IP addresses or host names of specified servers, but IP addresses are preferred.

General Recommendations

Below are some general recommendations for setting up NIS. Because these are only general recommendations, you may need to tailor them to fit your specific site requirements.

1. During the planning phase, sketch the NIS implementation for your network. Identify the master server, slave servers, and client systems. If you have multiple domains, include them in your drawing.
2. If your domain spreads over several networks, ensure that there are at least two slave servers per network in case of system or network failures.
3. Create an alternate password file to be used by NIS only, that does not have any root user IDs. For example, specify `/etc/passwd.nis` as the NIS password file.

4. To simplify administration and troubleshooting, maintain only one master server for all maps within a single domain.
5. Plan to do all database creation and modification on the master server.

Setting Up and Testing NIS

Setting up NIS consists of three general procedures: setting up the master server, setting up the slave servers, and setting up the clients. The instructions in this chapter explain how to set up NIS by guiding you through procedures on sample NIS systems in a sample NIS domain.

This chapter contains these sections:

This sample setup in this chapter is representative of what must be done to set up NIS on any network, regardless of its specific characteristics. It assumes that NSD is running on all machines involved. When you use these instructions, substitute your own values for the ones shown in our examples. In our examples, NIS entities have these names:

- The domain name is `shapes`.
- The master server name is `circles`.
- Slave server names are `slave1` and `slave2`.

Note: Host names used in the NIS environment must be the official host names, not nicknames. The official host name is the name returned by the `hostname` command. See the `hostname`(1) man page for usage details.

Setting Up the NIS Master Server

There are four parts to the procedure for setting up the NIS master server.

1. Setting the master server's domain name.
2. Building the master maps.
3. Starting NIS on the master server.
4. Testing the NIS master server.

Setting the Master Server's Domain Name

Set the system's domain name based on your site's configuration. Recall that the domain name for this example is `shapes`. As you do this step, replace `shapes` with the domain name you chose for your site.

If your site configuration consists of only NIS domains and/or the NIS domain names are not the same as the Internet domain names, do the following:

1. Set the domain name as follows:

   ```
   circles# echo shapes > /var/yp/ypdomain
   circles# domainname shapes
   ```

2. Verify the domain name setting with the `domainname` command, as follows:

   ```
   circles# domainname
   shapes
   ```

   If the domain name is correctly set, the `domainname` command returns the domain name you specified in step1 of this procedure. If your output is not correct, reissue the commands in step1.

If your site configuration consists of NIS domains and Internet domains with the same names, do the following (the example assumes that the NIS and Internet domains are both named `widgets.com`):

1. Set the official host name for the master server (the host name for our example is `circles`):

   ```
   circles# echo circles.widgets.com > /etc/sys_id
   ```

2. Reboot the system:

   ```
   circles# /etc/reboot
   ```

The `/var/yp/ypdomain` file is not required if the domain names for the NIS and Internet domains are the same. Also, the domain name must be part of the official host name set in the `/etc/sys_id` file. If a `/var/yp/ypdomain` file exists, the domain name set in the `/var/yp/ypdomain` file overrides the domain name specified in the `/etc/sys_id` file.

Building the Master Maps

The command `ypinit` builds NIS maps using the text files with `/var/yp/mdbm_parse`. (See Chapter 2, “Preparing to Manage NIS,” for a list of the default files that are converted to maps in this step. See also the `ypinit` (1M) man page for details of the `ypinit` command.)

1. Start building the master NIS maps using the `ypinit` command on host `circles`:

   ```
   circles# cd /var/yp
   circles# ./ypinit -m
   We now need to contruct a list of hosts which run NIS servers.
   Enter the names or addresses of these hosts one at a time,
   excluding this host, then simply hit <Enter> to end the list.
   Name (<Enter> to exit): squares
   Name (<Enter> to exit): triangles
   Name (<Enter> to exit):
   Parsing configuration files into databases.
   ```

   The `–m` flag denotes that `circles` is an NIS master server.

   If there is any doubt about the integrity of the database maps, always go and rebuild the maps from scratch.

If you are creating a new master server for an already existing domain with functioning slave servers, you must run `yppush` to propagate the new maps to the slave servers (see Chapter 5, "Maintaining NIS," for details on changing a master server).

Starting NIS on the Master Server

The NIS service is available to clients as soon as you start it on the master server. You can start NIS by any one of these methods:

- Reboot the NIS master server.
- Stop and restart the network using the `/etc/init.d/network` script.
- Start the daemons manually.

Give the following command to start the daemons manually:

```
circles# /usr/etc/rpc.passwd /etc/passwd.nis -m passwd
```

Note that the `rpc.passwd` process initiated in this command sequence assumes the existence of a specific NIS password file called `/etc/passwd.nis`. See "Selecting the NIS Master Server" in Chapter 3 for details on setting up a nonstandard NIS password file. See also the `rpc.passwd`(1M) man page for more information on the command.

NIS master machines must be configured with the `chkconfig` command set on for `ypserv` and `ypmaster`. NIS slave servers must be configured with the `chkconfig` command set on for `ypserv`. Finally, run `nsadmin` to restart the daemon.

To set the flags on and restart the `nsd` daemon, give these commands:

```
circles# /etc/chkconfig ypmaster on
circles# /etc/chkconfig ypserv on
circles# nsadmin restart
```

Testing the NIS Master Server

Finally, to ensure that NIS services are functioning properly on the NIS master server, enter the `ypwhich` command. Since the NIS master server is also a client, it should return with the name of the server to which it is bound. Remember, an NIS master server is bound to itself, and it returns its own name.

Example:

```
circles# ypwhich
circles.widgets.com
```

The response `localhost` indicates that `nsd` is correctly bound to the NIS server on the local system. Instead of `localhost`, it may return its name as reported by `hostname`. For further details, refer to the `ypwhich`(1M), `nsd`(1M), and `hostname`(1) man pages.

Setting Up NIS Slave Servers

Use the following procedure to set up the NIS slave server. If you have more than one NIS slave server, repeat each part of the procedure for each slave server.

1. Set the slave server's domain name (see "Setting the Slave Server's Domain Name" on page 37).
2. Enter the following command:
   `chkconfig yp on`
3. Enter the following command:
   `chkconfig ypserv on`
4. Bind to an NIS server (see "Binding to Another NIS Server" on page 37).
5. Build the duplicate maps (see "Building the Duplicate Maps" on page 38).
6. Start NIS on the slave server (see "Starting NIS on the Slave Server" on page 39).
7. Test the NIS slave server (see "Testing the NIS Slave Server" on page 39).

Setting the Slave Server's Domain Name

Follow the instructions in "Setting the Master Server's Domain Name" in this chapter to complete this step.

Binding to Another NIS Server

To propagate NIS database maps from the NIS master server to a NIS slave server, the slave server must be bound to a valid NIS server in its domain.

In the following examples, since `circles` is a valid NIS server, this slave server binds to `circles`. Binding need not be to a master server, however.

1. If the slave is not on the same network as `circles`, verify that the master server has an entry in the `/etc/hosts` file on the slave server:

   ```
   slave1# grep circles /etc/hosts
   192.0.2.4 circles.rad.sgx.com circles
   ```

2. Whether or not the slave is on the same network as `circles`, you should halt any existing `nsd` command, and start the binding process with the specified attribute of `nsd`, as follows:

   ```
   slave1# killall -TERM nsd
   slave1# /usr/etc/nsd -a nis_security=local
   ```

3. Give the `ypset` command to point NSD at the server `circles`.

   ```
   slave1# ypset circles
   ```

4. Verify that the server is bound by giving the `ypwhich` command:

   ```
   slave1# ypwhich
   circles
   ```

   The output of `ypwhich` returns the name of the NIS server to which this server is currently bound. The example shows that this slave server is successfully bound to `circles`.

Building the Duplicate Maps

The command `ypinit` builds the duplicate database maps by transferring a copy of the original maps from the NIS master server.

1. Determine which system is the master server by using the `ypwhich` command:

   ```
   slave1# ypwhich -m
   ```

2. Start building NIS slave server maps with the `ypinit` command. In this example the `–s` flag specifies that this system is to be an NIS slave server, and `circles` is the master server:

   ```
   slave1# cd /var/yp
   slave1# ./ypinit -s circles
   ```

   Each line of the `ypinit` output contains the name of a map and the name of the master server where the map was created.

```
Transferring map networks.byname from server circles.shapes.
```

```
Transferring map services.byname from server circles.shapes.
Transferring map passwd.byname from server circles.shapes.
Transferring map hosts.byaddr from server circles.shapes.
Transferring map ktools from server circles.shapes.
Transferring map ypservers from server circles.shapes.
Transferring map hosts.byname from server circles.shapes.
Transferring map networks.byaddr from server circles.shapes.
Transferring map protocols.byname from server circles.shapes.
Transferring map group.byname from server circles.shapes.
Transferring map netgroup from server circles.shapes.
Transferring map mail.aliases from server circles.shapes.
Transferring map ethers.byname from server circles.shapes.
Transferring map protocols.bynumber from server circles.shapes.
Transferring map netgroup.byhost from server circles.shapes.
Transferring map group.bygid from server circles.shapes.
Transferring map passwd.byuid from server circles.shapes.
Transferring map ethers.byaddr from server circles.shapes.
Transferring map netgroup.byuser from server circles.shapes.
```

Starting NIS on the Slave Server

The NIS service is available to clients as soon as you start it on this slave server. You can start NIS by any one of these methods:

- Reboot the NIS slave server.
- Stop and restart the network by using the `/etc/init.d/network` script.
- Start the daemons manually.

Give the following command to start the daemons manually:

```
slave1# ./ypinit -s circles
```

The NIS maps are now available from the server `slave1` in the domain `shapes`.

Testing the NIS Slave Server

Finally, to ensure that NIS services are functioning properly on the NIS slave server, enter the `ypwhich` command. Since the NIS slave server is also a client, it should return with the name of the server to which it is bound. This server can be bound to either itself or to the NIS master server you set up in the previous section: either result is acceptable. Example:

```
slave1# ypwhich
localhost
```

The response, `localhost`, indicates that `nsd` is correctly bound to the NIS server on the local system. The response could have also been the name of another NIS server within the same domain on the same local area network.

Setting Up NIS Clients

Use the following procedure for setting up the NIS client. Repeat these steps for each NIS client you need to set up. Each step is described in the sections that follow.

1. Set the domain.
2. Configure NIS on the client.
3. Start NIS on the client.
4. Test the NIS client.

Setting the Domain Name

Follow the instructions in "Setting the Master Server's Domain Name" on page 34 to complete this step.

Configuring NIS on the Client

If the NIS service is to start automatically when this client (`triangles`) is booted, the NIS environment must be configured using the `chkconfig` command. The `yp` flag allows this system to access NIS database information from an NIS server. To set the flag on, enter this command:

```
triangles# /etc/chkconfig yp on
```

Edit the `/etc/nsswitch.conf` file, using any standard editor, by adding NIS to the hosts line:

```
hosts:                  files nis dns
```

Starting NIS on the Client

The NIS service operates on this client as soon as you start it. You can start NIS by any one of these methods: rebooting this client, stopping and restarting the network with the `/etc/init.d/network` script, or starting the NSD daemon manually by performing one of the following commands:

```
triangles# nsadmin restart
```

```
triangles# ypinit -c
```

Testing the NIS Client

To ensure that the NIS services are functioning properly on the NIS client, give the `ypwhich` command. It should return with the name of the server to which it is bound, for example:

```
# ypwhich
squares
```

The client can be bound to any NIS server on the same network as the request is broadcast. This client is currently bound to the server `squares`, which means that `squares` must be on the same network as the client. If more than one NIS server is on the same network, the client binds to the server that responds first.

Chapter 5

Maintaining NIS

This chapter describes, and provides procedures for, NIS maintenance as indicated in the titles of the following sections:

Adding a New User to a System

To add a new user to a system that is an NIS client, perform these steps:

1. On the NIS master server, add an entry for the new user to the NIS password file (`/etc/passwd` by default). (For more information, see the `passwd`(4) man page, the *Personal System Administration Guide*, or the book *IRIX Admin: System Configuration and Operation*.)
2. On the NIS master server, update the NIS `passwd` map on that system by entering:

   ```
   # cd /var/yp
   # ./ypmake passwd
   ```

3. If this user is to be a member of any netgroups, modify `/etc/netgroups` on the NIS master server (see "Using Netgroups" in this chapter).
4. On the new user's system, modify `/etc/passwd` in one of these ways:
   - Add the same entry as you added in step 1 in this section. Duplicating the entry enables the user to log in when the network is down.
   - Add this entry:

     +*userid*

     Where *userid* is the login name of the new user. When this type of entry is used, all `/etc/passwd` information for this user is supplied by NIS.
   - Use the Users tool of System Manager to add the new user. Choose NIS rather than Local for each item. All `/etc/passwd` information for this user is supplied by NIS.
   - Add this entry:

     +

     When this type of entry is used, all `/etc/passwd` information for all users is supplied by NIS. Every user in the NIS password database can log in to this system, assuming that the home directory exists. For alternative NIS entries, see the `passwd`(4) man page.

5. Make a home directory for the new user on the user's system:

   ```
   # cd parentdir
   # mkdir userid
   # chown uid userid
   # chgrp group userid
   ```

   The variables are:

parentdir	The parent directory of the home directory you are creating.
userid	The login name of the user (the first field in the password entry).
uid	The unique user ID number for this user (the third field in the password entry). *userid* can be used instead of *uid* if the local `/etc/passwd` entry duplicates the NIS password entry or if NIS maps have been propagated to this system (it takes about 15 minutes after step 2 for updates to propagate).
group	The group number for this user (the fourth field in the password entry).

6. Finish adding the new user by setting up the user's login environment (create `.login` and `.cshrc` files, for example), adding him or her to groups in `/etc/group`, and doing other setup tasks that are usually done at your site.

7. Have the user add a password to his or her account using `yppasswd`:

   ```
   % yppasswd
   ```

   `yppasswd` prompts the user to enter the new password twice.

8. If you added a complete password entry to `/etc/passwd` on the user's system (the first option in step 4), have the user add his or her password to the local `/etc/passwd` file, using `passwd`:

   ```
   % passwd
   ```

   `passwd` prompts for the password twice.

Changing NIS Passwords

In general, all NIS accounts should be password protected. This reduces the risk of malicious or accidental data corruption. When you change your password with the `passwd` command, you change the entry explicitly given in your own local `/etc/passwd` file. To change your NIS password, use the `yppasswd` command.

1. To change the NIS password for the user `tim`, enter

   ```
   % yppasswd tim
   Changing NIS password for tim on master_name
   Old password: <response not echoed>
   ```

2. Enter your old NIS password (if the account is not password protected, press **`Enter`**.)

   ```
   New password: <response not echoed>
   ```

3. Enter your new NIS password:

   ```
   Retype new password: <response not echoed>
   ```

4. Reenter the new password:

   ```
   NIS passwd changed on master_name
   ```

Your NIS password change has been logged on the master server and will be updated soon. Note that it takes a little while for the change to propagate throughout the domain. You can update the information more quickly by using the `ypmake` command on the NIS master to push the data back to all slave servers after updating its own databases. For more information, see the `crontab` entries on the NIS master server and see "Propagating an NIS Map" on page 57.

If your local password is not given explicitly but rather is pulled in from NIS with a plus (+) entry, then the `passwd` command prints this error message:

```
Not in passwd file
```

In this case, you must use `yppasswd` to change your password.

To enable the `yppasswd` service, the system administrator must start up the daemon `rpc.passwd` server on the system serving as the master server for the NIS password file in your domain.

Using Netgroups

The `/etc/netgroup` file on the NIS master server contains a list of network-wide groups of systems and users. These groups are used for administrative purposes. For example, to define a set of users that should be given access to a specific system, you can create a netgroup for those users.

The daemons `login`, `mountd`, `rlogind`, and `rshd` use netgroups for permission checking. `login` consults them for user classifications if it encounters netgroup names in `/etc/passwd. mountd` consults them for system classifications if it encounters netgroup names in `/etc/exports. rlogind` and `rshd` consult the netgroup map for both system and user classifications if they encounter netgroup names in `hosts.equiv` or `.rhosts`.

The NIS master server uses `/etc/netgroup` to generate three NIS maps: `netgroup`, `netgroup.byuser`, and `netgroup.byhost`. The NIS map `netgroup` contains the basic information in `/etc/netgroup`. The two other NIS maps contain a more specific form of the information to speed the lookup of netgroups given the system or user.

Below is a sample `/etc/netgroup` file. (See the `netgroup`(4) man page for a description of file format and definition of lines and fields.)

```
# Engineering: Everyone but eric has a machine;
# he has no machine.
# The machine 'testing' is used by all of hardware.
#
engineering hardware software
hardware (mercury,alan,sgi) (venus,beth,sgi) (testing,-,sgi)
software (earth,chris,sgi) (mars,deborah,sgi) (-,eric,sgi)
#
# Marketing: Time-sharing on jupiter
#
marketing (jupiter,fran,sgi) (jupiter,greg,sgi)
#
# Others
#
allusers (-,,sgi)
allhosts (,-,sgi)
```

Note: IRIX (as well as most other UNIX operating systems) supports only 1,024 bytes per line/entry in the `netgroup` file.

Table 5-1 shows the users in each group.

Table 5-1 Sample User Groups

Group	**Users**
`hardware`	`alan`, `beth`
`software`	`chris`, `deborah`, `eric`
`engineering`	`alan`, `beth`, `chris`, `deborah`, `eric`
`marketing`	`fran`, `greg`
`allusers`	(every user in the NIS map `passwd`)
`allhosts`	(no users)

Table 5-2 shows how the systems are classified.

Table 5-2 Sample Host Groups

Group	**Hosts**
`hardware`	mercury, `venus`, `testing`
`software`	`earth`, `mars`
`engineering`	`mercury`, `venus`, `earth`, `mars`, `testing`
`marketing`	`jupiter`
`allusers`	(no systems)
`allhosts`	(all systems in the NIS map `hosts`)

For more details, see these man pages: `yppasswd`(1), `hosts.equiv`(4), `export`(4), `passwd`(4), `group`(4), `netgroup`(4), and `rpc.passwd`(1M).

Creating a Nonstandard NIS Map Manually: Pre-IRIX 6.5

You can use `ypinit` and `/var/yp/local.make.script` (see the `ypmake`(1M) man page) to do almost everything necessary to create and modify a map, unless you add nonstandard maps to the database or change the set of NIS slave servers after the system is already running. Whether you use `/var/yp/local.make.script` or some other procedure, the goal is the same: a new well-formed `mdbm` file in the domain directory on the NIS master server.

You can create new maps in two ways: using an existing ASCII file as input or using standard keyboard input. The next two sections demonstrate how to create a simple, nonstandard NIS map called `yourmap` using each method. `yourmap` consists of the keys `al`, `bl`, `cl`, and so on (`l` for left); and one set of values, `ar`, `br`, `cr`, and so on (`r` for right).

ASCII File Input

Assume the ASCII file `/etc/yourmap` has been created with an editor or shell script, and that the map from `/etc/yourmap` is part of the database for the `shapes` domain. To create the NIS map for this file, enter these commands:

```
# cd /var/yp
# makemdbm /etc/yourmap /var/ns/domains/shapes/yourmap.m
```

This command sequence creates a map called `yourmap` in the directory `/var/ns/domains/shapes`.

Standard Keyboard Input

When no original ASCII file exists, you can create the NIS map described in the previous example from the keyboard with these commands:

```
# cd /var/ns/domains
# makemdbm - /var/ns/domains/shapes/yourmap.m
```

Enter these lines:

```
al ar
bl br
cl cr
Ctrl-D
```

The `makemdbm` switch is used to indicate that input is coming directly from the keyboard. The result of your entries is the same as the previous example: a map called `yourmap` in the directory `/var/ns/domains/shapes`.

Adding the New NIS Map

Add an entry for the new NIS map in the `/var/ns/domains/<`*domainname*`>/nsswitch.conf` file with these commands:

```
# cd /var/ns/domains
# cat >> shapes/nsswitch.conf
yourmap: nisserv
Ctrl-D
```

The `nsswitch.conf` file now contains the name of the map followed by the protocol to be used. For more information, see the `nsswitch.conf`(4) man page.

Finally, you must send the `nsd` daemon a SIGHUP signal for the change to take effect. Enter this command:

```
# killall -HUP nsd
```

Creating a Nonstandard NIS Map Manually: IRIX 6.5 and Later

The IRIX operating system is shipped without default NIS server settings for some common NIS maps, such as `auto.master` and `auto.home`. This remains true for IRIX 6.5 and subsequent releases. However, because of the advent of Unified Name Services, NSD, and the demise of the `ypserv` process, a new procedure is required to serve these maps under IRIX 6.5.

The following steps describe how to serve these maps under IRIX 6.5 and later. Use this procedure if your system has been upgraded to IRIX 6.5 or a subsequent release and was formerly an NIS master server:

1. Rename your current (that is, earlier than IRIX 6.5) `/var/yp/local.make` file.

   Caution: If you do not rename this file, `make.script` will try to run the old `local.make.script` file (see step 6) and will fail.

When you create a new script to make and push the maps, you can edit your pre-IRIX 6.5 `/var/yp/local.make.script` file to conform to the changes that are made to the `local.make.script` file (see step 6).

2. Make sure that `check configuration` is `on` for `yp`, `ypmaster`, and `ypserv`, and that NIS services are running on the system.

3. Execute the `ypwhich -m` command. This command should produce the following output:

```
bootparams master
capability.byname master
ethers.byname master
group.byname master
group.bygid master
group.bymember master
hosts.byaddr master
hosts.byname master
mail.aliases master
mail.byaddr master
netgroup master
netgroup.byuser master
netgroup.byhost master
netid.byname master
networks.myaddr master
networks.byname master
passwd.byname master
passwd.byuid master
protocols.bynumber master
protocols.byname master
rpc.bynumber master
rpc.byname master
services master
services.byname master
ypservers master
```

4. Make sure that the NIS server is fully functional and that the default maps are updated and pushed properly.

5. Add the following line to the `/var/ns/domains/<yourdomain>/nsswitch.conf` and `/var/ns/nsswitch.conf.nisserv` files to add a new map, called `auto.foo`. This file is based on a file named `/etc/auto.foo`:

```
auto.foo:          nisserv
```

Note: You must edit both files. This is because `/var/ns/nsswitch.conf.nisserv` is a prototype file copied to the `/var/ns/domain/`*yourdomain*`/nsswitch.conf` file when the `ypinit -m` command executes. This enables a single host to serve more than one NIS domain.

6. Make a copy of the old `local.make.script` file that you renamed in step 1. Then, edit this file to conform to the syntax and to account for the following major differences between the NIS server for IRIX 6.5 and previous versions:

   - The IRIX 6.5 NIS server uses `mdbm` instead of `dbm`.
   - The location and names of most files, including the database file, are different.
   - There may be differences in the behavior of the `make` command.
   - The `ypmake` command is an `sh` script that runs `mdbm_parse`, `local.mdbm_parse`, and `make make.script`, in that order.

     The `mdbm_parse` program is a Perl program that is used to maintain the default maps. A local `mdbm_parse` program can be used by any type of program, although Perl is preferred. `make.script` is still provided, but it is a null script that has no effect on the default `yp` maps. It is only used in the event that it contains an `sinclude` call to call a legacy `local.make.script`.

     The following script is an example of a `local.make.script` file for the `auto.foo` map modified to work under IRIX 6.5.

     Note: Do not substitute spaces for tabs in this script.

```
AUTO_MASTER=auto_master

#Makedbm distributed with IRIX that builds maps for
#audofsd/automount correctly

O_MAKEMDBM =     /usr/sbin/makemdbm

localall: all $(AUTO_MASTER)

#master map for autofsd:
$(AUTO_MASTER): $(DIR)/$(AUTO_MASTER)
     date "+$(UPDATEFMT) $(AUTO_MASTER) $(DATEFMT)"
     $(O_MAKEMDBM) $(DIR)/$(AUTO_MASTER) $(YPDBDIR)/$(AUTO_MASTER).m
```

```
if [ ! $(NOPUSH) ]; then \
     $(YPPUSH) $(AUTO_MASTER); \
     date "+$(PUSHFMT) $(AUTO_MASTER) $(DATEFMT)";\
fi
```

7. Execute the `ypmake` command.
8. Once the `local.make.script` creates the `mdbm` file, and the local script is working, execute `nsadmin restart` to restart NSD with the new database.
9. Execute the `ypwhich -m` command to ensure that the new map is being served by the NIS server.
10. Edit the `/etc/nsswitch.conf` file to add the map to all IRIX 6.5 clients, including the NFS servers.

Additional Tips for Creating Nonstandard Maps Under IRIX 6.5

Observe the following when you create nonstandard maps under IRIX 6.5 and later:

- Each time you execute the `ypinit -m` command, you must repeat the previous procedure as shown. This is because the `ypinit -m` command removes the `/var/ns/domain/`*yourdomain* directory.
- For NIS clients running versions of IRIX prior to 6.5.1, you should assume that host and domain names are case-sensitive.
- When a problem occurs, the following commands can be helpful: `nsadmin restart`, `nsadmin cat`, `killall -USR2 nsd`, and so forth. See also the `nsadmin`(1M) man page for related information.
- The `makemdbm` command treats the first word or symbol of a line, up to the first white space, as the key. This includes comments. Because comments in the input file are all likely to begin with a # (pound sign), `makemdbm` puts the first comment line that it encounters into the database with a key of "#", and ignores subsequent comments.

  To ensure that NIS clients do not misinterpret comments as valid records, you should pass the input file through a regular expression that removes the comment lines.

Modifying NIS Maps After NIS Installation

To change any NIS map, you must change the databases on the master server for the domain. The method you use to modify the map depends on whether you are changing a standard or nonstandard map.

Modifying a Standard NIS Map

A standard NIS map is any map that has an ASCII file and is included in the `/var/yp/mdbm_parse` file. The procedure for modifying a standard NIS map consists of editing the ASCII file for the map and updating the map with `ypmake` on the master server. For example, to modify the password database map, edit the ASCII file for the map and run `ypmake` on the master server. To add the user `tom` to the password database, perform these steps:

1. Edit the ASCII file:

   ```
   # vi /etc/passwd.nis
   ```

2. Add this line to the password file:

   ```
   tom::2349:20:Tom Cat:/usr/people/tom:/bin/csh
   ```

3. Update the `password` map:

   ```
   # /var/yp/ypmake passwd
   ```

By default, the `ypmake` program updates the map on the master server based on information contained in the make script (`/var/yp/mdbm_parse` and `/var/yp/local.make.script`). It also propagates the updated map to all slave servers listed in the `ypservers` database map.

Modifying a Nonstandard NIS Map

Nonstandard or custom NIS maps are databases that are specific to the application of a particular vendor site but are not part of the NFS release. You can manually modify custom maps. You can also manually change maps that are not expected to change and maps for which no ASCII form exists.

The general procedure is to use `makemdbm` with a switch to disassemble the map. The disassembled map is in a form you can modify with standard tools such as `awk`, `sed`, or `vi`. You then build a new map from the changed version using `makemdbm`.

Use this procedure to modify a nonstandard NIS map:

1. Disassemble the map, as shown in this sample command:

   ```
   # cd /var/yp
   # makemdbm -u /var/ns/domains/shapes/mymap.m > /var/tmp/mymap.txt
   ```

2. Edit the text file (`/var/tmp/mymap.txt`, in this example) with any text editor.
3. Build the new map, as shown in this sample command:

   ```
   # makemdbm /var/tmp/mymap.txt /var/ns/domains/shapes/mymap.m
   ```

4. Remove the temporary ASCII file, as shown in this sample:

   ```
   # rm /var/tmp/mymap.txt
   ```

This procedure modifies and updates custom maps but does not propagate the map to slave servers.

Preparing to Propagate Nonstandard Maps

Preparing to propagate a nonstandard NIS map consists of setting up its `mdbm` files in the domain directory on each NIS server (the transfer mechanism is described in the next section). The files must be set up correctly on the master and each slave server in the domain.

On the NIS master server, create a new file called `/var/yp/local.make.script` so you can conveniently rebuild the map. This example shows a copy of `/var/yp/local.make.script` to create and push the maps `auto.test`, `auto.direct`, `auto_master`, and `auto.home` from the files `/etc/auto.test`, `/etc/auto.direct`, `/etc/auto_master` (used by the `autofsd` daemon), and `/etc/auto.home`.

```
AUTO_MASTER=auto_master
DIRECT=auto.direct
INDIRECT=auto.test
HOME=auto.home

O_MAKEMDBM =    /usr/sbin/makemdbm

localall: all $(AUTO_MASTER) $(DIRECT) $(INDIRECT) $(HOME)


# master map for autofsd:
```

```
$(AUTO_MASTER): $(DIR)/$(AUTO_MASTER)
        date “+$(UPDATEFMT) $(AUTO_MASTER) $(DATEFMT)”
        $(O_MAKEMDBM) $(DIR)/$(AUTO_MASTER) $(YPDBDIR)/$(AUTO_MASTER).m
        if [ ! $(NOPUSH) ]; then \
                $(YPPUSH)  -v $(AUTO_MASTER); \
                date “+$(PUSHFMT) $(AUTO_MASTER) $(DATEFMT)”;\
        fi
# DIRECT map:
$(DIRECT): $(DIR)/$(DIRECT)
        date “+$(UPDATEFMT) $(DIRECT) $(DATEFMT)”
        $(O_MAKEMDBM) $(DIR)/$(DIRECT) $(YPDBDIR)/$(DIRECT).m
        if [ ! $(NOPUSH) ]; then \
                $(YPPUSH) -v $(DIRECT); \
                date “+$(PUSHFMT) $(DIRECT) $(DATEFMT)”;\
        fi

# INDIRECT map:
$(INDIRECT):    $(DIR)/$(INDIRECT)
        date “+$(UPDATEFMT) $(INDIRECT) $(DATEFMT)”
        $(O_MAKEMDBM) $(DIR)/$(INDIRECT) $(YPDBDIR)/$(INDIRECT).m
        if [ ! $(NOPUSH) ]; then \
                $(YPPUSH) $(INDIRECT); \
                date “+$(PUSHFMT) $(INDIRECT) $(DATEFMT)”;\
        fi

# HOME map:
$(HOME):        $(DIR)/$(HOME)
        date “+$(UPDATEFMT) $(HOME) $(DATEFMT)”
        $(O_MAKEMDBM) $(DIR)/$(HOME) $(YPDBDIR)/$(HOME).m
        if [ ! $(NOPUSH) ]; then \
                $(YPPUSH) $(HOME); \
                date “+$(PUSHFMT) $(HOME) $(DATEFMT)”;\
        fi
```

Also, add the following lines in the server’s `/var/ns/nsswitch.conf.nisserv` file (or `/var/ns/domains/`*domainname*`/nsswitch.conf`) so client systems can query the maps:

```
auto_master:            nisserv
auto.direct:            nisserv
auto.home:              nisserv
auto.test:              nisserv
```

Send the NSD daemon a SIGHUP signal for the change to take effect and restart `nsadmin`:

```
# killall -HUP nsd
# nsadmin restart
```

Typically, `/var/yp/local.make.script` filters each readable ASCII file for which a map is to be built (such as `/etc/auto.master`) through `awk`, `sed`, and/or `grep` to make two databases suitable for input to `makemdbm`. For example, the database might be stored as `/var/ns/domains/circles/auto.master.m.`

To create a customized make script, `/var/yp/local.make.script`, use the existing `/var/yp/local make.script.demo` as a source of programming examples. Make use of mechanisms already in place in `/var/yp/localmake.script.demo` when deciding how to create dependencies that the `make` command recognizes; specifically, using `.time` files allows you to see when the script was last run for the map.

If new maps are to propagate properly on slave servers, `ypxfr` shell scripts must contain the appropriate entries. To get an initial copy of the map, run `ypxfr` manually on each slave server. A map must be available on all servers before clients begin to access it. If a map is unavailable on some NIS servers, client programs may behave unpredictably. For details on the use of these commands, refer to the `make`(1) and `ypxfr`(1M) man pages.

Propagating an NIS Map

During slave server setup, `ypinit` calls `ypxfr` to transfer maps from the master to the new slave server. Once the slave server is operating, maps can be transferred in two ways: by running `ypxfr` periodically from `crontab` or by executing `ypmake`, `ypxfr`, or `yppush` from a command line.

Periodic Propagation: crontab

The standard root `crontab`, `/var/spool/cron/crontabs/root`, has entries to run `ypxfr` periodically from shell scripts at a suggested rate for the standard maps in your NIS database. The `crontab` entries test whether the system is configured as a slave server; if the test succeeds, the `ypxfr` scripts are executed. If your NIS database has only standard maps, the default entries in root's `crontab` ensures that the maps are kept reasonably up to date. The shell scripts, by default, are run on each NIS slave server in the domain to ensure database consistency throughout the domain. The `cron` shell script entries for `ypxfr` look similar to the following example. Note that each entry in the crontab file must be seen as one line. (For documentation purposes, line wraps are indicated with a backslash [\].)

```
# If this machine is running NIS and it’s a slave server, the following
# commands keep the NIS databases up-to-date.
#
13      9       *       *       *       if /etc/chkconfig yp; then find \
/var/yp -type f -name ’xfr.*’ -mtime +1 -exec rm -f ’{}’ ’;’ ; fi
15      *       *       *       *     if test -x /var/yp/ypxfr_1ph;\
then /var/yp/ypxfr_1ph; fi
17      9,15    *       *       *     if test -x /var/yp/ypxfr_2pd;\
then /var/yp/ypxfr_2pd; fi
19      9       *       *       *     if test -x /var/yp/ypxfr_1pd;\
then /var/yp/ypxfr_1pd; fi
```

The ypxfr shell scripts reside in /var/yp. Three standard scripts are included with the NFS release: ypxfr_1phr, ypxfr_1pd, and ypxfr_2pd. These scripts transfer specified maps once per hour, once per day, and twice per day, respectively. If the rates of change are inappropriate for your environment, you can modify the root crontab to suit your needs.

Also, you should alter the crontab entries so that the exact time of the ypxfr shell executions varies from one server to another to prevent the transfers from slowing down the master server, the network, or both.

Typically, changes to the ypxfr shell scripts are required in these cases:

- To reflect required map update schedules for your site
- To add nonstandard maps
- If you want to transfer a map from a server other than the master (use the –h option on the ypxfr command)

For more information on how to use crontab, see the crontab(1) man page.

Interactive Map Propagation

The next three sections describe three methods of manually propagating NIS maps.

Using ypmake

NIS maps on the master server can be manually propagated using the `ypmake` command. This command looks at the `/var/yp/mdbm_parse` and/or `/var/yp/local.make.script` to determine which maps to make. The make script calls `makemdbm`, which updates the maps and calls `yppush`. The `yppush` command reads the `ypservers` map to determine which slave servers to contact, then it proceeds to contact `ypserv` on the selected slave servers and requests `ypxfr` service. The slave server can now transfer the maps by using `ypxfr`.

Use `ypmake` to update and propagate maps throughout your domain when you want the change to take place immediately and do not want to wait for `cron`. These are some usage examples for `ypmake`:

- To update all out-of-date maps, enter

  ```
  # /var/yp/ypmake
  ```

- To update and propagate an out-of-date `hosts.byname` and `hosts.byaddr` map, enter:

  ```
  # /var/yp/ypmake hosts
  ```

- To force the creation and propagation of a new `passwd.byname` and `passwd.byuid` map, out-of-date or not, enter:

  ```
  # /var/yp/ypmake -f passwd
  ```

- To rebuild all of the maps, but not push them to other servers, enter:

  ```
  # /var/yp/ypmake UPDATE=1 NOPUSH=1
  ```

The `ypmake` program is also automatically called for in root's `crontab`, `/var/spool/cron/crontabs/root`. The entry in `crontab` tests whether the system is configured to run NIS and whether it is configured as the master. If the test succeeds, `cron` periodically executes the `ypmake` command to update and propagate maps to the appropriate slave servers. The `crontab` entry looks similar to the following. Note that each entry in the `crontab` file must be seen as one line. For documentation purposes, line wraps are indicated with a backslash (\).

```
# If this machine is a NIS master, ypmake will rotate the
# log file and ensure that the databases are pushed out with
# some regularity.
#
1,16,31,46 *    *       *       *       if /etc/chkconfig \
ypmaster && /etc/chkconfig yp && \
test -x /var/yp/ypmake; then \
/var/yp/ypmake; fi
```

Using `ypxfr`

You can run `ypxfr` as a command on slave servers to transfer a specified map from the master or other stable server to the requesting slave server. Typically, you run `ypxfr` only in exceptional situations. For example, `ypxfr` is used when setting up a temporary NIS server to create a test environment, or when an NIS slave server has been out of service and must quickly be made consistent with the other servers.

The `ypxfr` command has options that force map transfer and specify alternate domains and servers from which to obtain the map. Below are examples of `ypxfr` command usage:

- To transfer the `hosts.byaddr` map from the master server for the map, enter:

  ```
  # /var/yp/ypxfr hosts
  ```

- To force the transfer of the `passwd.byname` map from the slave server `purple` within the domain `colors`, enter:

  ```
  # /var/yp/ypxfr -f -h purple -d colors
  ```

Using yppush

Although `yppush` is usually called by `ypmake`, it can also be run manually. You must run `yppush` on the NIS master server. The syntax for using `yppush` is explained below:

- To force a copy of the map `myworld` with verbose messages, enter:

  ```
  # yppush -v myworld
  ```

- To force a copy of the map `yourmap` in the domain `yourworld`, enter:

  ```
  # yppush -d yourworld yourmap
  ```

Use `yppush` to force a copy of an updated version of a specified map from the master server to the slave servers. It can be used to move an infrequently changed, nonstandard map from the master server to slave servers.

In any of the cases mentioned above, you can capture the transfer attempts of `ypxfr` and the results in a log file. If `/var/yp/ypxfr.log` exists, `ypxfr` appends results to it. No attempt is made to limit the log file; you are in charge of that. To turn off logging, remove the log file. In addition, the file `/var/yp/ypmake.log` records `ypmake` transactions. This file can also be useful for troubleshooting propagation problems.

Adding an NIS Slave Server

To add a new NIS slave server, you must first modify an NIS server map on the NIS master server. If the new server has not been an NIS slave server before, you must add the new server's name to the map `ypservers` in the default domain.

This procedure explains how to add a new server to an NIS configuration:

1. On the master server, change to the `/var/yp` directory:

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

2. Create a new `hosts` map, if needed.

   The new server's host name and address must be in the `hosts` map. If the NIS slave server you are adding is not included in the `hosts` map, edit `/etc/hosts` and save your changes. Then, create a new `hosts` map:

   ```
   # vi /etc/hosts
   ```

   Enter and save your changes:

   ```
   # ./ypmake hosts
   ```

3. Edit the `/var/ns/domains/domainname/ypservers` file and add the new server's host name:

   ```
   # vi ypservers
   ```

4. Propagate the map with `ypmake`:

   ```
   # ./ypmake ypservers
   ```

5. Transfer the database from the master server.

   Remotely log in to the new NIS slave server. Use `ypinit` to transfer the database from the NIS master server to the new slave server:

   ```
   # /var/yp/ypinit -s mastername
   ```

6. Perform the steps described in "Building the Duplicate Maps" on page 38. The new slave server is ready for service after you build the duplicate maps.

Changing the Master Server

To switch the master server to a different system, you must rebuild all maps to reflect the name of the new master server and distribute the new maps to all slave servers.

To change the master server, perform these steps:

1. Set up the system that is to be the new master server as if it is to be a slave server. See "Setting Up NIS Slave Servers" on page 37 and follow the directions in the sections "Setting the Slave Server's Domain Name," "Binding to Another NIS Server," and "Building the Duplicate Maps."
2. Copy the map source files from the old master server to the new master server. The source files are listed in Table 3-1.
3. Rebuild all of the maps on the new master server, but do not push them to other servers:

   ```
   newmaster# /var/yp/ypmake UPDATE=1 NOPUSH=1
   ```
4. Use `ypxfr` on the old master server to transfer each of the new maps from the new master server to the old master server. Give this command for each of the maps listed in "Standard and Nonstandard Maps" on page 13:

   ```
   oldmaster# ypxfr -h newmaster -f mapname
   ```

   newmaster is the host name of the new master server and *mapname* is a map name from "Standard and Nonstandard Maps" on page 13. `ypxfr` is used for this step rather than `yppush` because of a security feature of `yppush`. When a map is pushed to a server, that server consults its own copy of the map to verify that the map is coming from the master server. Since the old master server still believes that it is the master server, it will not accept maps from the new master server.
5. On the old master server, transfer copies of the new maps to all slave servers by giving this command for each of the maps listed in "Standard and Nonstandard Maps" on page 13:

   ```
   oldmaster# yppush mapname
   ```

   Maps are pushed from the old master server to the slave servers because the slave servers' maps still contain the old master server. The new maps contain the name of the new master server.

Using Secure ypset

The ypset tool allows the root user on NIS clients to change the binding association for the client. By default, ypset is now an attribute, and to obtain the functionality equivalent to the previous ypserv command the function is set in this manner:

```
(nis_security=true)
```

To enable changing the binding association in the domain shapes, first verify the domain name, then set ypset with these commands:

```
# domainname
shapes
```

```
# nsd -a (nis_security)
```

To enable changing the binding association at the server level, edit the server password file so that it reads:

```
passwd (nis_security)
```

To enable changing the binding association at the local level, edit the /etc/nsswitch.conf to include this line:

```
(nis_security=true)
```

If you desire to mimic the previous behavior of ypsetme, replace

```
(nis_security=true)
```

in the previous examples with either of the following:

```
(nis_security=false)
```

```
(nis_security=local)
```

The result is equivalent to the previous **ypsetme**.

Previously the file `/etc/config/ypbind.options` contained the `–ypsetme` option that enabled `ypset`. Normally, the `–ypsetme` option should be present when creating an NIS master because, if it is not present, `ypmake` displays error messages when building an NIS master. In secure installation sites, however, the `–ypsetme` option should be removed.

The `ypset` tool was designed for debugging and not for casual use. As with any network tool that bases security on IP address checking, `ypset` can compromise security on networks where packets may be introduced to the network by nontrusted individuals.

Chapter 6

Troubleshooting NIS

This chapter provides information to be used in troubleshooting the NIS environment. The chapter is divided into two parts: problems seen on an NIS server and problems seen on an NIS client. Each section describes general trouble symptoms followed by a discussion of probable causes.

This chapter contains these sections:

Debugging an NIS Server

Before trying to debug an NIS server, be sure you understand the concepts in Chapter 1, “Understanding NIS,” and Chapter 2, “Preparing to Manage NIS,” in this guide.

Different Map Versions

Since NIS works by propagating maps from the NIS master server to NIS slave servers within the same domain, you may find different versions of a map on different servers. Each time a map is updated, a new order number (map version) is attached to the map. This information can be obtained with the `yppoll` command.

Version skew, or out-of-sync maps, between servers is normal when maps are being propagated from the NIS master server to the slave servers. However, when the maps on different servers remain unsynchronized even after the NIS environment has stabilized, it usually indicates a problem.

The normal update of NIS maps is prevented when an NIS server or some gateway system between the NIS master server and NIS slave servers is down during a map transfer attempt. This condition is the most frequent cause of out-of-sync maps on servers. Normal update procedures are described in Chapter 5, “Maintaining NIS.” When all the NIS servers and all the gateways between the NIS master and NIS slave servers are up and running, `ypxfr` should successfully transfer maps and all NIS servers’ maps should be in sync.

The next section describes how to use `ypxfr` manually to update NIS maps. If `ypxfr` transfers maps successfully when it is initiated manually but still fails intermittently, it requires additional investigation on your part, which is described in the section, “Intermittent, Consistent Map Propagation Failures” on page 67.

Isolated, One-Time Map Propagation Failures

If a particular slave server has an isolated, one-time problem updating a particular map or its entire map set, follow these steps to resolve the problem by running `ypxfr` manually:

1. `ypxfr` requires a complete map name rather than a nickname, so get a list of complete map names for maps in your domain, by giving this command:

   ```
   # ypwhich -m
   ```

The system returns a list of complete map names and the name of the NIS master server for each map. Output should be similar to this output for an NIS master server named `circles`:

```
ypservers circles
netid.byname circles
bootparams circles
mail.aliases circles
netgroup.byhost circles
netgroup.byuser circles
netgroup circles
protocols.byname circles
protocols.bynumber circles
services.byname circles
rpc.bynumber circles
networks.byaddr circles
networks.byname circles
ethers.byname circles
ethers.byaddr circles
hosts.byaddr circles
hosts.byname circles
group.bygid circles
group.byname circles
passwd.byuid circles
passwd.byname circles
mail.byaddr circles
```

2. For each map that is not being updated, transfer the map manually using `ypxfr`:

   ```
   # ypxfr -f map.name
   ```

   map.name is the complete name of the map, for example, `hosts.byname`.

   If `ypxfr` fails, it supplies an error message that points you to the problem. If it succeeds, you should see output similar to this:

   ```
   Transferred map hosts.byname from NIS_master (1091 entries).
   ```

Intermittent, Consistent Map Propagation Failures

This section describes several procedures you can use to help isolate intermittent map propagation problems.

If the error message `Transfer not done: master’s version isn’t newer` appears, check the dates on the master and slave servers.

On the NIS master server, check to ensure that the NIS slave server is included in the `ypservers` map within the domain. If the slave server is not in the `ypservers` map, the master server does not know to propagate any changed and updated maps automatically to the server. If the server has the correct entry in its `crontab` file to have `ypxfr` request updated maps from the master server, the slave server gets the updated maps, but this action is not initiated by the NIS master server. These steps illustrate how to verify the `ypservers` map:

1. Review the contents of the ASCII file used to create the `ypservers` map:

   ```
   # cat /var/yp/ypservers
   ```

   If the server is not listed, add the server's name using any standard editor.

2. Once the `/var/yp/ypservers` file has been edited, if necessary, ensure that the actual map is updated on the master server. This is a special map and no attempt is made to push it to the other servers. Give this command:

   ```
   # /var/yp/ypmake -f ypservers
   ```

Another possible reason for out-of-sync maps is a bad `ypxfr` script. Inspect root's `crontab` (`/var/spool/cron/crontabs/root`) and the `ypxfr` shell scripts it invokes (`/var/yp/ypxfr_1ph`, `/var/yp/ypxfr_1pd`, and `/var/yp/ypxfr_2pd`). Typographical errors in these files can cause propagation problems, as do failures to refer to a shell script within `crontab`, or failures to refer to a map within any shell script. Also ensure that the configuration flags are on for `yp` and `nsd` with the `chkconfig` command. For details see the `chkconfig`(1M) man page.

Finally, if the above suggestions do not solve the intermittent map propagation problem, you need to monitor the `ypxfr` process over a period of time. These steps show how to set up and use the `ypxfr` log file:

1. Create a log file to enable message logging. Enter these commands:

   ```
   # cd /var/yp
   # touch ypxfr.log
   ```

   This saves all output from `ypxfr`. The output looks much like the output from `ypxfr` when run interactively, but each line in the log file is timestamped. You may see unusual ordering in the timestamps. This is normal; the timestamp tells you when `ypxfr` began its work. If copies of `ypxfr` ran simultaneously, but their work took differing amounts of time, they may actually write their summary status line to the log files in an order different from the order of invocation.

Any pattern of intermittent failure shows up in the log. Look at the messages to determine what is needed to fix the failure. You know that you have fixed it when you no longer receive failure messages.

2. When you have fixed the problem, turn off message logging by removing the log file. Give this command:

```
# rm ypxfr.log
```

Note: If you forget to remove the log file, the log file grows without limit.

As a last resort and while you continue to debug, you can transfer the map using the remote file copy command, `rcp`, to copy a recent version from any healthy NIS server. You may not be able to do this as *root*, but you probably can do it by using the `guest` account on the master server. For instance, to copy the map `hosts` in the domain `shapes.com` from the master server `circles` to the slave server `squares`, enter this command:

```
# rcp guest@circles:/var/ns/domains/shapes.com/hosts.\* \
/var/ns/domains/shapes.com
```

The escaped asterisk (*) allows the remote copy of all `mdbm` record files for the `hosts` map.

nsd Fails

If `nsd` fails almost immediately each time it is started, look for a more general networking problem. Because NIS uses Remote Procedure Calls (RPC), the portmapper must be functioning correctly for NIS to work.

To verify that the portmapper is functioning and that the `nsd` protocol is registered with the portmapper, enter this command on the server:

```
# /usr/etc/rpcinfo -p | grep ypserv
```

If your `portmap` daemon is functional, the output looks something like this:

```
100004    2    udp    1051    ypserv
100004    2    tcp    1027    ypserv
```

If these entries are not in your output, `nsd` has been unable to register its services with the `portmap` daemon. If the `portmap` daemon has failed or is not running, you get this error message:

```
rpcinfo: can’t contact portmapper: Remote system error - connection
refused
```

If the information returned by `rpcinfo` does not match the information shown above or if the error message is returned, reboot the server. Rebooting the server ensures that the network daemons, specifically `portmap` and `nsd`, are started in the correct order. See the `nsd`(1M), `portmap`(1M), and `rpcinfo`(1M) man pages for further details.

Debugging an NIS Client

Before trying to debug an NIS server, be sure you understand the concepts in Chapter 1, “Understanding NIS,” and Chapter 2, “Preparing to Manage NIS,” in this guide.

Command Hangs

The most common problem on an NIS client is for a command to hang and generate SYSLOG messages such as this:

```
NIS v.2 server not responding for domain domain_name; still trying
```

Sometimes many commands begin to hang, even though the system as a whole seems to be working and you can run new commands.

The messages above indicates that `nsd` on the local system is unable to communicate with `nsd` in the domain *domain_name*. This can happen as a result of any of these situations:

- The network has been disconnected on the NIS client; for example, the Ethernet cable is unplugged.
- An incorrect domain name has been specified.
- The network or the NIS server is so overloaded that `nsd` cannot get a response back to the `nsd` daemon within the time-out period.
- `nsd` on the NIS server has crashed.

- The NIS server has crashed or is unreachable via the network.
- There is a physical impairment on the local area network. Under these circumstances, all the other NIS clients on the same local area network should show the same or similar problems.

A heavily loaded network and/or NIS server may be a temporary situation that might resolve itself without any intervention. However, in some circumstances, the situation does not improve without intervention. If intervention becomes necessary, the following four questions help to isolate and correct the situation.

Question 1: Is the client attached to the network?

Typically, if there is a problem with the physical connection from the client to the network, a message similar to this appears in the console window on the system:

```
ec0: no carrier: check Ethernet cable
```

If NIS commands hang and you have the message shown above, verify that the physical connection from the client to the local area network is secure and functioning. If you do not know how to check your physical connection, see the *Owner's Guide* for your system for more details. Also check to ensure that the client is attached to the correct physical network.

Question 2: Does the client have the correct domain set?

Clients and servers must use the same domain name if they want to belong to the same domain. Servers supply information only to clients within their domain. The domain names must match exactly. The domain `shapes.com` is not the same as the domain `SHAPES.com`. Clients must use a domain name that the NIS servers for their domain recognize.

Verify the client's current domain name by giving the `domainname` command and by looking at the contents of the file `/var/yp/ypdomain`, which is read at system startup. Perform these steps to determine the client's current domain:

1. Determine the current domain name:

   ```
   # domainname
   current_domain_name
   ```

2. Look at `/var/yp/ypdomain` to determine the domain name set at system startup:

   ```
   # cat /var/yp/ypdomain
   current_domain_name
   ```

Compare these values to those found on the servers. If the domain name on the client differs from the domain name on the server, change the domain on the client:

1. Edit, using any standard editor, `/var/yp/ypdomain` to reflect the correct domain name. This file assures that the domain name is correctly set every time the client boots. There should be only one entry in this file:

   correct_domain_name

2. Set *domainname* by hand so it is fixed immediately. Give this command:

   ```
   # domainname correct_domain_name
   ```

3. Restart `nsd` so that the client is bound within the correct domain. Give these commands:

   ```
   # /etc/killall -HUP nsd
   ```

Question 3: Do you have enough NIS servers?

NIS servers do not have to be dedicated systems; and as multipurpose systems, they are susceptible to load escalations. If an NIS server is overloaded, the client's `nsd` process automatically switches to another less heavily loaded server. Check to ensure that designated servers are functioning and accessible via the network.

By default, when an NIS client boots it can only bind to a server that resides on the same local network. It cannot bind to a server that resides on a remote network. There must be at least one NIS server running on the local network in order for a client in the same domain to bind. Two or more NIS servers per local network improve availability and response characteristics for NIS services.

Question 4: Are the NIS servers up and running?

Check other clients on your local network. If several client systems have NIS-related problems simultaneously, suspect the NIS server. It may be that the NIS server system is down or inaccessible or that the `nsd` process has crashed on the NIS server.

If an NIS server crashes or becomes unavailable, it should not affect NIS performance if there are multiple NIS servers on a network. The clients automatically switch to another server. If there is only one server on the network, check to ensure that the server is up by remotely logging in to the server.

If the server is up, the problem may be that the `nsd` process has crashed on the server. Enter these commands to find out if `nsd` is running and restart it if it is not:

1. Log into the NIS server system. Look for `nsd` processes. Give this command:

   ```
   # ps -ef | grep nsd
   ```

   You should see output similar to this:

   ```
   root   128     1  0  Sep 13  ?        1:35 /usr/etc/nsd
   ```

2. If the server's `nsd` daemon is not running, start it up by typing:

   ```
   # nsadmin restart
   ```

3. Give the command `ypwhich` on the NIS server system:

   ```
   # ypwhich
   ```

   If `ypwhich` returns no answer, `nsd` is probably not working.

4. If `nsd` is not working, give this command to kill the existing `nsd` process and start a new one:

   ```
   # nsadmin restart
   ```

NIS Command Fails

Another problem that can occur on an NIS client is for a command to fail due to a problem with the NIS daemon, `nsd`. These examples illustrate typical error messages you might see when you give an NIS command and `nsd` has failed:

```
# ypcat hosts
ypcat: can't bind to NIS server for domain domain_name.
Reason: can't communicate with nsd.
# yppoll aliases
Sorry, I can't make use of the NIS. I give up.
```

In addition to the preceding error messages, these general symptoms may also indicate that the `nsd` process has crashed:

- Some commands appear to operate correctly while others terminate, printing an error message about the unavailability of NIS.
- Some commands work slowly in a backup-strategy mode peculiar to the program involved.
- Some commands do not work and/or daemons crash with obscure messages or no message at all.

To correct this situation, stop and restart the `nsd` process on the client with the following command:

```
# nsadmin restart
```

Give this command to verify that the `nsd` process is running:

```
# ps -ef | grep nsd
```

You should see output similar to this:

```
root 26995     1  0 17:35:31 ?        0:00 /usr/etc/nsd
```

`ypwhich` Output Inconsistent

When you enter the `ypwhich` command several times on the same client, the answer you receive may vary because the NIS server has changed. This response is normal. The binding of an NIS client to an NIS server changes over time on a busy network and when the NIS servers are busy. Whenever possible, the system stabilizes at a point where all clients get acceptable response time from the NIS servers. As long as the client gets NIS service, it does not matter where the service comes from. An NIS server may get its own NIS services from another NIS server on the network.

Before You Call for Help

Before you call your support provider, please use the recommendations in this chapter for solving your problems independently. If your problems persist and you find it necessary to call, please have this information ready:

- System serial number.
- Operating system and NFS version numbers (from `versions`). Include `eoe` and `nfs`.
- A specific description of the problem. Write down and be prepared to provide any error messages that might help in isolating the problem.
- Are there other vendors' systems involved?
- What does the physical layout look like? Are there gateways?
- How many slave servers do you have per network?

- What are the names of the master server, slave server(s), and domain?
- How many systems are in your domain?
- Do you have multiple domains?

Index