NIS Administration Guide

CONTRIBUTORS

Written by Kim Simmons, Pam Sogard, and Susan Ellis
Edited by Nancy Schweiger
Production by Kay Maitz
Engineering contributions by Andrew Cherenson and Dana Treadwell
Other contributions by Joe Yetter

NIS Administration Guide
Document Number 007-2161-001

Contents

Contents

Figures

Figures

Tables

x

About This Guide

The *NIS Administration Guide* documents the Silicon Graphics®
implementation of the network information service NIS. NIS is adapted
from Sun Microsystems, Inc.'s 4.0 release of NIS, with bug fixes from release
4.1. This guide was prepared in conjunction with the release of IRIX™
Release 5.1.

NIS is a database service that provides location information about network
entities to other network services, such as 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 the Network File
System, NFS™, as well as NIS. Information about NFS is now published in
a separate volume called the *NFS Administration Guide*.

Using This Guide

The purpose of this guide is to provide the information needed 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 In-1 contains a summary of each chapter in this guide and suggests how to use the chapter.

Table In-1   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.

Table In-1 (continued)   Contents of Each Chapter

Chapter	Summary	When to Read
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 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

- Silicon Graphics network implementation

Supplementary Documentation

You can find supplementary information in these documents and books:

- *IRIX Advanced Site and Server Administration Guide* (Silicon Graphics publication) explains the fundamentals of system and network administration for Silicon Graphics systems on a local area network.

- *NFS Administration Guide* (Silicon Graphics publication) explains how to set up and maintain Silicon Graphics' implementation of the Network File System.

- Stern, Hal, *Managing NFS and NIS*, O'Reilly & Associates, Inc. 1991. This book contains detailed, but not Silicon Graphics-specific, information about NIS and how to administer and use it.

Conventions

This guide uses several font conventions:

italics   Italics are used for command and manual page names, filenames, variables, host names, user IDs, map names, and the first use of new terms.

Courier   Courier is used for examples of system output and for the contents of files.

Courier bold   Courier bold is used for commands and other text that you are to type literally.

Product Support

Silicon Graphics offers a comprehensive product support and maintenance program for IRIS® products. For information about using support services for this product, refer to the *Release Notes* that accompany it.

Understanding NIS

This chapter contains a general description of the Silicon Graphics 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:

- "What Is NIS?" on page 1
- "NIS Portability" on page 2
- "The Client-Server Model" on page 4
- "The Server Hierarchy" on page 4
- "NIS Maps" on page 5
- "NIS Domains" on page 6

What Is 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. Typically, the default lookup order is NIS first, DNS (BIND) second, then the appropriate local files.

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 or User Datagram Protocol 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

The 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.

The 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 comprised 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 (*dbm*(3B)). Since data in *dbm* format is faster to find than ASCII data, using *dbm* 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; *values* are attributes of the key returned from the query. For example, in the map called *hosts.byname*, the keys are the names of individual systems, and the values are their Internet addresses. A query on the system *triangle* might return the value 192.0.2.5.

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*.
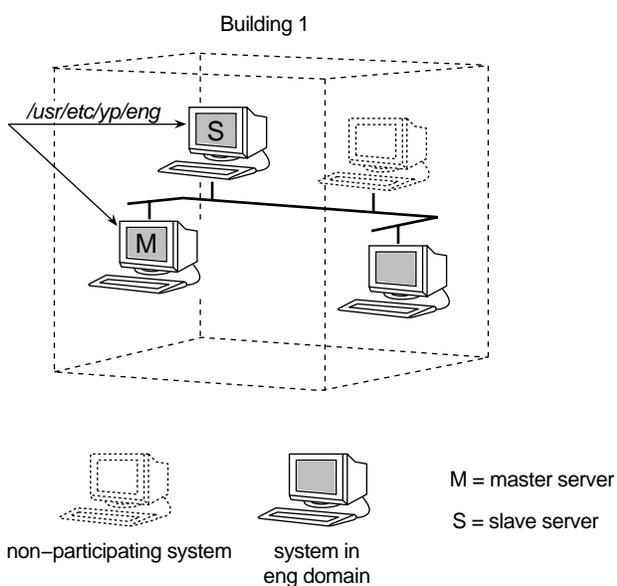
Figure 1-2   Basic NIS Domain

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/yp*. 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/yp/eng*.

NIS and Internet Domains

NIS domains can be organized to coincide with domains of the Internet, a registered organization of wide area networks originally funded by DARPA (U.S. Defense Advanced Research Projects Agency). In the Internet naming hierarchy, a domain name is comprised of a stream of names separated by dots. For example, *eng.widgets.com* is an Internet domain. This could also be the name of the NIS domain.

Some administrators name their NIS domains with simple names and the Internet domain names separated with dots, for example, *eng* for the NIS domain name and *eng.widgets.com* for the Internet domain name. Using this naming scheme, the two domains can be easily distinguished. Other administrators prefer to have the NIS and Internet domain names coincide. This is strictly a matter of choice; there is no explicit relationship between NIS and Internet domains. However, to avoid confusion, choosing one of these two naming schemes is recommended. (See the *IRIX Advanced Site and Server Administration Guide* for details on Internet domains.)

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*.
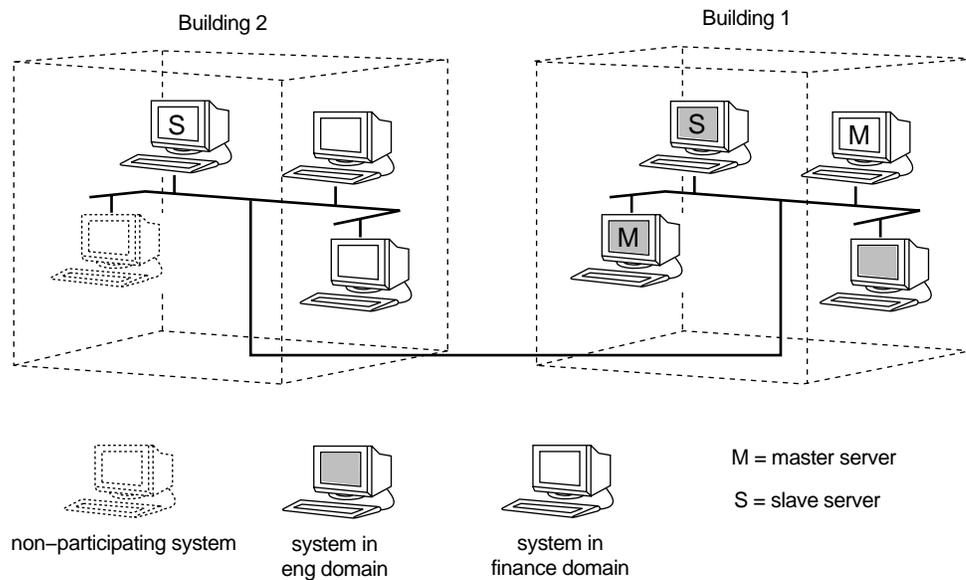
Figure 1-3   Multiple NIS Domains

The master and slave server for the *eng* domain both contain a database of information for the *eng* domain in */var/yp/eng*, and the master and slave server for the *finance* domain both contain a copy of the database for the *finance* domain in */var/yp/finance*. Notice that one system n 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.)

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

	Client	Slave	Master
ypbind	X	X	X
ypserv		X	X
rpc.passwd			X

The binder daemon, *ypbind*(1M), runs on all NIS clients. This daemon is responsible for remembering information necessary for communicating with the NIS server process, *ypserv*(1M).

The server daemon, *ypserv,* runs on all NIS servers. The *ypserv* daemon is the database server and is responsible for answering client inquiries. It is also responsible for managing database updates. Most NIS servers are also NIS clients; they use the NIS database information. As NIS clients, NIS servers also run the binder daemon, *ypbind*.

The NIS master server also runs the server process daemon, *ypserv,* to answer client inquiries. The NIS master server typically runs the binder process daemon, *ypbind*, to solicit information from the NIS database. The master server also runs a third daemon, */usr/etc/rpc.passwd*, which allows NIS users to remotely modify their NIS password with *yppasswd*(1) and to modify some other password file fields with *ypchpass*(1).

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 with the *chkconfig*(1M) command). There are three *chkconfig* flags for NIS: *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 *ypserv* 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, *ypbind*, and the server daemon, *ypserv,* determines whether or not an NIS connection is bound or unbound. A brief summary of the binding process is given below.

An application, through library routines, contacts the local *portmap*(1M) program to determine the appropriate port number for *ypbind*. The *portmap* program returns the port number to use to contact *ypbind*. Port numbers are defined in */etc/rpc*. Once the local system has the port number for *ypbind*, it requests an IP address and port number for an NIS server process, *ypserv,* within its domain.

To obtain the IP address and port number for the NIS server process, *ypbind* 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, *ypserv,* are remembered by the *ypbind* process and used to obtain NIS database information. The association between the binder daemon, *ypbind*, and the server daemon, *ypserv,* must be present for binding to occur.

Figure 2-1 illustrates the binding process initiated for an *ls*(1) 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*(3C), 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.
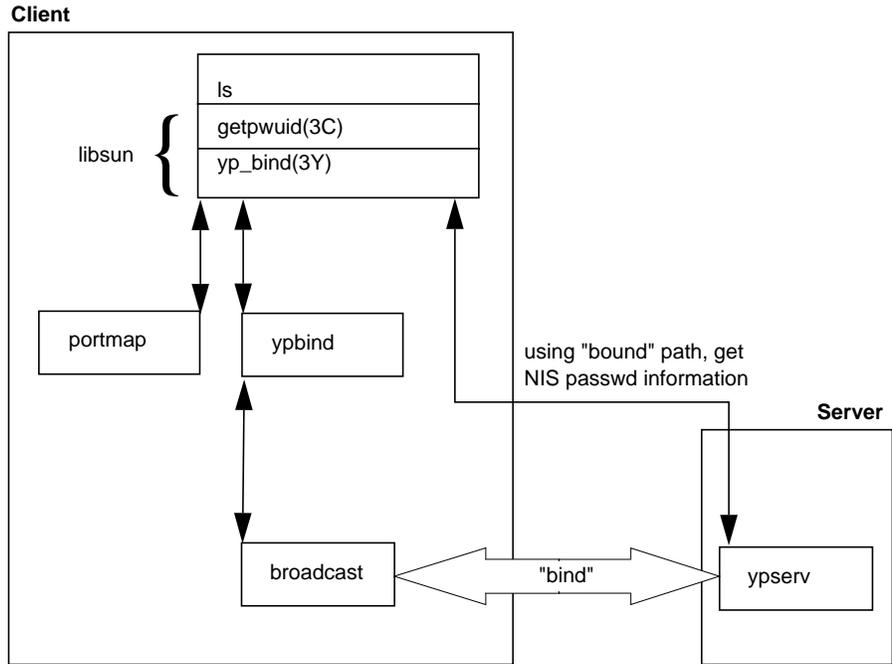
Figure 2-1   NIS Binding Process

When a client boots, *ypbind* broadcasts, by default, asking for an NIS server. Similarly, *ypbind* broadcasts asking for a new NIS server if the old server fails to respond. The *ypwhich*(1) command gives the name of the server to which *ypbind* is currently bound.

The NIS Database

The NIS database is a collection of files in *dbm* format. To create the database, the NIS tool *makedbm*(1M) converts input files (usually ASCII) to pairs of output files. The output files have *.dir* and *.pag* extensions. Each pair is a map. For example, the *aliases* map is composed of the files *aliases.dir* and *aliases.pag*.

A typical listing of NIS database files looks like this:

```
aliases.dir        host.addr.dir      netid.byname.dir   pw.name.dir
aliases.pag        host.addr.pag      netid.byname.pag   pw.name.pag
bootparams.dir     host.name.dir      nets.addr.dir      pw.uid.dir
bootparams.pag     host.name.pag      nets.addr.pag      pw.uid.pag
ether.addr.dir     mail.byaddr.dir    nets.name.dir      rpc.nbr.dir
ether.addr.pag     mail.byaddr.pag    nets.name.pag      rpc.nbr.pag
ether.name.dir     netgrp.dir         proto.name.dir     svc.name.dir
ether.name.pag     netgrp.pag         proto.name.pag     svc.name.pag
grp.gid.dir        netgrp.hist.dir    proto.nbr.dir      ypservers.dir
grp.gid.pag        netgrp.hist.pag    proto.nbr.pag      ypservers.pag
grp.name.dir       netgrp.usr.dir     pw.name.dir
grp.name.pag       netgrp.usr.pag     pw.name.pag
```

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.

Table 2-2 shows the standard NIS maps.

Table 2-2   Standard Set of NIS Maps

bootparams	*hosts.byname*	*netid.byname*	*protocols.bynumber*
ethers.byaddr	*mail.aliases*	*networks.byaddr*	*rpc.bynumber*
ethers.byname	*mail.byaddr*	*networks.byname*	*services.byname*
group.bygid	*netgroup.byhost*	*passwd.byname*	*ypservers*
group.byname	*netgroup.byuser*	*passwd.byuid*	
hosts.byaddr	*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 2-3 shows the list of default nicknames and full names for maps supplied in the NIS release.

Table 2-3   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** since 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 with *crontab*(1) 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 *ypmake*(1M), *ypmake* looks at */var/yp/make.script* and */var/yp/local.make.script* (if it exists) to determine which maps to make. */var/yp/make.script* calls *makedbm*(1M), which updates the maps and calls *yppush*(1M). *yppush* reads the *ypservers* map to determine which slave servers to contact; *yppush* contacts *ypserv* on the selected slave servers and requests *ypxfr*(1M) service. The slave server can now transfer the maps with *ypxfr*.

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

Master Server **Slave Server**

```
  start ──▶ ypmake
               │
               ▼
          make.script
               │
               ▼
           makedbm
               │
               ▼
           yppush ──transfer request──▶ ypserv
                                           │
                                           ▼
          ypserv ◀──transfer request──   ypxfr
                 ──map transferred──▶
```
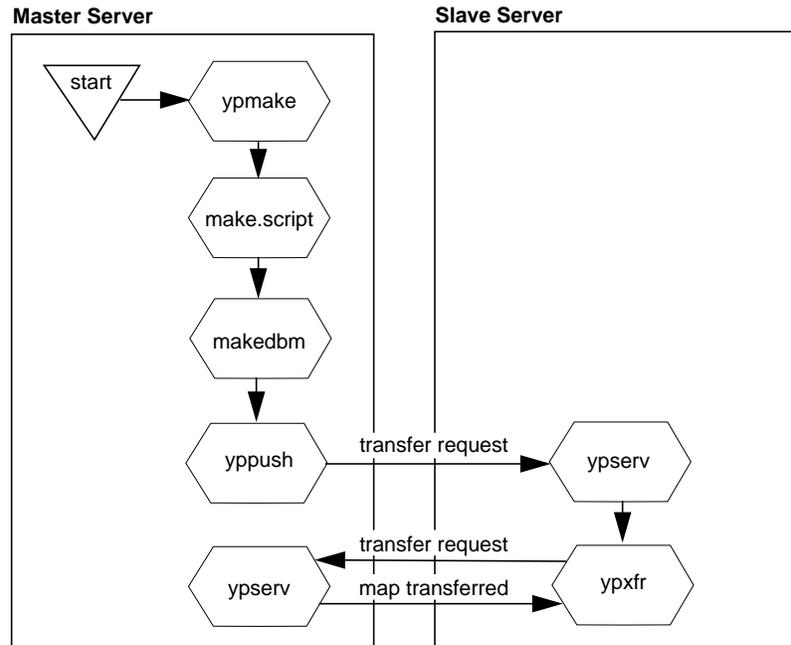
Figure 2-2   Map Propagation between Servers

NIS and Other Network Files

When a system becomes an NIS client, it completely relinquishes control over some network-related files and maintains varying levels of control over others. The amount of control relinquished on certain files depends upon the NIS syntax used within the file.

These network-related files can be divided into two groups: local files and global files. Local files are those files that NIS first checks on the local system and may continue checking in the NIS database. Global files are those files that reside in the NIS database and are always consulted by programs using NIS. The next two sections discuss the local and global files consulted by NIS. More information on these configuration files is included in the *IRIX Advanced Site and Server Administration Guide*.

Local Files

Table 2-4 shows the local files that NIS consults. Local files allow levels of control. For example, a program that calls *getpwent*(3C) to access */etc/passwd* (a local file) first looks in the password file on your system; the NIS password file is consulted only if your system's password file contains a plus sign (+) entry (see *passwd*(4)).

Table 2-4   Local Files Consulted by NIS

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

If shadow passwords are used (*/etc/shadow* exists on the local system), the NIS password file isn't consulted at all.

The */etc/group* and */etc/passwd* files may have plus (**+**) or minus (**–**) entries to designate levels of local control. Table 2-5 shows some examples of **+/–** entries for the local */etc/group* and */etc/passwd* files. Note that the position of **+/–** entries in the files does affect processing. The first entry, **+/–** or regular, is the one that is used.

Table 2-5   Local File Entries to Control Access

Local File	Example Entry	Meaning of the Entry
/etc/group	`+:`	Get all group information from the NIS group database.
	`+angels:::mark,john`	Assign the users *mark* and *john* to the group *angels*. *mark* and *john*'s group password (if any) and group ID is taken from the NIS *angels* group entry.
	`-spies`	Disallow subsequent entries, both local and NIS, for the group *spies*.
/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*.

17

In *etc/hosts.equiv,* 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 *.rhosts*.

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

Global Files

Table 2-6 shows the global files that NIS consults. Local copies of these files are ignored when NIS is running, except in the cases noted below.

Table 2-6   Global Files Consulted by NIS

Global Files
etc/bootparams
etc/ethers
etc/hosts
etc/netgroup
etc/networks
etc/protocols
etc/rpc
etc/services

The information in global files is network-wide data, and only NIS accesses it. However, you must make sure each client has an entry for itself in */etc/hosts* when booting. If NIS is running, NIS checks only global files; NIS does not consult global files on your local system. For *bootparams*, *services* and *rpc*, the files on the local system are consulted only if the answer is not in the NIS database. By default, the local file */etc/hosts* is not consulted if NIS is running. To change the default, edit */etc/resolv.conf* as specified in the *resolver*(4) manual page.

NIS Software Quick Reference Guide

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

NIS Daemons

rpc.passwd(1M) A server process that allows users with NIS accounts to change their NIS password and other NIS password-related fields.

ypbind(1M) The binder process that runs on all systems that use NIS services, both servers and clients. It remembers information that lets client processes on a single node communicate with some *ypserv* processes.

ypserv(1M) The NIS database server process that must run on each NIS server system. It looks up information in its local database of NIS maps.

NIS Configuration Files

/var/yp The default location of NIS database files. For more information, see *ypfiles*(4).

/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 *rpc.passwd*(1M).

/etc/config/ypbind.options

Specifies default options to use with *ypbind*. Options that can be included in this file are a secure mode and the name or IP address of the NIS server to bind to at system startup. This eliminates the need for an NIS server on each network. For more information, see *ypbind*(1M).

/etc/config/ypmaster.options

        Alters default NIS variables, including the NIS password filename, the aliases filename, the source file directory, additional domains, to push maps or not, and the directory containing NIS programs. For more information, see *ypmake*(1M).

/etc/config/ypserv.options

        Specifies non-default options to use with *ypserv*. Options that can be included in this file are forklimits, operation with *named*(1M) and/or DNS, log information, and verbose output. For more information, see *ypserv*(1M).

NIS Tools

makedbm(1M)   A low-level tool for building a *dbm* file that is a valid NIS map. You can use *makedbm* to build or rebuild databases not built from */var/yp/make.script*. You can also use *makedbm* 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 *makedbm*. Create the local site make script, */var/yp/local.make.script*, to avoid using this low-level tool except in rare circumstances.

ypcat(1)   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, *rlogin*(1C) or *rsh*(1C) to that server and use *makedbm*.

ypchpass(1)   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*(1) to change your NIS password.

ypinit(1M)	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.
ypmake(1M)	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*. */var/yp/ypmake.log* is the log file for all *ypmake* activity.
ypmatch(1)	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.
yppasswd(1)	Allows NIS users to remotely change their NIS passwords.
yppoll(1M)	Asks any *ypserv* for the information it holds internally about a single map.
yppush(1M)	Runs on the master NIS server. It requests each of the *ypserv* 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.
ypset(1M)	Tells a *ypbind* process (the local one, by default) to get NIS services for a domain from a named NIS server. By default, *ypbind* disallows the use of *ypset*. See the *ypset*(1M) manual page for details on enabling *ypset*.
ypwhich(1)	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.

ypxfr(1M)   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*(1M). Also, *ypserv* 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.

In addition to these NIS tools, the *rpcinfo*(1M) and *crontab* tools are also useful for administering NIS.

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 a master and slave servers. Finally, this chapter provides general recommendations to help you make planning decisions.

This chapter contains these sections:

- "Establishing Multiple NIS Domains" on page 23
- "Verifying ASCII File Contents" on page 28
- "Selecting a Domain Name" on page 30
- "Selecting the NIS Master Server" on page 30
- "Selecting the NIS Slave Servers" on page 31
- "General Recommendations" on page 32

Establishing Multiple NIS Domains

Before you set up NIS, you need to 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* that provide a host name-to-address mapping.
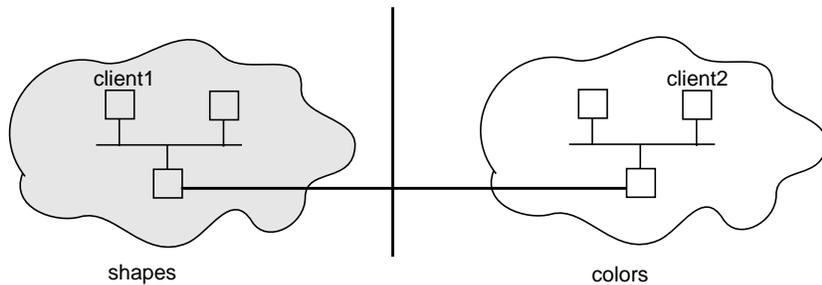
Figure 3-1   Boundary Problem with Multiple Domains

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. While there may be a physical connection between *client1* and *client2*, there is not a 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.
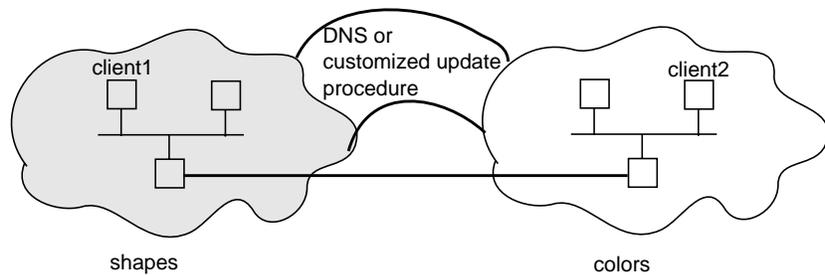
Figure 3-2   Boundary Solutions for Multiple Domains

Using the Domain Name System (DNS)

DNS, sometimes referred to as BIND (Berkeley Internet Name Daemon) or *named*(1M), 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 Advanced Site and Server Administration Guide* provides detailed information on setting up DNS.

By default, host name resolution is done by first checking NIS, If NIS isn't running, then DNS is checked. If DNS isn't running, then the local /etc/hosts file is checked. There are two methods for specifying a host resolution order other than the default:

- Use *ypserv*(1M) with the **–i** option.

- Redefine the host resolution order in */etc/resolv.conf*.

These methods are described in the next two sections.

Using *ypserv* **–i**

ypserv can be set up to access DNS systems databases as well as NIS databases. To specify a DNS database lookup, the *ypserv* process on all NIS servers must be running with the **–i** option and *all* NIS servers must also be set up as DNS clients. The **–i** option allows *ypserv* to resolve nonlocal host names and addresses (systems not in the same NIS domain) with DNS servers. This option can be set automatically at *ypserv* startup if you specify the **–i** option in the */etc/config/ypserv.options* file.

A sample */etc/config/ypserv.options* file that supports DNS looks like this:

```
-i
```

If the **–i** option is set in the */etc/config/ypserv.options* file, it causes *ypserv* to look at the NIS host database first, then the DNS hosts database to resolve a system's identity.

Using */etc/resolv.conf*

The lookup order for resolving a system's identity can be configured in a variety of ways with */etc/resolv.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.

The */etc/resolv.conf* file does not exist by default. To override the default system lookup order, you must create this file with the specified lookup order and nameserver list on all master servers, slave servers, and clients in all domains, and set up DNS servers as well.

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

```
hostresorder bind
```

If applications are to search only DNS and */etc/hosts*, put this line in */etc/resolv.conf*:

```
hostresorder bind local
```

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*, put this line in */etc/resolv.conf*:

```
hostresorder nis bind local
```

See the *resolver*(4) manual 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*(1M) 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*(1M), 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.
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 key is host name instead of Ethernet address.
group.bygid	*/etc/group*	Contains group security information with group ID as key.
group.byname	*/etc/group*	Contains group security information with group name as key.
hosts.byaddr	*/etc/hosts*	Contains host names and IP addresses, with IP address as key.
hosts.byname	*/etc/hosts*	Contains host names and IP addresses, with host name as key.
mail.aliases	*/etc/aliases*	Contains aliases and mail addresses, with aliases as key.
mail.byaddr	*/etc/aliases*	Contains mail addresses and aliases, with mail address as key.
netgroup.byhost	*/etc/netgroup*	Contains group names, user names, and host names, with host name as key.

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

Map Name	ASCII File	Description
netgroup.byuser	*/etc/netgroup*	Same as *netgroup.byhost*, except that key is user name.
netgroup	*/etc/netgroup*	Same as *netgroup.byhost*, except that key is group name.
netid.byname	*/etc/group*, */etc/hosts*, */etc/netid*	Contains user, group, and host information, with user name as 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 key is name of network.
passwd.byname	*/etc/passwd*	Contains password information with user name as key.
passwd.byuid	*/etc/passwd*	Same as *passwd.byname*, except that key is user ID.
protocols.byname	*/etc/protocols*	Contains network protocols known to your network, with protocol name as key.
protocols.bynumber	*/etc/protocols*	Same as *protocols.byname*, except that key is protocol number.
rpc.bynumber	*/etc/rpc*	Contains program number and name of RPCs known to your system. Key is RPC program number.
services.byname	*/etc/services*	Lists Internet services known to your network. Key is service name.
ypservers	*/var/yp/ypservers*	Lists NIS servers known to your network. Initially created by *ypinit* when 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*(1) 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.

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 two files to support the NIS password file, */etc/passwd.nis*: */etc/config/rpc.passwd.options* and */etc/config/ypmaster.options*. The contents of */etc/config/rpc.passwd.options* and */etc/config/ypmaster.options* should look like these examples:

```
# cat /etc/config/rpc.passwd.options
/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 the server the client should bind to at boot time.

To specify an NIS server at client startup, modify */etc/config/ypbind.options*. By default, this file contains a parameter that enables *ypset*(1M) on the local system (see the *ypset*(1M) and *ypbind*(1M) manual pages for more detailed information). The default file, when viewed with *cat*(1), contains this entry:

```
# cat /etc/config/ypbind.options
-ypsetme
```

To specify an NIS server at client startup, add the host name or IP address of the NIS server to the */etc/config/ypbind.options* file. The specified server's name must be in the client's local */etc/hosts* file for specific binding to work. The */etc/config/ypbind.options* file on a client binding to server *squares* at boot time might look like this when viewed with *cat*:

```
# cat /etc/config/ypbind.options
-ypsetme squares
```

The example above contains the option -ypsetme. If you want additional security or do not need the *ypset* tool at this time, you can remove the parameter.

General Recommendations

Below are some general recommendations for setting up NIS. As 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 for NIS use only that does not have any root UIDs. For example, specify */etc/passwd.nis* as the NIS password file.

4. To simplify administration and troubleshooting, maintain one and only 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:

- "Setting Up the NIS Master Server" on page 34

- "Setting Up NIS Slave Servers" on page 39

- "Setting Up NIS Clients" on page 42

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. 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 *squares* and *triangles*.

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*(1) command. See the *hostname*(1) manual 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:

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

2. Verify the domain name setting with *domainname*(1):

    ```
    # domainname
    shapes
    ```

    If the domain name is correctly set, the *domainname* command returns the domain name you specified in instruction 1 of this step. If your output is not correct, reissue the commands in instruction 1.

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*):

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

2. Reboot the system:

    ```
    # /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*(1M) builds NIS maps using the text files and the */var/yp/make.script*. (See Chapter 2, "Preparing to Manage NIS," for a list of the default files that are converted to maps in this step.)

1. Start building the master NIS maps with the *ypinit* command:

    ```
    # cd /var/yp
    # ./ypinit -m
    Installing the NIS data base will require that you answer
    a few questions.Questions will all be asked at the
    beginning of the procedure.
    ```

    The **–m** flag denotes that this is an NIS master server.

2. You do not want this procedure to quit on nonfatal errors, so answer **n** (no) to this prompt:

    ```
    Do you want this procedure to quit on nonfatal errors?
    [y/n: n] n
    OK, please remember to go back and redo manually whatever
    fails. If you don't, some part of the system (perhaps the
    NIS itself) won't work.
    ```

3. Enter the names of all NIS slave servers for the domain *shapes* followed by **<Ctrl-D>** in response to this prompt:

```
At this point, we have to construct a list of the hosts
which will run NIS servers. circles is in the list of NIS
server hosts. Please continue to add the names for the
other hosts, one per line. When you are done with the
list, type a <ctl D>.
next host to add:  circles
next host to add:  squares
next host to add:  triangles
next host to add:  <Ctrl-D>

The current list of NIS servers looks like this:

circles
squares
triangles
```

The NIS master server (*circles* for this example, shown in the prompt) is automatically added to the list of NIS servers. The NIS slave server names you specify must be the servers' official names, not their nicknames. The official names are the names returned by the *hostname*(1) command.

The names supplied here are put in the ASCII file */var/yp/ypservers*.

4. Confirm that list of NIS slave servers is correct:

```
Is this correct?  [y/n: y] y
```

If the output in instruction 3 shows any errors, enter **n** (no) and repeat that instruction.

After your response, *ypinit* reads */var/yp/make.script* to determine which maps to make. It takes a few minutes to build the maps. The amount of time required depends on the size of your input files. When the maps are complete, you see this output:

```
There will be no further questions. The remainder of the
procedure should take 5 to 10 minutes.
Building NIS databases:
> updated passwd (14:10:10 08/15/93)
> updated group (14:10:11 08/15/93)
> updated hosts (14:10:18 08/15/93)
> updated ethers (14:10:19 08/15/93)
> updated networks (14:10:21 08/15/93)
```

```
> updated rpc (14:10:21 08/15/93)
> updated services (14:10:22 08/15/93)
> updated protocols (14:10:23 08/15/93)
> updated netgroup (14:10:24 08/15/93)
3 aliases, longest 11 bytes, 44 bytes total
> updated aliases (14:10:26 08/15/93)
> updated bootparams (14:10:26 08/15/93)
> updated netid (14:10:29 08/15/93)
> updated ypservers (14:10:30 08/15/93)
```

When the NIS master server *circles* is set up without any errors, you see a message like this one:

```
circles has been set up as the NIS master server without
any errors.
```

Any other message suggests that the master server's maps may be corrupt. If there is any doubt about the integrity of the database maps, always go back to instruction 1 of this step to rebuild the maps from scratch.

5. Answer **y** (yes) to this prompt so the NIS master server daemons start when this master server is booted:

```
Start NIS master daemons during system startup? [y/n: y] y
```

This does a *chkconfig*(1M) on the appropriate NIS daemons. It does not start the daemons unless the system is rebooted.

6. Run *yppush*(1M) and *ypinit*, if needed:

```
If there are running slave NIS servers, run yppush now
for any data bases which have been changed. If there are
no running slaves, run ypinit on those hosts which are to
be slave servers.
```

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). If this setup is new, you must run *ypinit* on the selected systems to build the slave server databases.

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: rebooting the NIS master server, stopping and restarting the network with the */etc/init.d/network* script, or starting the daemons manually by giving these commands:

```
# /usr/etc/ypserv
# /usr/etc/ypbind
# /usr/etc/rpc.passwd /etc/passwd.nis -m passwd
```

To specify a DNS database lookup for host name resolution, add the **–i** option to the *ypserv*(1M) command (see "Using ypserv –i" in Chapter 3 for more information). Note that the *rpc.passwd*(1M) 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.

Testing the NIS Master Server

Finally, to ensure that NIS services are functioning properly on the NIS master server, give the *ypwhich*(1) 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.

```
# ypwhich
localhost
```

The response `localhost` indicates that *ypbind*(1M) is correctly bound to the NIS server on the local system. Instead of `localhost`, it may return its name as reported by *hostname*(1).

Setting Up NIS Slave Servers

There are five parts to the procedure for setting 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. Setting the slave server's domain name

2. Binding to an NIS server

3. Building the duplicate maps

4. Starting NIS on the slave server

5. Testing the NIS slave server

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.

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 on the same network as *circles*, start the binding process:

    # **/usr/etc/ypbind**

2. If the slave is not on the same network as *circles*, verify that the master server has an entry in */etc/hosts* on the slave server, and give the *ypset*(1M) command:

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

3. Verify that the server is bound by giving the *ypwhich* command:

    ```
    # 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 with the *ypwhich* command:

    ```
    # ypwhich -m
    ```

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

2. Start building NIS slave server maps with the *ypinit* command:

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

    The **–s** flag specifies that this system is to be an NIS slave server and *circles* is the master server.

    ```
    Installing the NIS data base will require that you answer
    a few questions. Questions will all be asked at the
    beginning of the procedure.
    ```

3. You do not want this procedure to quit on non-fatal errors, so answer **n** (no) to this prompt:

    ```
    Do you want this procedure to quit on non-fatal errors?
    [y/n: n] n
    OK, please remember to go back and redo manually whatever
    fails. If you don't, some part of the system (perhaps the
    NIS itself) won't work.

    There will be no further questions. The remainder of the
    procedure should take a few minutes, to copy the data
    bases from circles.
    ```

```
Transferring ypservers...
Transferred map ypservers from master (4 entries).
.
.
.
Transferring passwd.byuid...
Transferring map passwd.byuid from master (17 entries).
Transferring passwd.byname...
Transferred map passwd.byname from master (24 entries).
```

When the NIS slave server *squares* is set up without any errors, you see a message like this one:

```
squares's NIS database has been set up without any errors.
```

Any other message suggests that the slave server's maps may be corrupt. If there is any doubt about the integrity of the database maps, always go back to instruction 1 of this step to rebuild the maps from scratch.

4. Answer **y** (yes) to the prompt so that the NIS slave server daemons start when this slave server is booted:

```
Start NIS slave daemons during system startup? [y/n: y] y

At this point, make sure that /etc/passwd, /etc/hosts,
/etc/networks, /etc/group, /etc/protocols, /etc/services,
/etc/rpc and /etc/netgroup have been edited so that when
the NIS is activated, the data bases you have just
created will be used.
```

Answering yes does a *chkconfig* on the appropriate NIS daemons. It does not start the daemons unless the system is rebooted.

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: rebooting the NIS master server, stopping and restarting the network with the */etc/init.d/network* script, or start the daemons manually by giving these commands:

```
# /etc/killall ypbind
# /usr/etc/ypserv
# /usr/etc/ypbind
```

Testing the NIS Slave Server

Finally, to ensure that NIS services are functioning properly on the NIS slave server, give 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.

```
# ypwhich
localhost
```

The response, localhost, indicates that *ypbind* 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

There are four parts to the procedure for setting up the NIS client. Repeat these steps for each NIS client you need to set up:

1. Setting the domain

2. Configuring NIS on the client

3. Starting NIS on the client

4. Testing the NIS client

Setting the Domain

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

Configuring NIS on the Client

If the NIS service is to start automatically when this client is booted, the NIS environment must be configured with the *chkconfig* command. The *yp* flag allows this system to access NIS database information from an NIS server. This flag starts up the *ypbind* daemon, which must be running to access NIS database information. Give this command to set the flag on:

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

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 start the NIS daemon manually by giving this command:

```
# /usr/etc/ypbind
```

Testing the NIS Client

Finally, 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 *ypbind* 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.

Maintaining NIS

This chapter explains how to maintain NIS after it is in service. It contains procedures for adding new users to NIS, changing passwords, using netgroups, creating a nonstandard NIS map, modifying existing maps, new maps, adding an NIS slave server, changing the NIS master server, and increasing the security of *ypset*(1M).

This chapter contains these 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 a password entry for the new user to the NIS password file (*/etc/passwd* by default). (See *passwd*(4), the *Personal System Administration Guide*, or the *IRIX Advanced Site and Server Administration Guide* for more information.)

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 password entry as you added in instruction 1 in this section. Duplicating the entry enables the user to log in when the network is down.

   - Add this password entry:

     +userid

     userid is the login name of the new user. When this type of entry is used, all */etc/password* 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/password* information for this user is supplied by NIS.

   - Add this password entry:

     +

     When this type of entry is used, all */etc/password* 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.

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/password* entry duplicates the NIS password entry or if NIS maps have been propagated to this system (it takes about 15 minutes after instruction 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*(1):

```
% 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 instruction 4), have the user add his or her password to the local */etc/passwd* using *passwd*(1):

```
% passwd
```

passwd prompts for the password twice.

47

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.

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>
```

Enter your old NIS password (if the account is not password protected, press the enter key, **<Enter>**):

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

Enter your new NIS password:

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

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.

If your local password is not given explicitly but rather is pulled in from NIS with a plus (**+**) entry, then the *passwd* command prints the 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*(1M) 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*(1), *mountd*(1M), *rlogind*(1M), and *rshd*(1M) 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*(4) or *.rhosts*(4).

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.

Here is a sample */etc/netgroup* file. See *netgroup*(4) 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)
```

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 manual pages: *yppasswd*(1), *hosts.equiv*(4), *export*(4), *passwd*(4), *group*(4), *netgroup*(4), and *rpc.passwd*(1M).

Creating a Nonstandard NIS Map Manually

You can use *ypinit*(1M) and */var/yp/local.make.script* (see *ypmake*(1M)) 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 pair of well-formed *dbm* files 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
# makedbm /etc/yourmap shapes/yourmap
```

This command sequence creates a map called *yourmap* in the directory */var/yp/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/yp
# makedbm - shapes/yourmap
```

Enter these lines:

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

The *makedbm*(1M) – 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/yp/shapes*.

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/make.script* 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/make.script* 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 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 nonstandard databases. You can also manually change databases that are rarely expected to change and databases for which no ASCII form exists.

The general procedure is to use *makedbm* with a switch to disassemble the map. The disassembled map is in a form you can modify with standard tools such as *awk*(1), *sed*(1), or *vi*(1). You then build a new map from the changed version using *makedbm*.

Use this procedure to modify a nonstandard database:

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

```
# cd /var/yp
# makedbm -u shapes/mymap > /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:

```
# makedbm /var/tmp/mymap.txt shapes/mymap
```

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

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

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

Preparing to Propagate Nonstandard Maps

Preparing for propagating a nonstandard NIS map consists of setting up its *dbm* 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.indirect*, *auto.direct*, *auto.master*, and *auto.home* from the files */etc/auto.indirect*, */etc/auto.direct*, */etc/auto.master*, and */etc/auto.home*.

```
#
#        auto.indirect            indirect automount YP map.
#        auto.direct              direct automount YP map.
#        auto.master              main auto.master automount map
#        auto.home                homedir map for automounter

INDIRECT=auto.indirect
DIRECT=auto.direct
HOME=auto.home

localall: all $(INDIRECT) $(DIRECT) $(HOME) auto.master

auto.master:    auto.master.time
$(DIRECT):      direct.time
$(HOME):        home.time
$(INDIRECT):    indirect.time

indirect.time:  $(DIR)/$(INDIRECT)
        -@if [ -f $(DIR)/$(INDIRECT) ]; then \
                sed -e '/^#/d' $(DIR)/$(INDIRECT) | \
                $(MAKEDBM) - $(YPDBDIR)/$(INDIRECT); \
                touch indirect.time; \
                echo "Updated $(INDIRECT)"; \
                if [ ! $(NOPUSH) ]; then \
                        $(YPPUSH) $(INDIRECT); \
                        echo "pushed $(INDIRECT)"; \
                else \
                        : ; \
                fi \
        else \
                echo "couldn't find $(DIR)/$(INDIRECT)";\
```

```
                            fi

direct.time:  $(DIR)/$(DIRECT)
        -@if [ -f $(DIR)/$(DIRECT) ]; then \
                sed -e '/^#/d' $(DIR)/$(DIRECT) | \
                $(MAKEDBM) - $(YPDBDIR)/$(DIRECT); \
                touch indirect.time; \
                echo "Updated $(DIRECT)"; \
                if [ ! $(NOPUSH) ]; then \
                        $(YPPUSH) $(DIRECT); \
                        echo "pushed $(DIRECT)"; \
                else \
                        : ; \
                fi \
        else \
                echo "couldn't find $(DIR)/$(DIRECT)";\
        fi

auto.master.time:       $(DIR)/auto.master
        -@if [ -f $(DIR)/auto.master ]; then \
                sed -e '/^#/d' $(DIR)/auto.master | \
                $(MAKEDBM) - $(YPDBDIR)/auto.master; \
                touch auto.master.time; \
                echo "Updated auto.master"; \
                if [ ! $(NOPUSH) ]; then \
                        $(YPPUSH) auto.master; \
                        echo "pushed auto.master"; \
                else \
                        : ; \
                fi \
        else \
                echo "couldn't find $(DIR)/auto.master";\
        fi

home.time:      $(DIR)/$(HOME)
        -@if [ -f $(DIR)/$(HOME) ]; then \
                sed -e '/^#/d' $(DIR)/$(HOME) | \
                $(MAKEDBM) - $(YPDBDIR)/$(HOME); \
                touch home.time; \
                echo "Updated $(HOME)"; \
                if [ ! $(NOPUSH) ]; then \
                        $(YPPUSH) $(HOME); \
                        echo "pushed $(HOME)"; \
                else \
                        : ; \
```

```
                        fi \
        else \
                echo "couldn't find $(DIR)/$(HOME)";\
        fi
```

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*(1)
to make two databases suitable for input to *makedbm*. For example, the
databases might be stored as */var/yp/circles/auto.master.pag* and
/var/yp/circles/auto.master.dir.

To create a customized make script, */var/yp/local.make.script*, use the existing
/var/yp/make.script as a source of programming examples. Make use of
mechanisms already in place in */var/yp/make.script* when deciding how to
create dependencies that *make*(1) 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*(1M) 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.

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*(1M) or
by executing *ypmake*, *ypxfr*, or *yppush*(1M) 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*(1M) 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
  ypxfr's **–h** option)

For more information on how to use *crontab*, see *crontab*(1).

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/make.script* and/or */var/yp/local.make.script* to determine which maps to make. The make script calls *makedbm*, which updates the maps and calls *yppush*. *yppush* 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 with *ypxfr*.

Use *ypmake* to update and propagate maps throughout your domain when you want the change to take place immediately and don't 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 -u passwd**

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

  # **/var/yp/ypmake -u 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.

ypxfr, as a command, has options that force map transfer and specify alternate domains and servers from which to obtain the map. Some examples of *ypxfr* command usage are:

- 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*

While *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:

- 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 *ypxfr*'s transfer attempts 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 */usr/etc/yp/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" in Chapter 4. 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" in Chapter 4 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 don't push them to other servers:

   newmaster# **/var/yp/ypmake -u 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 Table 2-2:

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

newmaster is the host name of the new master server and *mapname* is a map name from Table 2-2. *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 won't 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 Table 2-2:

```
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, the file */etc/config/ypbind.options* contains the **–ypsetme** option that enables *ypset*. Normally, the **–ypsetme** option should be present when creating an NIS master since, if 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.

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" on page 63
- "Debugging an NIS Client" on page 68
- "Before You Call for Help" on page 74

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*(1M) 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*(1M) 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 following section, "Intermittent, Consistent Map Propagation Failures."

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:

   ```
   # ypmatch -x
   ```

   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 -u 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 *ypserv* with the *chkconfig*(1M) command.

Finally, if the above suggestions don't 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. Give 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*(1), 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*, give this command:

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

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

ypserv Fails

If *ypserv*(1M) fails almost immediately each time it is started, look for a more general networking problem. Since NIS uses RPC, the portmapper must be functioning correctly for NIS to work.

To verify that the portmapper is functioning and that the *ypserv* process is registered with the portmapper, on the server give this command:

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

If your *portmap*(1M) daemon is functional, the output looks something like this:

```
100004    2    udp    1051    ypserv
100004    2    tcp    1027    ypserv
100004    1    udp    1051    ypserv
100004    1    tcp    1027    ypserv
```

If these entries are not in your output, *ypserv* 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*(1M) 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*, *ypbind*(1M), and *ypserv*, are started in the correct order.

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 console messages such as:

```
NIS v.1 server not responding for domain domain_name; still
trying
```

```
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 *ypbind* on the local system is unable to communicate with *ypserv* 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 *ypserv* cannot get a response back to the *ypbind* daemon within the time-out period.

- *ypserv* 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 questions help to isolate and correct the situation.

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 more details. Also check to ensure that the client is attached to the correct physical network.

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*(1) 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 *ypbind* so that the client is bound within the correct domain. Give these commands:

   # **/etc/killall ypbind**
   # **/usr/etc/ypbind**

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 *ypbind* 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.

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 *ypserv* process has crashed on the NIS server.

If an NIS server crashes or becomes unavailable, it should not impact 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 *ypserv* process has crashed on the server. Give these commands to find out if *ypserv* is running and restart it if it is not:

1. Log into the NIS server system. Look for *ypserv* and *ypbind* processes. Give this command:

   # **ps -ef | grep yp**

   You should see output similar to this:

   ```
   root    128     1  0  Sep 13  ?          1:35 /usr/etc/ypbind
   root    127     1  7  Sep 13  ?         61:16 /usr/etc/ypserv
   ```

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

   # **/usr/etc/ypbind**

3. Give the command *ypwhich*(1) on the NIS server system:

   # **ypwhich**

   If *ypwhich* returns no answer, *ypserv* is probably not working.

4. If *ypserv* is not working, give these commands to kill the existing *ypserv* process and start a new one:

   # **/etc/killall -v ypserv**
   # **/usr/etc/ypserv**

71

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, *ypbind*. These examples illustrate typical error messages you might see when you give an NIS command and *ypbind* has failed:

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

In addition to the error messages listed above, these general symptoms may also indicate that the *ypbind* 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, the *ypbind* process on the client must be stopped and restarted. Use this command sequence to stop and start *ypbind*:

```
# /etc/killall ypbind
# /usr/etc/ypbind
```

Give this command to verify that the *ypbind* process is running:

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

You should see output similar to this:

```
root 26995     1  0 17:35:31 ?          0:00 /usr/etc/ypbind -ypsetme
root 27007 26952  0 17:45:45 ttyq3      0:00 grep ypbind
```

ypbind Fails

If *ypbind* fails almost immediately each time it is started, look for a more general networking problem. Because NIS uses RPC, the portmapper must be functioning correctly for NIS to work.

To verify that the portmapper is functioning and that the *ypbind* process is registered with the portmapper, on the client give this command:

/usr/etc/rpcinfo -p | grep ypbind

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

```
100007   2   tcp   1026   ypbind
100007   2   udp   1046   ypbind
100007   1   tcp   1026   ypbind
100007   1   udp   046    ypbind
```

If these entries are not there, *ypbind* has been unable to register its services with the *portmap* daemon. If the *portmap* daemon has crashed or it 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 client. Rebooting the client ensures that the network daemons, specifically *portmap* and *ypbind*, are started in the correct order.

For more information about *ypbind*, see "Using Secure ypset" in Chapter 5.

ypwhich Output Inconsistent

When you give 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 eoe1 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

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document

- Omission of material that you expected to find

- Technical errors

- Relevance of the material to the job you had to do

- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-2161-001.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:

    - On the Internet: techpubs@sgi.com

    - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs

- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801

- To send your comments by **traditional mail**, use this address:

Technical Publications
Silicon Graphics, Inc.
2011 North Shoreline Boulevard, M/S 535
Mountain View, California 94043-1389