Solaris Naming Setup and Configuration Guide

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

Part Number 806–1386–10
February 2000

Contents

Preface

Solaris Naming Setup and Configuration Guide describes how to set up and configure the four Solaris™ name services: NIS+, NIS, FNS, LDAP, and DNS. This manual is part of the Solaris 8 release System and Network Administration manual set.

Who Should Use This Book

This book is for system and network administrators who want to set up one or more of the four Solaris name services. It assumes you are an experienced system administrator.

Although this book introduces networking concepts relevant to Solaris name services, it makes no attempt to explain networking fundamentals or describe the administration tools offered by the Solaris environment. If you administer networks, this manual assumes you already know how they work and have already chosen your favorite tools.

Solaris Naming Administration Guide contains a thorough description of the Solaris name services, a glossary of name service terms, and a listing of common error messages.

How This Book Is Organized

This book has five parts:

Part 1, Naming Service Setup

This part describes how to use the `nsswitch.conf` file to specify how different name services work together.

- Chapter 1 describes the name service switch and provides step-by-step instructions for configuring it.

Part 2, NIS+ Setup and Configuration

This part describes how to set up and configure an NIS+ namespace. The first chapter gives an introduction to NIS+. The second and third chapters describe how to set up an NIS+ namespace using the NIS+ setup scripts. The setup scripts are the preferred method for NIS+ setup and configuration. The last five chapters describe how to set up and configure an NIS+ namespace using the NIS+ command set.

- Chapter 2 provides a brief overview of *Network Information Service Plus* (NIS+), lists the tasks you need to perform before setting up NIS+, identifies the minimum requirements of an NIS+ namespace, and describes the two methods of NIS+ setup.
- Chapter 3 describes the NIS+ scripts and what they can and cannot do.
- Chapter 4 describes how to configure a basic NIS+ namespace using the `nisserver`, `nispopulate`, and `nisclient` scripts in combination with a few NIS+ commands.
- Chapter 5 provides step-by-step instructions for setting up the root domain and DES authentication using the NIS+ command set.
- Chapter 6 provides step-by-step instructions for setting up NIS+ clients using the NIS+ command set and three different initialization methods. These instructions apply to clients in both the root domain and subdomains, whether all-NIS+ or NIS-compatible.
- Chapter 7 provides step-by-step procedures for using the NIS+ command set to set up NIS+ servers (except the root master) and add replica servers to existing NIS+ domains.
- Chapter 8 provides step-by-step instructions for using the NIS+ command set to configure a subdomain domain (also known as a nonroot domain) including designating its master and replica servers.
- Chapter 9 provides step-by-step instructions for using the NIS+ command set to populate NIS+ tables on a master server from `/etc` files or NIS maps, how to transfer information back from NIS+ tables to NIS maps, and how to limit access to the passwd column of the `passwd` table.

Part 3, NIS Setup and Configuration

Chapter 10 describes initial setup and configuration of the Network Information Service (NIS).

Part 4, FNS Setup and Configuration

Chapter 11 describes how to initially set up and configure the Federated Naming Service (FNS) in an NIS+, NIS, or `/etc` namespace environment.

Part 5, DNS Setup and Configuration

This part describes how to set up DNS clients and servers.

- Chapter 12 describes how to set up DNS service on client machines.
- Chapter 13 describes how to set up a DNS name server.

Related Books

You can consult the following books for more information on NIS+ and DNS. These books are also part of the Solaris 8 release System and Network Administration manual set:

- *Solaris Naming Administration Guide*—describes how to customize and administer an existing NIS+ namespace.
- *NIS+ Transition Guide*—Describes how to make the transition from NIS to NIS+.

Additional books not part of the Solaris 8 release manual set:

- Cricket Lui and Paul Albitz, *DNS and Bind* (O'Reilly, 1992).
- Stern, Hal, *Managing NFS and NIS* (O'Reilly 1991).

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

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

What Typographic Conventions Mean

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

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*.
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

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

TABLE P–2 Shell Prompts

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

PART I Naming Service Setup

This part describes how to use the `nsswitch.conf` file to specify how different name services work together.

- Chapter 1, “Setting Up the Name Service Switch”

CHAPTER **1**

Setting Up the Name Service Switch

This section describes the name service switch and provides step-by-step instructions for configuring it.

Name Service Switch

The name service switch controls how a client workstation or application obtains network information. The name service switch is often referred to as *the switch*. The switch determines which naming services, and in what order, an application uses to obtain naming information. The switch is a file called `nsswitch.conf`, which is stored in each machine's `/etc` directory.

The `nsswitch.conf` File

Each workstation has a `nsswitch.conf` file in its `/etc` directory. Each line of that file identifies a particular type of network information, such as host, password, and group, followed by one or more sources, such as NIS+ tables, NIS maps, the DNS hosts table, or local `/etc`, where the client is to look for that information. For additional information on the `nsswitch.conf` file, see *Solaris Naming Administration Guide*.

An `/etc/nsswitch.conf` file is automatically loaded into every workstation's `/etc` directory by the Solaris 8 release software, along with the following alternate (template) versions:

- `/etc/nsswitch.nisplus`

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

These alternate template files contain the default switch configurations used by the NIS+ and NIS services, local files, and LDAP. No default file is provided for DNS, but you can edit any of these files to use DNS (see “Enabling a Machine to Use DNS” on page 27). When the Solaris operating environment is first installed on a workstation, the installer selects the workstation’s default name service: NIS+, NIS, local files, or LDAP. During installation, the corresponding template file is copied to `/etc/nsswitch.conf`. For example, for a workstation client using NIS+, the installation process copies `nsswitch.nisplus` to `nsswitch.conf`.

If your network is connected to the Internet and you want users to be able to access Internet hosts using DNS, you must now enable DNS forwarding, as described in “Enabling a Machine to Use DNS” on page 27.

Unless you have an unusual namespace, the default template file as copied to `nsswitch.conf` (with or without DNS, as described above) should be sufficient for normal operation.

Default NIS+ Version of Switch File

The NIS+ version of the switch file supplied with Solaris 7 release is named `nsswitch.nisplus`.

CODE EXAMPLE 1–1 Default `nsswitch.nisplus` File

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

# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
passwd: files nisplus
group: files nisplus
# consult /etc "files" only if nisplus is down.
hosts: nisplus [NOTFOUND=return] files
# Uncomment the following line, and comment out the above, to use
# both DNS and NIS+. You must also set up the /etc/resolv.conf
# file for DNS name server lookup. See resolv.conf(4).
# hosts: nisplus dns [NOTFOUND=return] files
services: nisplus [NOTFOUND=return] files
```

(continued)

(Continuation)

```
networks: nisplus [NOTFOUND=return] files
protocols: nisplus [NOTFOUND=return] files
rpc: nisplus [NOTFOUND=return] files
ethers: nisplus [NOTFOUND=return] files
netmasks: nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files
publickey: nisplus
netgroup: nisplus
automount: files nisplus
aliases: files nisplus
sendmailvars: files nisplus
```

Default NIS Version of Switch File

The NIS version of the switch file supplied with Solaris 7 release is named `nsswitch.nis`.

CODE EXAMPLE 1–2 Default `nsswitch.nis` File

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

ethers: nis [NOTFOUND=return] files
netmasks: nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey: nis [NOTFOUND=return] files
netgroup: nis
automount: files nis
aliases: files nis
# for efficient getservbyname() avoid nis
services: files nis
sendmailvars: files
```

Default Files Version of Switch File

The local files version of the switch file supplied with Solaris 7 release is named `nsswitch.files`.

CODE EXAMPLE 1–3 Default `nsswitch.files` File

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

netgroup: files
automount: files
aliases: files
services: files
sendmailvars: files
```

Default LDAP Version of Switch File

The LDAP version of the switch file supplied with Solaris operating environment is named `nsswitch.ldap`.

CODE EXAMPLE 1–4 LDAP Switch File Template

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

(continued)

(Continuation)

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

hosts:      ldap [NOTFOUND=return] files

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

netgroup:   ldap

automount:  files ldap
aliases:    files ldap

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

Selecting a Different Configuration File

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

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

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

Security Considerations

You must perform this operation as superuser.

Setting Up the Name Service Switch

TABLE 1–1 Task Map: Setting Up the Name Service Switch

Task	Description	For Instructions, Go To
Setting Up the Name Service Switch	Set up a configuration file for NIS+ or NIS. and reboot the workstation	"How to Select a Different Configuration File" on page 26

▼ How to Select a Different Configuration File

To change to a switch file, follow these steps:

1. **Log in to the client as superuser.**

2. **Copy the alternate file appropriate for the machine's name service over the** `nsswitch.conf` **file.**

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

```
client1# cd /etc
client1# cp nsswitch.nisplus nsswitch.conf
```

   NIS Version

```
client1# cd /etc
client1# cp nsswitch.nis nsswitch.conf
```

   Local `/etc` *Files Version*

```
client1# cd /etc
client1# cp nsswitch.files nsswitch.conf
```

3. **Reboot the workstation.**

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

Enabling a Machine to Use DNS

This section describes how to set up the name service switch configuration file for the NIS+ or local files name services so that a machine can also use the Domain Name System (DNS). DNS forwarding is inherent in the NIS name service. You do not have to (and should not) add a DNS entry to the hosts line of switch file of a machine using the NIS service. The steps described below apply *only* to those machines using local /etc files or NIS+.

Prerequisites

The machine must have a properly configured /etc/resolv.conf file (as described in "The Resolver" on page 217).

Security Considerations

You must perform this operation as superuser.

Enabling a Machine to Use DNS-Task Map

TABLE 1–2 Enabling a Machine to Use DNS

Task	Description	For Instructions, Go To
Enabling a Machine to Use DNS	Modify the `/etc/nsswitch.conf` file and enable an NIS+ client to use DNS	"How to Enable an NIS+ Client to Use DNS" on page 28

▼ How to Enable an NIS+ Client to Use DNS

1. **Log in as superuser.**

2. **Open the `/etc/nsswitch.conf` file.**

3. **Specify DNS as a source of hosts information.**

   DNS can be the *only* source or an *additional* source for the hosts information. Locate the `hosts` line and use `DNS` in one of the ways shown below:

   ```
   hosts: files dns
   ```

   or

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

   or

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

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

4. **Save the file and reboot the workstation.**

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

Adding Compatibility With +/- Syntax

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

Adding Compatibility With +/- Syntax-Task Map

TABLE 1–3 Adding Compatibility With +/- Syntax

Task	Description	For Instructions, Go To
Adding Compatibility With +/- Syntax	Modify the `/etc/passwd`, `/etc/shadow`, and `/etc/group` files to add DNS compatibility with +/- syntax.	"How to Add DNS Compatibility With +/- Syntax" on page 29

Security Considerations

You must perform this operation as superuser.

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

▼ How to Add DNS Compatibility With +/- Syntax

1. **Log in as superuser.**

2. **Open the** `/etc/nsswitch.conf` **file.**

3. **Change the passwd and groups sources to** `compat`**.**
   - For use with NIS, enter:

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

- For NIS+, enter:

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

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

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

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

---

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

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

Enabling a Machine to Use IPv6 Addresses

The `nsswitch.conf` file controls search criteria for IPv6 addresses. IPv6 increases the IP address size from 32 bits to 128 bits to support more levels of addressing hierarchy and provide a greater number of addressable nodes. For more information about IPv6, its configuration and implementation see “Overview of IPv6” in *System Administration Guide, Volume 3* and “Transitioning From IPv4 to IPv6” in *System Administration Guide, Volume 3*.

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

Enabling a Machine to Use IPv6-Task Map

TABLE 1–4 Enabling a Machine to Use IPv6

Task	Description	For Instructions, Go To
Enabling a Machine to Use IPv6	Modify the `/etc/nsswitch.conf` file and enable an NIS+ client to use IPv6	"How to Enable an NIS+ Client to Use IPv6" on page 31

▼ How to Enable an NIS+ Client to Use IPv6

1. **Log in as superuser.**

2. **Edit the** `/etc/nsswitch.conf` **file.**

3. **Add the new** `ipnodes` **source and specify the name service (such as ldap).**

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

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

4. **Save the file and reboot the workstation.**

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

PART II NIS+ Setup and Configuration

This part describes how to set up and configure an NIS+ namespace:

- The first chapter gives an introduction to NIS+.
- The second and third chapters describe how to set up an NIS+ namespace using the NIS+ setup scripts. The setup scripts are the preferred method for NIS+ setup and configuration.
- The last five chapters describe how to set up and configure an NIS+ namespace using the NIS+ command set.

This part has eight chapters:

CHAPTER 2

Getting Started With NIS+

This chapter provides a brief overview of *Network Information Service Plus* (NIS+), lists the tasks you need to perform before setting up NIS+, identifies the minimum requirements of an NIS+ namespace, then describes the two methods of NIS+ setup.

- "NIS+ Overview" on page 35
- "Setup and Configuration Preparation" on page 36
- "Preparing the Existing Namespace" on page 37
- "Two Configuration Methods" on page 38

NIS+ Overview

NIS+ is a network name service similar to NIS but with more features. NIS+ is not an extension of NIS; it is a new software program.

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

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

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

NIS+ includes a sophisticated security system to protect the structure of the namespace and its information. It uses authentication and authorization to verify whether a client's request for information should be fulfilled. *Authentication* determines whether the information requestor is a valid user on the network. *Authorization* determines whether a particular user is allowed to have or modify the information requested.

Solaris clients use the name service switch (the `/etc/nsswitch.conf` file) to determine from where a workstation will retrieve network information. Such information may be stored in local `/etc` files, NIS, DNS, or NIS+. You can specify different sources for different types of information in the name service switch.

For a more thorough description of NIS+, see the *Solaris Naming Administration Guide*.

Setup and Configuration Preparation

Before configuring your NIS+ namespace, you must:

- Install properly configured `nsswitch.conf` files on all the machines that use NIS+. See Chapter 1 for details.
- Plan your NIS+ layout. This includes:
  - Planning your namespace. What will your domain name be? Will you have subdomains, and if so how will they be organized? Which machines will be in which domain? Will your domain be connected to a higher domain or to the Internet?
  - Determining your server requirements. How many replica servers will be needed for each domain? What type of server, processor speed, and memory is required? How much server disk space is needed?

    See the *NIS+ Transition Guide* for a detailed description of these and other planning issues, and recommended guidelines.
- Prepare your existing namespace (if any). See "Preparing the Existing Namespace" on page 37.
- Choose a root server machine.

- Make sure that you have at least one system already running at your site that can be used as your root master server. This machine must contain at least one user (root) in the system information files, such as `/etc/passwd.` (Machines usually come with root in the system files, so this should not be a problem.)

Preparing the Existing Namespace

If an NIS domain already exists at your site, you can use the same flat domain structure for your NIS+ namespace. (You can change it later to a hierarchical structure.) Read the *NIS+ Transition Guide* before you start your transition from NIS to NIS+ for important planning and preparation information. The NIS+ scripts enable you to start NIS+ with data from NIS maps. Chapter 4 shows you how to use the NIS+ scripts to create a NIS+ namespace from either system files or NIS maps.

In order for the scripts to run smoothly, however, you must prepare your existing namespace (if you have one) for conversion to NIS+. These preparations are described fully in *NIS+ Transition Guide*.

For your reference, key preparations are summarized below:

- *Domain and host names*. Domains and hosts must not have the same name. For example, if you have a `sales` domain you cannot have a machine named `sales`. Similarly, if you have a machine named `home`, do not create a domain named `home`. This caution also applies to subdomains; for example, if you have a machine named `west,` you don't want to create a `sales.west.myco.com` subdirectory.
- *No dots in host names*. Because NIS+ uses dots (periods) to delimit between machine names and domains and between parent and subdomains, you cannot have a machine name containing a dot. Before converting to NIS+ (before running the scripts) you must eliminate any dots in your host names. You should convert host name dots to hyphens. For example, you cannot have a machine named `sales.alpha`. You can convert that name to `sales-alpha`.
- *Root server must be running*. The machine that is designated to be the root server must be up and running and you must have superuser access to it.
- View any existing local `/etc` files or NIS maps that you will load data from. Make sure that there are no spurious or incorrect entries. Make sure that the right data is in the correct place and format. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. You can always add individual entries after configuration is completed. That is easier than trying to load incomplete or damaged entries.

Caution - In Solaris 2.4 and earlier, the `/var/nis` directory contained two files named *hostname*`.dict` and *hostname*`.log`. It also contained a subdirectory named `/var/nis/`*hostname*. When you install NIS+ for Solaris 2.5, the two files are named `trans.log` and `data.dict`, and the subdirectory is named `/var/nis/data`. In Solaris 2.5, the *content* of the files has also been changed and they are not backward compatible with Solaris 2.4 or earlier. Thus, if you rename either the directories or the files to match the Solaris 2.4 patterns, the files will not work with either the Solaris 2.4 or the Solaris 2.5 version of `rpc.nisd`. Therefore, you should rename neither the directories nor the files.

Two Configuration Methods

The rest of this part of the manual describes two different methods of configuring an NIS+ namespace:

- *With the setup (configuration) scripts.* Chapters 2 and 3 describe how to configure NIS+ using the three NIS+ scripts: `nisserver`, `nispopulate`, and `nisclient`. This is the easiest, as well as recommended, method.
- *With the NIS+ command set.* Chapters 4 through 9 describe how to configure NIS+ using the NIS+ command set. While this method gives you more flexibility than the scripts method, it is more difficult. This method should be used only by experienced NIS+ administrators who need to configure a namespace with characteristics significantly different than those provided by the configuration scripts.

Note - If you use the NIS+ command set, you must also make sure that each machine using NIS+ for its name service has the correct `nsswitch.conf` file in its `/etc` directory as described in Chapter 1. If you use the NIS+ configuration scripts on a given machine, this step is performed for you.

See *Solaris Naming Administration Guide* for information on how to remove an NIS+ directory or domain, an NIS+ server, or the NIS+ namespace.

CHAPTER 3

NIS+ Setup Scripts—Introduction

This chapter describes the NIS+ scripts and what they will and will not do.

- "About the NIS+ Scripts" on page 39
- "What the NIS+ Scripts Will Not Do" on page 40

About the NIS+ Scripts

Caution - Before running the NIS+ setup scripts, make sure you have performed the steps described in "Setup and Configuration Preparation" on page 36.

The three NIS+ scripts—`nisserver`, `nispopulate`, and `nisclient`—enable you to set up a NIS+ namespace. The NIS+ scripts are Bourne shell scripts that execute groups of NIS+ commands so you do not have to type the NIS+ commands individually. Table 3–1 describes what each script does.

TABLE 3–1 NIS+ Scripts

NIS+ Script	What It Does
`nisserver`	Sets up the root master, non-root master and replica servers with level 2 security (DES)
`nispopulate`	Populates NIS+ tables in a specified domain from their corresponding system files or NIS maps
`nisclient`	Creates NIS+ credentials for hosts and users; initializes NIS+ hosts and users

What the NIS+ Scripts Will Do

In combination with a few NIS+ commands, you can use the NIS+ scripts to perform all the tasks necessary for setting up an NIS+ namespace. See the `nisserver`, `nispopulate`, and `nisclient` man pages for complete descriptions of these commands and their options. Chapter 4 shows you how to use the NIS+ scripts to set up an NIS+ namespace.

You can run each of the scripts without having the commands execute by using the `–x` option. This option lets you see what commands the scripts call and their approximate output without the scripts actually changing anything on your systems. Running the scripts with `–x` can minimize unexpected surprises.

What the NIS+ Scripts Will Not Do

While the NIS+ scripts reduce the effort required to create an NIS+ namespace, the scripts do not completely replace the individual NIS+ commands. The scripts only implement a subset of NIS+ features. If you are unfamiliar with NIS+, you may want to refer back to this section after you have created the sample NIS+ namespace.

The `nisserver` script only sets up an NIS+ server with the standard default tables and permissions (authorizations). This script does *not*:

- Set special permissions for tables and directories
- Add extra NIS+ principals to the NIS+ admin group

See Chapter 4 for how to use the `nisgrpadm` command instead of one of the NIS+ scripts to add extra NIS+ principals to the NIS+ admin group.

- Create private tables
- Run an NIS+ server at any security level other than level 2
- Start the `rpc.nisd` daemon on remote servers, which is required to complete server installation

See Chapter 4 for how to use the `rpc.nisd` command instead of one of the NIS+ scripts to change NIS+ client machines into non-root servers.

The `nisclient` script does not set up an NIS+ client to resolve host names using DNS. You need to explicitly set DNS for clients that require this option.

CHAPTER 4

Configuring NIS+ With Scripts

This chapter describes how to configure a basic NIS+ namespace using the `nisserver`, `nispopulate`, and `nisclient` scripts in combination with a few NIS+ commands.

This chapter also describes the following procedures:

- "Configuring a Client as an NIS+ Server" on page 69
- "How to Create a Root Replica" on page 71
- "How to Create a New Non-Root Domain" on page 76
- "Populating the Master Server Tables" on page 80
- "How to Create a Replica" on page 82
- "How to Initialize a New Subdomain Client Machine" on page 83
- "How to Initialize an NIS+ Subdomain User" on page 84

NIS+ Configuration Overview

Using the configuration scripts is the recommended method of setting up and configuring an NIS+ namespace. Using these scripts is easier than to trying to set up an NIS+ namespace with the NIS+ command set, as described in the subsequent chapters of this Part.

(See the `nisserver`, `nispopulate`, and `nisclient` man pages for complete descriptions of the scripts. See the glossary in *Solaris Naming Administration Guide* for definitions of terms and acronyms you do not recognize.)

You should *not* use the small sample NIS+ namespace referred to in this tutorial manual as a basis for your actual NIS+ namespace. You should destroy the sample namespace after you finish exploring it, instead of adding on to it. It is better to begin again and carefully plan your NIS+ hierarchy before you create your actual namespace.

Table 4–1 summarizes the recommended generic configuration procedure. The left column lists the major configuration activities, such as configuring the root domain or creating a client. The text in the middle describes the activities. The third column lists which script or NIS+ commands accomplish each step.

TABLE 4–1 Recommended NIS+ Configuration Procedure Overview

Activity	Description	Script/NIS+ Commands
Plan your new NIS+ namespace	Plan your new NIS+ namespace. See *NIS+ Transition Guide* for a full discussion of planning requirements and steps. (If you are just following the NIS+ tutorial in a test-bed network, this step has been done for you.)	
Prepare your existing namespace	In order for the scripts to work best, your current namespace (if any) must be properly prepared. See "Preparing the Existing Namespace" on page 37 and the *NIS+ Transition Guide* for a description of necessary preparations. (If you are just following the NIS+ tutorial in a test-bed network, this step has been done for you.)	
Configure the Diffie-Hellman key length	If you intend to use DES authentication, consider using Diffie-Hellman keys longer than the 192-bit default. The extended key length must be the same on all machines in the domain. Specify the desired key length before running the respective initialization scripts.	`nisauthconf`
Configure root Domain	Create the root domain. Configure and initialize the root master server. Create the root domain admin group.	`nisserver`

Populate tables	Populate the NIS+ tables of the root domain from text files or NIS maps. Create credentials for root domain clients. Create administrator credentials.	`nispopulate`
		`nisgrpadm`
		`nisping`

Configure root domain clients	Configure the client machines. (Some of them will subsequently be converted into servers.) Initialize users as NIS+ clients.	`nisclient`
Enable servers	Enable some clients of the root domain to become servers. Some servers will later become root replicas; others will support lower-level domains.	`rpc.nisd`
Configure root replicas	Designate one or more of the servers you just configured as replicas of the root domain.	`rpc.nisd` `nisserver`
Configure non-root domains	Create a new domain. Designate a previously enabled server as its master. Create its admin group and admin credentials.	`rpc.nisd` `nisserver`

TABLE 4–1 Recommended NIS+ Configuration Procedure Overview *(continued)*

Activity	Description	Script/NIS+ Commands
Populate tables	Create credentials for clients of the new domain. Populate the NIS+ tables of the new domain from text files or NIS maps.	`nispopulate`
Configure non-root domain clients	Configure the clients of the new domain. (Some may subsequently be converted into servers for lower-level domains.) Initialize users as NIS+ clients.	`nisclient`

The NIS+ scripts enable to you to skip most of the individual procedures included in the above activities.

Creating a Sample NIS+ Namespace

The procedures in this chapter show you how to create a sample NIS+ namespace. The sample NIS+ namespace will be created from `/etc` files and NIS maps. This sample shows you how to use the scripts both when your site is not running NIS and when NIS is running at your site. You can set your servers to NIS-compatibility mode if they will be serving NIS clients. See the *NIS+ Transition Guide* and the *Solaris Naming Administration Guide* for more information on NIS-compatibility mode.

Note - Your site's actual NIS+ namespace and its domain hierarchy probably differs from the sample namespace's, and yours probably contains a different *number* of servers, clients, and domains. Do not expect any resemblance between your final domain configuration or hierarchy and the sample one. The sample namespace is only an illustration of how to use the NIS+ scripts. After you have created this sample namespace, you should have a clear idea about how to create domains, servers, and clients at your site.

The sample namespace contains the following components:

- A root master server named `master` for the `doc.com.` domain
- Four clients of the root domain, `doc.com.`:
  - The first client, `client1`, will become a root replica (for the `doc.com.` domain).

- The second client, `client2`, will become a master server for a new subdomain (for the `sub.doc.com.` domain).

- The third client, `client3`, will become a non-root replica server of the new subdomain (for the `sub.doc.com.`) domain.

- The fourth client, `client4`, will remain solely a client of the root domain (`doc.com.`).

- Two clients, `subclient1` and `subclient2`, of the subdomain (`sub.doc.com.`).

This scenario shows the scripts being used to configure NIS+ at a site that uses both system information files, such as `/etc/hosts`, and NIS maps to store network service information. The sample NIS+ namespace uses such a mixed site purely for example purposes.

Summary of NIS+ Scripts Command Lines

Table 4–2 contains the generic sequence of NIS+ scripts and commands you will use to create a ample NIS+ domain. Subsequent sections describe these command lines in detail. After you are familiar with the tasks required to create NIS+ domains, servers, and clients, use Table 4–2 as a quick-reference guide to the appropriate command lines. Table 4–2 is a summary of the actual commands with the appropriate variables that you type to create the sample NIS+ namespace.

TABLE 4–2 NIS+ Domains Configuration Command Lines Summary

Action	Machine	Command

Include `/usr/lib/nis` in root’s path; C shell or Bourne shell.	Root master server and client machines as superuser	`setenv PATH $PATH:/usr/lib/nis`
		or
		`PATH=$PATH:/usr/lib/nis; export PATH`

Optionally, if using DES authentication, select the Diffie-Hellman key length	Server and client machines as superuser	`nisauthconf –dh`*key-length-alg-type* `des`
Create a root master server without or with NIS (YP) compatibility.	Root master server as superuser	`nisserver –r–d`*newdomain.* or `nisserver –Y–r–d` *newdomain.*

TABLE 4–2 NIS+ Domains Configuration Command Lines Summary *(continued)*

Action	Machine	Command
Populate the root master server tables from files or from NIS maps.	Root master server as superuser	`nispopulate –F–p` */files* `–d` *newdomain.* or `nispopulate –Y–d` *newdomain.* `–h` *NISservername\* `–a` *NIS_server_ipaddress* `–y` *NIS_domain*
Add additional users to the NIS+ admin group.	Root master server as superuser	`nisgrpadm–aadmin`*.domain.name.domain.*
Make a checkpoint of the NIS+ database.	Root master server as superuser	`nisping– C` *domain.*
Initialize a new client machine.	Client machine as superuser	`nisclient– i–d` *domain* . `–h` *master1*
Initialize user as an NIS+ client.	Client machine as user	`nisclient–u`

Start the `rpc.nisd` daemon—required to convert a client to a server without or with NIS (and DNS) compatibility.	Client machine as superuser	`rpc.nisd`
		or
		`rpc.nisd–Y`
		or
		`rpc.nisd –Y –B`

Convert a server to a root replica.	Root master server as superuser	`nisserver–R–d`*domain.* `–h` *clientname*
Convert a server to a non-root master server.	Root master server as superuser	`nisserver –M–d`*newsubdomain.domain*`.` `–h\`*clientmachine*
Populate the new master server tables from files or from NIS maps.	New subdomain master server as superuser	`nispopulate –F–p`*/subdomaindirectory* `–d \` *newsubdomain .domain* . or `nispopulate –Y–d`*newsubdomain .domain*`.–h` *NISservername* `–a`*NIS_server_ipaddress* `–y` *NIS_domain*
Convert a client to a master server replica.	Subdomain master server as superuser	`nisserver–R–d`*subdomain .domain*`. – h` *clientname*

TABLE 4–2 NIS+ Domains Configuration Command Lines Summary *(continued)*

Action	Machine	Command
Initialize a new client of the subdomain. Clients can be converted to subdomain replicas or to another server.	New subdomain client machine as superuser	`nisclient –i –d` *newsubdomain.domain*`. – h \` *subdomainmaster*
Initialize user as an NIS+ client.	Client machine as user	`nisclient –u`

Note - To see what commands an NIS+ script calls, without actually executing the commands, use the `–x` option. The `–x` option causes the command names and their approximate output to echo to the screen as if you were actually running the script. Running the scripts for the first time with `–x` can minimize unexpected results. For more information, see the man pages for the scripts.

Setting Up NIS+ Root Servers

Setting up the root master server is the first activity towards establishing NIS+ domain. This section shows you how to configure a root master server using the `nisserver` script with default settings. The root master server uses the following defaults:

- Security level 2 (DES)—the highest level of NIS+ security
- NIS compatibility set to OFF (instructions for setting NIS compatibility are included)
- System information files (`/etc`) or NIS maps as the source of name services information
- `admin.` *domainname* as the NIS+ group

Note - The `nisserver` script modifies the name service switch file for NIS+ when it sets up a root master server. The `/etc/nsswitch.conf` file can be changed later. See *Solaris Naming Administration Guide* and Chapter 1 for information on the name service switch.

Prerequisites to Running `nisserver`

Check to see that the `/etc/passwd` file on the machine you want to be root master server contains an entry for root.

Information You Need

You need the following:

- The superuser password of the workstation that will become the root master server
- The name of the new root domain. The root domain name must have at least two elements (labels) and end in a dot (for example, *something*`.com.`). The last element may be anything you want, but in order to maintain Internet compatibility, the last element must be either an Internet organizational name (as shown in Table 4–3), or a two or three character geographic identifier such as `.jp`. for Japan.

TABLE 4–3 Internet Organizational Domains

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

In the following example, the machine that is designated as the root master server is called `master1`, and `doc.com.` becomes the new root domain.

Note - Domains and hosts should not have the same name. For example, if you have `doc.com.` as a root domain, you should not have a machine named `doc` in any of your domains. Similarly, if you have a machine named `home`, you do not want to create a domain named `home`. This caution also applies to subdomains; for example, if you have a machine named `west`, you do not want to create a `sales.west.myco.com` subdomain.

▼ How to Create a Root Master Server

1. **Set the superuser's** `PATH` **variable to include** `/usr/lib/nis`**.**

   Either add this path to root's `.cshrc` or `.profile` file or set the variable directly.

2. **Optionally, if using DES authentication, specify the Diffie-Hellman key length.**

   To use 640–bit Diffie-Hellman keys as well as the default 192–bit keys, type:

   ```
   nisauthconf dh640-0 des
   ```

   To allow only 640–bit keys (rejects 192–bit keys), type:

   ```
   nisauthconf dh640-0
   ```

3. **Type the following command as superuser (root) to configure a root master server.**

   The –`r` option indicates that a root master server should be configure. The –`d` option specifies the NIS+ domain name.

```
master1# nisserver -r -d doc.com.
This script sets up this machine ‘‘master1’’ as a NIS+ root master
server for domain doc.com.
Domain name : doc.com.
NIS+ group : admin.doc.com.
NIS (YP) compatibility : OFF
Security level : 2=DES
Is this information correct? (type ’y’ to accept, ’n’ to change)
```

“NIS+ group” refers to the group of users who are authorized to modify the information in the `doc.com.` domain. (Domain names always end with a period.) Modification includes deletion. `admin.`*domainname* is the default name of the group. See “How to Change Incorrect Information” on page 53 for instructions on how to change this name.

“NIS compatibility” refers to whether an NIS+ server accepts information requests from NIS clients. When set to `OFF`, the default setting, the NIS+ server does not fulfill requests from NIS clients. When set to `ON`, an NIS+ server fulfills such requests. You can change the NIS-compatibility setting with this script. See “How to Change Incorrect Information” on page 53.

Note - This script sets machines up only at security level 2, the highest level of NIS+ security. You cannot change the security level when using this script. After the script has completed, you can change the security level with the appropriate NIS+ command. See *Solaris Naming Administration Guide* and the `rpc.nisd` man page for more information on changing security levels.

4. **Type `y` (if the information shown on the screen is correct).**

   Typing `n` causes the script to prompt you for the correct information. (See “How to Change Incorrect Information” on page 53 for what you need to do if you type `n`.)

```
Is this information correct? (type ’y’ to accept, ’n’’ to change)
y
This script will set up your machine as a root master server for
domain doc.com. without NIS compatibility at security level 2.
Use "nisclient -r" to restore your current network service environment.
Do you want to continue? (type ‘y’ to continue, ‘n’ to exit the script)
```

5. **Type y to continue NIS+ configuration.**

   (Typing `n` safely stops the script.) If you interrupt the script after you have chosen `y` and while the script is running, the script stops running and leaves configured whatever it has created so far. The script does not do any automatic recovery or cleaning up. You can always rerun this script.

```
Do you want to continue? (type ’y’ to continue, ’n’ to exit the script
y
setting up domain information ‘‘doc.com.’’ ...
setting up switch information ...
running nisinit ...
This machine is in the doc.com. NIS+ domain.
Setting up root server ...
All done.
starting root server at security level 0 to create credentials...
running nissetup ...
(creating standard directories & tables)
org_dir.doc.com. created
Enter login password:
```

The `nissetup` command creates the directories for each NIS+ table.

6. **Type your machine’s root password at the prompt and press Return.**

   In this case, the user typed the `master1` machine’s root password.

```
Wrote secret key into /etc/.rootkey
setting NIS+ group to admin.doc.com. ...
restarting root server at security level 2 ...
This system is now configured as a root server for domain doc.com.
You can now populate the standard NIS+ tables by using the
nispopulate or /usr/lib/nis/nisaddent commands.
```

Your root master server is now configured and ready for you to populate the NIS+ standard tables. To continue with populating tables, skip to "Populating NIS+ Tables" on page 55.

▼ How to Change Incorrect Information

If you typed `n` because some or all of the information returned to you was wrong in Step 4 on page 52 in the above procedure, you will see the following:

```
Is this information correct? (type 'y' to accept, 'n' to change)
 n
Domain name: [doc.com.]
```

1. **Press Return if the domain name is correct; otherwise, type the correct domain name and press Return.**

   In this example, Return was pressed, confirming that the desired domain name is `doc.com.`. The script then prompts for the NIS+ group name.

```
Is this information correct? (type 'y' to accept, 'n' to change)
 n
Domain name: [doc.com.]
NIS+ group: [admin.doc.com.]
```

2. **Press Return if NIS+ group is correct; otherwise, type the correct NIS+ group name and press Return.**

   In this example, the name was changed. The script then prompts for NIS compatibility.

```
NIS+ group: [admin.doc.com.] netadmin.doc.com.
NIS (YP) compatibility (0=off, 1=on): [0]
```

3. **Press Return if you do not want NIS compatibility; otherwise, type `1` and press Return.**

   In this example, Return was pressed, confirming that NIS compatibility status is correct. Once again, the script asks you if the information is correct.

   Note - If you choose to make this server NIS compatible, you also need to edit a file and restart the `rpc.nisd` daemon before it will work. See "Configuring a Client as an NIS+ Server" on page 69 for more information.

```
NIS (YP) compatibility (0=off, 1=on): [0]
Domain name : doc.com.
NIS+ group : netadmin.doc.com.
NIS (YP) compatibility : OFF
Security level : 2=DES
Is this information correct? (type 'y' to accept, 'n' to change)
```

When the information is correct, continue with Step 3 in "How to Create a Root Master Server" on page 51. You can keep choosing –`n` until the information is correct.

▼ How to Set Up a Multihomed NIS+ Root Master Server

The procedure for setting up a multihomed NIS+ server is the same as setting up a single interface server. The only difference is that there are more interfaces that need to be defined in the hosts database (`/etc/hosts` and `/etc/inet/ipnodes` files, and NIS+ `hosts` and `ipnodes` tables). Once the host information is defined, use the `nisclient` and `nisserver` scripts to set up the multihomed NIS+ server. For information about setting up a multihomed replica server, see "How to Set Up Multihomed NIS+ Replica Servers" on page 73

Caution - When setting up a multihomed NIS+ server, the server's primary name must be the same as the nodename for the system. This is a requirement of both Secured RPC and `nisclient`.

- Secured RPC relies on the nodename to create the netname for authentication.
- `nisclient` relies on the primary name to create the credential for the client.

If these names are different, Secure RPC authentication will fail to work properly causing NIS+ problems.

The following procedure shows how to set up a NIS+ root master server:

1. **On the root master, add the server host information into the `/etc/hosts` or `/etc/inet/ipnodes` file.**

   For example, the `/etc/hosts` file for the *hostA* system with three ethernet interfaces looks like:

```
127.0.0.1 localhost loghost
192.168.10.x hostA hostA-10 hostA-le0
192.168.11.y hostA hostA-11 hostA-le1
192.168.12.z hostA hostA-12
```

2. **Set up the server as a multihome NIS+ root server with `nisserver`.**

```
hostA# nisserver -r -d sun.com
```

   where our example shows *sun.com* as the root domain name. Issue the `nisserver` command using the name of your root domain name.

   After completing the steps for setting up a multihome NIS+ root server, the remainder of the setup is exactly the same as for a single interface server.

Populating NIS+ Tables

After the root master server has been configured, you can populate its standard NIS+ tables with name services information. This section shows you how to populate the root master server's tables with data from files or NIS maps using the `nispopulate` script with default settings. The script uses:

- The domain created in the previous example (doc.com.)
- System information files or NIS maps as the source of name services
- The standard NIS+ tables: auto_master, auto_home, ethers, group, hosts, networks, passwd, protocols, services, rpc, netmasks, bootparams, netgroup, and aliases

---

Note - The shadow file's contents are merged with the passwd file's to create the passwd table when files are the tables' information source. No shadow table is created.

Prerequisites to Running nispopulate

Before you can run the nispopulate script:

- View each local /etc file or NIS map from which you will load data. Make sure there are no spurious or incorrect entries. Make sure that the right data is in the correct place and format. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. You can always add individual entries after configuration is completed. That is easier than trying to load incomplete or damaged entries.
- The information in the files must be formatted appropriately for the table into which it will be loaded. *Solaris Naming Administration Guide* and Chapter 9 describe the format required for a text file to be transferred into its corresponding NIS+ table.
- Make sure that domain and host names are different. Domains and hosts cannot have the same name. For example, if you have a sales domain you cannot have a machine named sales. Similarly, if you have a machine named home, do not create a domain named home. This caution also applies to subdomains; for example, if you have a machine named west, do not create a sales.west.myco.com subdomain.
- Remove all dots and underscores in host names. NIS+ uses dots (periods) to delimit between machine names and domains and between parent and subdomains, so you cannot have a machine name containing a dot. You also cannot use underscores in hostnames, since DNS does not allow it. Before running the nispopulate script, you must eliminate any dots in your host names. You can convert host name dots to hyphens. For example, you cannot have a machine named sales.alpha. You can convert that name to sales-alpha.
- If you are setting up a network for the first time, you may not have much network information stored anywhere. In that case, you first need to collect the information, then type it into the *input file*—which is essentially the same as an /etc file.
- For safety's sake, you should make copies of the /etc files and use the copies to populate the tables instead of the actual ones. (This example uses files in a directory called /nisplusfiles, for instance.)

- Edit four of the copied NIS table files, `passwd`, `shadow`, `aliases`, and `hosts`, for security problems, particularly items that you do not want distributed across the namespace. For example, you might want to remove the following lines from the copy of your local `passwd` file so that they are not made available across the namespace:

```
root:x:0:1:0000-Admin(0000):/:/sbin/sh
daemon:x:1:3:0000-Admin(0000):/:
bin:x:3:5:0000-Admin(0000):/usr/bin:
sys:x:3:3:0000-Admin(0000):/:
adm:x:4:4:0000-Admin(0000):/var/adm:
lp:x:78:9:0000-lp(0000):/usr/spool/lp:
smtp:x:0:0:mail daemon user:/:
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp:x:7:8:0000-uucp (0000):/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:22:6:Network Admin:/usr/net/nls
nobody:x:60000:60000:uid no body:/:
noaccess:x:60002:60002:uid no access:/:
```

- The domain must have already been configured and its master server must be running.
- The domain's server must have sufficient disk space to accommodate the new table information.
- You must be logged in as an NIS+ principal (a client with appropriate credentials) and have write permission to the NIS+ tables in the specified domain. In this example, you must be the user `root` on the machine `master1`.

Information You Need

If populating from files, you need:

- The new NIS+ domain name
- The path of the appropriately edited text files whose data will be transferred
- Your root password

If populating from NIS maps, you need:

- The new NIS+ domain name
- The NIS domain name
- The NIS server's name
- The IP address of the NIS server
- Your root password

Note - The NIS domain name is case-sensitive, while the NIS+ domain name is not.

▼ How to Populate the Root Master Server Tables

1. **Perform either substep *a* or *b* to populate the root master server tables, then continue with Step 2 on page 59.**

   Substep *a* shows you how to populate tables from files. Substep *b* shows you how to populate tables from NIS maps. Type these commands in a scrolling window; otherwise, the script's output might scroll off the screen.

   Note - The `nispopulate` script can fail if there is insufficient `/tmp` space on the system. To keep this from happening, you can set the environment variable `TMPDIR` to a different directory. If `TMPDIR` is not set to a valid directory, the script uses the `/tmp` directory.

   ---

   a. **Type the following command to populate the tables from files.**

```
master1# nispopulate -F -p /nis+files -d doc.com.
NIS+ domain name : doc.com.
Directory Path : /nis+files
Is this information correct? (type 'y' to accept, 'n' to change)
```

The –`F` option indicates that the tables take their data from files. The –`p` option specifies the directory search path for the source files. (In this case, the path is `/nis+files`.) The –`d` option specifies the NIS+ domain name. (In this case, the domain name is `doc.com.`)

The NIS+ principal user is root. You must perform this task as superuser in this instance because this is the first time that you are going to populate the root master server's tables. The `nispopulate` script adds credentials for all members of the NIS+ admin group

   b. **Type the following command to populate the tables from NIS maps.**

```
master1# nispopulate -Y -d doc.com. -h salesmaster -a 130.48.58.111
-y sales.doc.com.
NIS+ domain name : doc.com.
NIS (YP) domain : sales.doc.com.
NIS (YP) server hostname : salesmaster
Is this information correct? (type 'y' to accept, 'n' to change)
```

The –`Y` option indicates that the tables take their data from NIS maps. The –`d` option specifies the NIS+ domain name. The –`h` option specifies the NIS server's machine name. (In this case, the NIS server's name is `salesmaster`.

You have to insert the name of a real NIS server at your site to create the sample domain.) The –a option specifies the NIS server’s IP address. (In this case, the address is 130.48.58.111. You have to insert the IP address of a real NIS server at your site to create the sample domain.) The –y option specifies the NIS domain name. (In this case, the domain’s name is sales.doc.com.; you have to insert the NIS domain name of the real NIS domain at your site to create the sample domain.

The NIS+ principal user is root. You must perform this task as superuser in this instance because this is the first time that you are going to populate the root master server’s tables. The nispopulate script also adds credentials for all members of the NIS+ admin group.

2. **Type y (if the information returned on the screen is correct).**

   Typing n causes the script to prompt you for the correct information. (See “How to Change Incorrect Information” on page 53 for what you need to do if the information is incorrect.)

   - If you performed substep *a* of Step 1 on page 58, you will see the following:

```
Is this information correct?
(type ’y’ to accept, ’n’ to change)
y

This script will populate the following NIS+ tables for domain doc.com. from
the files in /nis+files: auto_master auto_home ethers group hosts networks
passwd protocols services rpc netmasks bootparams netgroup aliases shadow
**WARNING: Interrupting this script after choosing to continue may leave
the tables only partially populated. This script does not do any automatic
recovery or cleanup.
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
```

   - If you performed substep *b* of Step 1 on page 58, you will see the following:

```
Is this information correct? (type ’y’ to accept, ’n’ to change)
y
This script will populate the following NIS+ tables for domain doc.com. from the
NIS (YP) maps in domain sales: auto_master auto_home ethers group hosts networks
passwd protocols services rpc netmasks bootparams netgroup aliases
**WARNING: Interrupting this script after choosing to continue may leave the
 tables only partially populated. This script does not do any automatic recovery
or cleanup.
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
```

(continued)

(Continuation)

3. **Type `y` to continue populating the tables.**

   (Typing `n` safely stops the script.) If you interrupt the script after you have chosen `y`—while the script's running—the script stops running and can leave the tables only partially populated. The script does not do any automatic recovery or cleaning up. You can safely rerun the script, but the tables will be overwritten with the latest information.

   - If you are populating tables from files, you see messages like the following as the script uses `hosts` and `passwd` information to create the credentials for hosts and users:

```
Do you want to continue? (type 'y' to continue, 'n' to exit this script)
y
populating auto_master table from file /nis+files/auto_master
... auto_master table done.
populating auto_home table from file /nis+files/auto_home
... auto_home table done.
Credentials have been added for the entries in the hosts and passwd table(s).
Each entry was given a default network password (also known as a Secure-
RPC password). This password is: nisplus
Use this password when the nisclient script requests the network password.
Done!
```

     Note and remember the Secure RPC password (`nisplus`, in the above example). Use this password when prompted for your network or Secure RPC password.

     The script continues until it has searched for all the files it expects and loads all the tables it can from the available files.

   - If you are populating tables from NIS maps, you will see messages like the following as the script uses `hosts` and `passwd` information to create the credentials for hosts and users:

```
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
y
populating auto_master table from sales.doc.com. NIS(YP) domain...
auto_master table done.
populating auto_home table from file sales.doc.com. NIS(YP) domain...
auto_home table done.
....
Credentials have been added for the entries in the hosts and passwd table(s).
Each entry was given a default network password (also known as a Secure-RPC password).
This password is: nisplus
Use this password when the nisclient script requests the network password.
Done!
```

Note and remember the Secure RPC password (`nisplus`, in the above example). Use this password when prompted for your network or Secure RPC password.

All the tables are now populated. You can ignore any `parse error` warnings. Such errors indicate that NIS+ found empty or unexpected values in a field of a particular NIS map. You may want to verify the data later after the script completes.

4. **(Optional) Add yourself and others to the root domain’s admin group.**

   For example, if your login ID is `topadm` and your co-worker’s ID is `secondadmin`, you enter:

```
master1# nisgrpadm -a admin.doc.com. topadm.doc.com. secondadm.doc.com.
Added ‘‘topadm.doc.com.’’ to group ‘‘admin.doc.com.’’.
Added ‘‘secondadm.doc.com.’’ to group ‘‘admin.doc.com.’’.
```

The `admin.doc.com.` argument in the `nisgrpadm –a` command above is the group name, which must come first. The remaining two arguments are the names of the administrators.

Note - This step is necessary only if you want to add additional users to the admin group now, which is a good time to add administrators to the root server. You can also add users to the admin group after you have configured NIS+.

You do not have to wait for the other administrators to change their default passwords to perform this step; however, they must already be listed in the `passwd` table before you can add them to the admin group. Members of the admin group will be unable to act as NIS+ principals until they add themselves to the domain. See “How to Initialize an NIS+ User” on page 67 for more information on initializing users. The group cache also has to expire before the new members become active.

5. **Type the following command to checkpoint the domain.**

```
master1# nisping -C doc.com.
Checkpointing replicas serving directory doc.com.
Master server is master1.doc.com.
 Last update occurred at date
Master server is master1.doc.com.
checkpoint scheduled on master1.doc.com.
```

This step ensures that all the servers supporting the domain transfer the new information from their initialization (`.log`) files to the disk-based copies of the tables. Since you have just configured the root domain, this step affects only the root master server, as the root domain does not yet have replicas.

Caution - If you do not have enough swap or disk space, the server will be unable to checkpoint properly, but it will not notify you. One way to make sure everything is correct is to list the contents of a table with the `niscat` command. For example, to check the contents of the `rpc` table, type:

```
master1# niscat rpc.org_dir
rpcbind rpcbind 100000
rpcbind portmap 100000
rpcbind sunrpc 100000
```

If you do not have enough swap space, you will see the following error message instead of the sort of output you see above.

```
can’t list table: Server busy, Try Again.
```

Even though it does not say so, in this context this message indicates that you do not have enough swap space. Increase the swap space and checkpoint the domain again.

Setting Up NIS+ Client Machines

After the root master server's tables have been populated from files or NIS maps, you can initialize NIS+ client machines. (Because the root master server is an NIS+ client of its own domain, no further steps are required to initialize it.) This section shows you how to initialize an NIS+ client by using the `nisclient` script with default settings. The script will use:

- The domain used in previous examples, `doc.com.`
- The Secure RPC password (also known as the network password) created by the `nispopulate` script in the previous example (`nisplus`, the default password)

Note - The `–i` option used in "How to Initialize a New Client Machine" on page 64 does not configure an NIS+ client to resolve host names requiring DNS. You need to explicitly include DNS for clients in their name service switch files. See *Solaris Naming Administration Guide* and Chapter 1 for more information on resolving host names through DNS.

Prerequisites to Running `nisclient`

Before you can use the `nisclient` script:

- The domain must have already been configured and its master server must be running.
- The master server of the domain's tables must be populated. (At a minimum, the hosts or ipnodes table must have an entry for the new client machine.)
- You must be logged in as superuser on the machine that is to become an NIS+ client. In this example, the new client machine is named `client1`.

Information You Need

You need:

- The domain name
- The default Secure RPC password (`nisplus`)
- The root password of the workstation that will become the client
- The IP address of the NIS+ server (in the client's home domain)
- If DES authentication is used, note the Diffie-Hellman key length used on the master server. Use `nisauthconf` to ascertain the master server Diffie-Hellman key length.

▼ How to Initialize a New Client Machine

1. **Optionally, if using DES authentication, specify the Diffie-Hellman key length.**

   On the master server, type

   ```
   nisauthconf
   ```

   Use the output as the arguments when running the `nisauthconf` command on the client. For example, if `nisauthconf` on the master server produces

   ```
   dh640dh-0 des
   ```

   type the following command on the client machine

   ```
   nisauthconf dh640dh-0 des
   ```

2. **Type the following command to initialize the new client on the new client machine.**

   The `–i` option initializes a client. The `–d` option specifies the new NIS+ domain name. (If the domain name is not specified, the default is the current domain name.) The `–h` option specifies the NIS+ server's host name.

```
client1# nisclient -i -d doc.com. -h master1
Initializing client client1 for domain ‘‘doc.com.’’.
Once initialization is done, you will need to reboot your machine.
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
```

3. **Type** `y`**.**

   Typing `n` exits the script. The script prompts you only for the root server's IP address if there is no entry for it in the client's `/etc/hosts` or `/etc/inet/ipnodes` file.

```
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
y
Type server master1’s IP address:
```

4. **Type the correct IP address, and press Return.**

   This example uses the hypothetical address `123.123.123.123`.

```
Type server master1’s IP address: 123.123.123.123
setting up the domain information...
setting up the name service switch information...
At the prompt below, type the network password (also known as the
Secure-RPC password) that you obtained either from your administrator or
from running the nispopulate script.
 Please enter the Secure-RPC password for root:
```

5. **Type the Secure RPC password (also known as the network password) only if the Secure RPC password differs from the root login password.**

   In this case, use the default, `nisplus` .

   The password does not echo on the screen. If you mistype it, you are prompted for the correct one. If you mistype it twice, the script exits and restores your previous network service. If this happens, try running the script again.

   ```
   Please enter the login password for root:
   ```

6. **Type the root password for this client machine.**

   The password does not echo on the screen. (If the Secure RPC password and the root login password happen to be the same, you will not be prompted for the root login password.)

   Typing the root password changes the credentials for this machine. The RPC password and the root password are now the same for this machine.

```
Please enter the login password for root:
Wrote secret key into /etc/.rootkey
Your network password has been changed to your login one.
Your network and login passwords are now the same.
Client initialization completed!!
Please reboot your machine for changes to take effect.
```

7. **Reboot your new client machine.**

   Your changes do not take effect until you reboot the machine.

   You can now have the users of this NIS+ client machine add themselves to the NIS+ domain.

Creating Additional Client Machines

Repeat the preceding client-initiation procedure on as many machines as you like. To initiate clients for another domain, repeat the procedure but change the domain and master server names appropriately.

The sample NIS+ domain described in this chapter assumes that you will initialize four clients in the `doc.com.` domain. You are then going to configure two of the clients as non-root NIS+ servers and a third client as a root replica of the root master server of the `doc.com.` domain.

Note - You always have to make a system into a client of the parent domain before you can make the same system a server of any type.

Initializing NIS+ Client Users

After a machine has become an NIS+ client, the users of that machine must add themselves to the NIS+ domain. Adding a user to the domain means changing the Secure RPC password to that user's login password. What actually happens is that the user's password and the Secure RPC password are bound together. This procedure uses the `nisclient` script.

Prerequisites to Running `nisclient`

Before you can use the `nisclient` script to initialize a user:

- The domain must have already been configured and its master server must be running.
- The master server of the domain's tables must be populated. (At a minimum, the hosts table must have an entry for the new client machine.)
- You must have initialized a client machine in the domain.
- You must be logged in as a *user* on the client machine. In this example, the user is named `user1`.
- Optionally, if using DES authentication, the client machine must use the same Diffie-Hellman key configuration as that used on the master server.

Information You Need

You need:

- A user's login name (`user1` in this example)

- The default Secure RPC password (`nisplus` in this example)
- The login password of the user who will become the NIS+ client

▼ How to Initialize an NIS+ User

1. **To become an NIS+ client, enter the following** `nisclient` **command while logged in as the user.**

```
user1prompt% nisclient -u
At the prompt below, type the network password (also known as the
Secure-RPC password) that you obtained either from your administrator
or from running the nispopulate script.
Please enter the Secure-RPC password for user1:
```

2. **Enter the Secure RPC password, which is** `nisplus` **in this case.**

   The password does not echo on the screen.

```
Please enter the login password for user1:
```

3. **Type the user's login password and press Return.**

   The password does not echo on the screen.

```
Your network password has been changed to your login one.
Your network and login passwords are now the same
```

This user is now an NIS+ client. You need to have all users make themselves NIS+ clients.

Setting Up NIS+ Servers

Now that the client machines have been initialized, you can change any of them to NIS+ servers of the following types:

- To be root replicas—to contain copies of the NIS+ tables that reside on the root master server
- To be master servers of subdomains of the root domain
- To be replicas of master servers of subdomains of the root domain

---

Note - You can have only one NIS+ master root server. Root NIS+ servers are a special type of NIS+ server. This section does not describe how to configure a root master server; see "Setting Up NIS+ Root Servers" on page 49 for more information.

You can configure servers any of these different ways:

- Without NIS compatibility
- With NIS compatibility
- With NIS compatibility and DNS forwarding—you only need to set DNS forwarding if you are going to have SunOS 4.x clients in your NIS+ namespace (see *NIS+ Transition Guide* for more information on using NIS-compatibility mode)

Servers and their replicas should have the same NIS-compatibility settings. If they do not have the same settings, a client that needs NIS compatibility set to receive network information may not be able to receive it if either the server or replica it needs is unavailable.

This example shows the machine `client1` being changed to a server. This procedure uses the NIS+ `rpc.nisd` command instead of an NIS+ script.

Prerequisites to Running `rpc.nisd`

Before you can run `rpc.nisd`:

- The domain must have already been configured and its master server must be running.
- The master server of the domain's tables must be populated. (At a minimum, the hosts table must have an entry for the new client machine.)
- You must have initialized the client machine in the domain.
- You must be logged in as root on the client machine. In this example, the client machine is named `client1`.
- Optionally, if using DES authentication, the client machine must use the same Diffie-Hellman key configuration as that used on the master server.

Information You Need

You need the superuser password of the client that you will convert into a server.

Configuring a Client as an NIS+ Server

Perform any of the following to alternate procedures to configure a client as a server. These procedures create a directory with the same name as the server and create the server’s initialization files which are placed in `/var/nis`.

Note - All servers in the same domain must have the same NIS-compatibility setting. For example, if the master server is NIS compatible, then its replicas should also be NIS compatible.

How to Configure a Server Without NIS Compatibility

1. **To configure a server without NIS compatibility, enter the following command:**

```
client1# rpc.nisd
```

How to Configure a Server With NIS Compatibility

1. **Edit the** `/etc/init.d/rpc` **file on the server to uncomment the whole line containing the string** `–EMULYP=’’-Y’’`**.**

   To do this, remove the # character from the beginning of the line.

2. **Type the following as superuser.**

```
client1# rpc.nisd -Y
```

How to Configure a Server With DNS and NIS Compatibility

This procedure configures a NIS+ server with both DNS forwarding and NIS+ compatibility. Both of these features are needed to support SunOS 4.x clients.

1. **Edit the** `/etc/init.d/rpc` **file on the server to uncomment the whole line containing the string** `EMULYP=’’-Y’’`**.**

   To do this, remove the # character from the beginning of the line.

2. **Add** `–B` **to the above line inside the quotes.**

   The line should read:

   `–EMULYP=’’-Y -B’’`

3. **Type the following command as superuser.**

```
client1# rpc.nisd -Y -B
```

Now this server is ready to be designated a master or replica of a domain.

Creating Additional Servers

Repeat the preceding client-to-server conversion procedure on as many client machines as you like.

The sample NIS+ domain described in this chapter assumes that you will convert three clients to servers. You will then configure one of the servers as a root replica, another as a master of a new subdomain, and the third as a replica of the master of the new subdomain.

Creating a Root Replica Server

To have regularly available NIS+ service, you should always create one or more root replica servers. Having replicas can also speed network-request resolution because multiple servers are available to handle requests.

For performance reasons, you should have no more than a few replicas per domain. If your network includes multiple subnets or different sites connected by a Wide Area Network (WAN), you may need additional replicas:

- *Subnets.* If you have a domain that spans multiple subnets, it is a good idea to have at least one replica server within each subnet so that if the connection between nets is temporarily out of service, each subnet can continue to function until the connection is restored.

- *Remote sites.* If you have a domain spanning multiple sites linked over a WAN, it is a good idea to have at least one replica server on each side of the WAN link. For example, it may make sense from an organizational point of view to have two physically distant sites in the same NIS+ domain. If the domain's master server and all of its replicas are at the first site, there will be much NIS+ network traffic between the first and second sites. Creating an additional replica at the second site should reduce network traffic. See *NIS+ Transition Guide* for more information on replica distribution.

See *Solaris Naming Administration Guide* for additional information on how to determine the optimum number of replicas.

"How to Create a Root Replica" on page 71 shows the machine `client1` being configured as a root replica for the `doc.com.` domain. This procedure uses the NIS+ `nisserver` script. (You can also use the NIS+ command set to configure a replica server as described in "Using NIS+ Commands to Configure a Replica Server" on page 131.)

Prerequisites to Running nisserver

Before you can run nisserver to create a replica:

- The domain must already have been configured and its master server must be running.
- The tables of the master server must be populated. (At a minimum, the hosts table must have an entry for the new client machine.)
- You must have initialized the new server as a client machine in the domain, as described in “Setting Up NIS+ Client Machines” on page 63.
- You must have started rpc.nisd on the new replica server, as described in “Setting Up NIS+ Servers” on page 67.
- You must be logged in as root on the root master server. In this example, the root master machine is named master1.

Information You Need

You need:

- The domain name
- The client machine name; (client1, in this example)
- The superuser password for the root master server

▼ How to Create a Root Replica

1. **To create a root replica, type the following command as superuser (root) on the NIS+ domain’s root master server.**

```
master1# nisserver -R -d doc.com. -h client1
This script sets up a NIS+ replica server for domain doc.com.
Domain name: :doc.com.
NIS+ server : :client1
Is this information correct? (type ’y’ to accept, ’n’ to change)
```

The –R option indicates that a replica should be configured. The –d option specifies the NIS+ domain name (doc.com., in this example). The –h option specifies the client machine (client1, in this example) that will become the root replica.

2. **Type y to continue.**

Typing n causes the script to prompt you for the correct information. (See “How to Change Incorrect Information” on page 53 for what you need to do if you type n.)

```
Is this information correct? (type ’y’ to accept, ’n’ to change)
y
This script will set up machine ‘‘client1’’ as an NIS+ replica server for domain
doc.com. without NIS compatibility. The NIS+ server daemon, rpc.nisd, must
be running on client1 with the proper options to serve this domain.
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
```

3. **Type** `y` **to continue.**

   Typing `n` safely stops the script. The script will exit on its own if `rpc.nisd` is *not* running on the client machine.

```
Is this information correct? (type ’y’ to continue, ’n’ to exit this script)
y
The system client1 is now configured as a replica server for domain doc.com..
The NIS+ server daemon, rpc.nisd, must be running on client1 with the proper
options to serve this domain. If you want to run this replica in NIS (YP)
compatibility mode, edit the /etc/init.d/rpc file on the replica server ’
to uncomment the line which sets EMULYP to "-Y". This will ensure that
rpc.nisd will boot in NIS-compatibility mode. Then, restart rpc.nisd with
the ‘‘-Y’’ option. These actions should be taken after this script completes.
```

Note - The above notice refers to an optional step. You need to modify only the `/etc/init.d/rpc` file if you want the root replica to be NIS compatible and it is not now NIS compatible. That is, the file needs modification only if you want the root replica to fulfill NIS client requests and it was not already configured as an NIS-compatible server. See “Configuring a Client as an NIS+ Server” on page 69 for more information on creating NIS-compatible servers.

4. **[Optional] Configure the replica to run in NIS (YP) compatibility mode.**

   If you want this replica to run in NIS compatibility mode, follow these steps:

   a. **Kill** `rpc.nisd`

   b. **Edit the server’s** `/etc/init.d/rpc` **file to uncomment the line that sets** `EMULYP` **to** `–Y`**.**

      In other words, delete the # character from the start of the `EMULYP` line.

   c. **Restart** `rpc.nisd`**.**

5. **Load your namespace data on to the new replica server.**

   You can do this in two ways:

   - The preferred method of loading data on to a new replica server is to use the NIS+ backup and restore capabilities to back up the master server, then "restore" that data on to the new replica server. This step is described in detail in "How to Load Namespace Data—`nisrestore` Method" on page 134.
   - Run `nisping`. Running `nisping` initiates a full resynch of all NIS+ data from the master server to this new replica. If your namespace is large, this can take a long time, during which your master server is very busy and slow to respond and your new replica is unable to answer NIS+ requests. This step is described in detail in "How to Load Namespace Data—`nisping` Method" on page 136.

   When you have finished loading your namespace data, the machine `client1` is now an NIS+ root replica. The new root replica can handle requests from the clients of the root domain. Because there are now two servers available to the domain, information requests can be fulfilled faster.

   Using these procedures, you can create as many root replicas as you need. You can also use these procedures to create replica servers for subdomains.

▼ How to Set Up Multihomed NIS+ Replica Servers

The procedure for setting up a multihomed NIS+ server is the same as setting up a single interface server. The only difference is that there are more interfaces that need to be defined in the hosts database (`/etc/hosts` and `/etc/inet/ipnodes` files, and NIS+ `hosts` and `ipnodes` tables). Once the host information is defined, use the `nisclient` and `nisserver` scripts to set up the multihomed NIS+ server.

Caution - When setting up a multihomed NIS+ server, the server's primary name must be the same as the nodename for the system. This is a requirement of both Secured RPC and `nisclient`.

- Secured RPC relies on the nodename to create the netname for authentication.
- `nisclient` relies on the primary name to create the credential for the client.

If these names are different, Secure RPC authentication will fail to work properly causing NIS+ problems.

This procedure shows how to set up any NIS+ non-root master servers. The following example creates a replica for the root domain. For information about setting up a multihomed root server, see "How to Set Up a Multihomed NIS+ Root Master Server" on page 54.

1. **Add the server host information into the `hosts` or `ipnodes` file.**

   For example, for the *hostB* system with three interfaces:

```
192.168.11.y hostB hostB-11
192.168.12.x hostB hostB-12
192.168.14.z hostB hostB-14
```

2. **On the root master server, use either** `nispopulate` **or** `nisaddent` **to load the new host information into the** `hosts` **or** `ipnodes` **table.**

   For example:

   ```
   hostA# nispopulate -F -d sun.com hosts
   ```

   where the example shows *sun.com* as the NIS+ root domain name. Issue the `nispopulate` command specifying the name of your NIS+ root domain name.

3. **On the root master server, use the** `nisclient` **script to create the credential for the new client.**

   For example:

   ```
   hostA# nisclient -c -d sun.com hostB
   ```

   where the example shows *sun.com* as the root domain name. Issue the `nisclient` command specifying the name of your root domain name.

4. **On the non-root master server, use** `nisclient` **to start the new server if it is not already running and initialize the machine as a NIS+ client.**

   For example:

   ```
   hostB# nisclient -i -d sun.com
   ```

   where the example shows *sun.com* as the root domain name. Issue the `nisclient` command specifying the name of your root domain name.

5. **On the root master server, use** `nisserver` **to create a non-root master.**

   For example:

   ```
   hostA# nisserver -M -d eng.sun.com -h hostB.sun.com.
   ```

   where the example shows *eng.sun.com* as the NIS+ domain name and *hostB.sun.com* as the fully-qualified hostname for the NIS+ server. Issue the `nisserver` command specifying the name of your NIS+ domain and the fully-qualified hostname for the NIS+ server.

6. **On the root master server, use** `nisserver` **to set up a replica server.**

For example:

```
hostA# nisserver -R -d sun.com -h hostB.sun.com.
```

where the example shows *sun.com* as the replica server and *hostB.sun.com* as the fully-qualified hostname for the NIS+ server. Issue the `nisserver` command specifying the name of your replica server and NIS+ domain.

After completing the steps for setting up a multihome NIS+ replica server, the remainder of the setup is exactly the same as for a single interface server.

Creating a Subdomain

This section shows you how to create the master server of a new non-root domain. The new domain will be a subdomain of the `doc.com.` domain. The hierarchical structure of NIS+ allows you to create a domain structure that parallels your organizational structure.

This example shows the machine `client2` being converted to the master server of a new domain called `sub.doc.com.`. This procedure uses the NIS+ script `nisserver`.

Prerequisites to Running `nisserver`

Before you can run `nisserver` to create a master server for a new non-root domain:

- The parent domain must already have been configured and its master server must be running.
- The parent domain's tables must be populated. (At a minimum, the hosts table must have an entry for the new client machine.)
- You must have initialized the new client machine in the parent domain.
- You must have started `rpc.nisd` on the client.
- You must have adequate permissions to add the new domain. In this case, you must be logged in as root on the parent master server. In this example, the parent master machine is named `master1`.

Information You Need

You need:

- A name for the new non-root domain—the name of the new domain includes the name of the parent domain with this syntax: *newdomain.rootdomain*.

- The client machine name (client2, in this example)
- The superuser password for the parent master server

In "How to Create a New Non-Root Domain" on page 76, the new non-root domain is called sub.doc.com..

Note - In Solaris release 2.6 and earlier, any NIS+ client can be converted to an NIS+ master server as long as it is itself in a domain above the domain it is serving. For example, an NIS+ client in domain sales.doc.com. can serve domains below it in the hierarchy, such as west.sales.doc.com. or even alameda.west.sales.doc.com.. This client cannot, however, serve the domain doc.com., because doc.com. is above the domain sales.doc.com. in the hierarchy. Root replicas are the only exception to this rule. They are clients of the domain that they serve.

Note - In Solaris release 7, the domainname of any non-root NIS+ server can be set to the domain it serves. The non-root server behaves as if it lives in its own domain. This allows you to configure applications on the non-root server to use the information provided by the domain above it in the hierarchy.

The non-root server's credentials must still be in the domain above it in the hierarchy. Configure the non-root servers as described in "How to Create a New Non-Root Domain" on page 76. Only after the servers are properly configured, can you change the domainname to that of the domain it serves. See the –k option of nisinit and the –d option of nisserver.

▼ How to Create a New Non-Root Domain

1. **Type the following command as superuser (root) on the NIS+ domain's root master server to create a new non-root domain master server.**

   The –M option indicates that a master server for a new non-root domain should be created. The –d option specifies the *new* domain name, sales.doc.com. in this instance. The –h option specifies the client machine, (client2, in this example), that will become the master server of the new domain.

```
master1# nisserver -M -d sales.doc.com. -h client2
This script sets up a non-root NIS+ master server for domain sales.doc.com.
Domain name : sales.doc.com.
NIS+ server : client2
NIS+ group : admin.sales.doc.com.
NIS (YP) compatibility : OFF
Security level : 2=DES
Is this information correct? (type 'y' to accept, 'n' to change)
```

(continued)

(Continuation)

Master servers of new non-root domains are created with the same set of default values as root servers. See “How to Create a Root Master Server” on page 51 for more information on NIS+ group, NIS compatibility, and security level.

2. **Type `y` to continue.**

   Typing `n` causes the script to prompt you for the correct information. (See “How to Change Incorrect Information” on page 53 for what you need to do if you type n.)

```
Is this information correct? (type ’y’ to accept, ’n’ to change)
y
This script sets up machine ‘‘client2’’ as an NIS+ non-
root master server for domain sales.doc.com.
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
```

3. **Type `y` to continue.**

   Typing `n` safely exits the script. The script exits on its own if `rpc.nisd` is *not* running on the client machine.

```
Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
y
running nissetup ...
org_dir.sales.doc.com. created
groups_dir.sales.doc.com. created
...
...
setting NIS+ group admin.sales.doc.com. ...
The system client2 is now configured as a non-root server for domain sales.doc.com.
You can now populate the standard NIS+ tables by using the nispopulate or
/usr/lib/nis/nisaddent commands.
```

The machine `client2` is now the master server of the `sales.doc.com.` domain. The `sales.doc.com.` domain is a subdomain of the `doc.com.` domain. The machine `client2` is simultaneously still a client of the root domain `doc.com.`, and the master server of the `sales.doc.com.` domain.

You can now populate the standard NIS+ tables on the new master server of the sales.doc.com. domain.

Creating Additional Domains

Repeat the preceding procedure for changing servers to master servers of new non-root domains on as many server machines as you like. Every new master server is a new domain. Plan your domain structure before you start creating a NIS+ namespace. See Chapter 2 for more information on planning an NIS+ hierarchy.

Populating the New Subdomain's Tables

After you have created a new domain, you need to populate its master server's standard NIS+ tables. You use the same procedure to populate the new master server's tables as you used to populate the root master server's tables. The major difference is that the nispopulate script is run on the new master server instead of on the root master server. The domain names and file paths or NIS servers' names may change as well.

This example shows the tables of the new domain, sales.doc.com., being populated.

Prerequisites to Running nispopulate

Before you can run the nispopulate script to populate the new master server's tables:

- The information in the files must be formatted appropriately for the table into which it will be loaded.
  - Before proceeding, view each local /etc file or NIS map that you will be loading data from. Make sure that there are no spurious or incorrect entries. Make sure that the right data is in the correct place and format. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. You can always add individual entries after configuration is completed. That is easier than trying to load incomplete or damaged entries.
  - If you are setting up a network for the first time, you may not have much network information stored anywhere. In that case, you'll need to first get the information and then enter it manually into the *input file*—which is essentially the same as an /etc file.

- You should make copies of the /etc files and use the copies to populate the tables instead of the actual ones for safety reasons. (This example uses files in a directory called /nis+files, for instance.)
- Edit four of the copied NIS table files, passwd, shadow, aliases, and hosts, for security reasons. For example, you might want to remove the following lines from the copy of your local passwd file so they are not distributed across the namespace:

```
root:x:0:1:0000-Admin(0000):/:/sbin/sh
daemon:x:1:3:0000-Admin(0000):/:
bin:x:3:5:0000-Admin(0000):/usr/bin:
sys:x:3:3:0000-Admin(0000):/:
adm:x:4:4:0000-Admin(0000):/var/adm:
lp:x:78:9:0000-lp(0000):/usr/spool/lp:
smtp:x:0:0:mail daemon user:/:
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp:x:7:8:0000-
uucp (0000):/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:22:6:Network Admin:/usr/net/nls:
nobody:x:60000:60000:uid no body:/:
noaccess:x:60002:60002:uid no access:/:
```

- The domain must have already been configured and its master server must be running.
- The domain's servers must have sufficient disk space to accommodate the new table information.
- You must be logged in as an NIS+ principal and have write permission to the NIS+ tables in the specified domain. In this example, you would have to be the user root on the machine client2.

Note - The nispopulate script can fail if there is insufficient /tmp space on the system. To keep this from happening, you can set the environment variable TMPDIR to a different directory. If TMPDIR is not set to a valid directory, the script uses the /tmp directory instead.

Information You Need

If populating from files, you need:

- The new NIS+ domain name
- The path of the appropriately edited text files whose data will be transferred
- The root password of the NIS+ master server

If populating from NIS maps, you need:

- The new NIS+ domain name

- The NIS domain name
- The NIS server’s name
- The IP address of the NIS server
- The root password of the NIS+ master server

Note - The NIS domain name is case-sensitive, while the NIS+ domain name is not.

Populating the Master Server Tables

Since this procedure is essentially the same as the procedure shown in “How to Populate the Root Master Server Tables” on page 58, this example shows you only what you would type to populate the tables of the new domain, sales.doc.com.. For more information about this procedure, see “How to Populate the Root Master Server Tables” on page 58.

Note - This script should be run on the new domain’s master server, not the root master server.

The alternate methods of populating the master server tables on the new master server are:

- You can populate master server tables from files.
- You can populate master server tables from NIS maps.

Whichever method you choose should be executed in a scrolling window as the script’s output might otherwise scroll off the screen.

How to Populate the Tables From Files

To populate master server tables from files, type the following commands.

```
client2# nispopulate -F -p /nis+files -d sales.doc.com.
NIS+ domain name : sales.doc.com.
Directory Path : /nis+files
Is this information correct? (type ’y’ to accept, ’n’ to change
```

How to Populate the Tables From NIS Maps

To populate master server tables from NIS maps, type the following commands.

```
client2# nispopulate -Y -d sales.doc.com. -h businessmachine -a
 IP_addr_of_NIS_server -y business.doc.com.
NIS+ Domain name : sales.doc.com.
NIS (YP) domain : business.doc.com.
NIS (YP) server hostname : businessmachine
Is this information correct? (type 'y' to accept, 'n' to change)
```

See "How to Populate the Root Master Server Tables" on page 58 for additional information.

Creating Subdomain Replicas

The same principles that apply to root domain replicas apply to subdomain replicas (see "Creating a Root Replica Server" on page 70).

You use the same procedure to create a subdomain replica as you do to create a root replica. The major difference between creating the root replica and a subdomain replica is that the machine you are going to convert to a subdomain replica remains a client of the domain above the one it serves as a replica. This example shows you only what you type to create a replica for the new domain. For the rest of the script's output, see "How to Create a Root Replica" on page 71.

Prerequisites to Running `nisserver`

Before you can run `nisserver` to create a replica:

- The domain must have already been configured and its master server must be running.
- The domain's tables must be populated. (At a minimum, the hosts table must have an entry for the new client machine.)
- You must have initialized the client machine in the parent domain.
- You must have started `rpc.nisd` on the client.
- You must be logged in as root on the master server. In this example, the master machine is named `client2`.

Information You Need

- The domain name
- The client machine name (`client3,` in this example)

- The superuser password for the root master server

▼ How to Create a Replica

♦ **Run the** `nisserver -R` **command as superuser (root) on the NIS+ domain's master server.**

```
client2# nisserver -R -d sales.doc.com. -h client3
This script sets up a NIS+ replica server for domain sales.doc.com.
Domain name  ::sales.doc.com.
NIS+ server :client
Is this information correct? (type 'y' to accept, 'n' to change)
```

In this example, `client2` is the master server. The `–R` option indicates that a replica should be configured. The `–d` option specifies the NIS+ domain name (`sales.doc.com.` in this example). The `–h` option specifies the client machine (`client3`, in this example) that will become the replica. Notice that this machine is still a client of the `doc.com.` domain and not a client of the `sales.doc.com.` domain.

See "How to Create a Root Replica" on page 71 for the rest of this script's output.

Initializing Subdomain NIS+ Client Machines

After the master server's tables have been populated from files or NIS maps, you can initialize an NIS+ client machine. This section shows you how to initialize an NIS+ client in the new domain using the `nisclient` script with default settings. The NIS+ client machine is a different workstation from the NIS+ master server.

Note - The `–i` option used in "How to Initialize a New Subdomain Client Machine" on page 83 does not configure an NIS+ client to resolve host names requiring DNS. You need to explicitly include DNS for clients in their name service switch files. See "Enabling a Machine to Use DNS" on page 27 for more information on resolving host names through DNS.

You use the same procedure to initialize a client in the new domain as you do to initialize a client in the root domain. This example shows you only what you would type to initialize a client for the new domain. For the rest of the script’s output, see “How to Initialize a New Client Machine” on page 64.

Prerequisites to Running `nisclient`

Before you can use the `nisclient` script to initialize a user:

- The domain must have already been configured and its master server must be running.
- The master server of the domain’s tables must be populated. (At a minimum, the host’s table must have an entry for the new client machine.)
- You must have initialized a client machine in the domain.
- You must be logged in as a *user* on the client machine. In this example, the user is named user1.

Information You Need

You need:

- The domain name (`sales.doc.com.`, in this example)
- The default Secure RPC password (`nisplus`)
- The root password of the workstation that will become the client
- The IP address of the NIS+ server (in the client’s home domain) (in this example, the address of the master server, `client2`)

▼ How to Initialize a New Subdomain Client Machine

♦ **Type the following command as superuser to initialize the new client on the new client machine.**

```
subclient1# nisclient -i -d sales.doc.com. -h client2
Initializing client subclient1 for domain ‘‘sales.doc.com.’’.
Once initialization is done, you will need to reboot your machine.
Do you want to continue? (type ’Y’ to continue, ’N’ to exit this script)
```

The `–i` option initializes a client. The `–d` option specifies the new NIS+ domain name. (If the domain name is not specified, the default becomes the current domain name.) The `–h` option specifies the NIS+ server’s host name.

See "How to Initialize a New Client Machine" on page 64 for the rest of this script's output.

Initializing Subdomain NIS+ Client Users

You use the same procedure (`nisclient`) to initialize a user in the new domain as you do to initialize a user in the root domain. All users must make themselves NIS+ clients. This example shows you only what you would type to initialize a user for the new domain. For the rest of the script's output, see "How to Initialize an NIS+ User" on page 67.

Prerequisites to Running `nisclient`

Before you can use the `nisclient` script to initialize a user:

- The domain must have already been configured and its master server must be running.
- The master server of the domain's tables must be populated. (At a minimum, the hosts table must have an entry for the new client machine.)
- You must have initialized a client machine in the domain.
- You must be logged in as a *user* on the client machine. In this example, the user is named `user2`.

Information You Need

You need:

- The user's login name (`user2,` in this example)
- The default Secure RPC password (`nisplus`)
- The login password of the user that will become the NIS+ client

▼ How to Initialize an NIS+ Subdomain User

♦ **To become an NIS+ client, type the following command while logged in as the user.**

```
user2prompt% nisclient -u
At the prompt below, type the network password (also known as the
Secure-RPC password) that you obtained either from your administrator
or from running the nispopulate script.
Please enter the Secure-RPC password for user2:
```

See “How to Initialize an NIS+ User” on page 67 for the rest of this script’s output.

Summary of Commands for the Sample NIS+ Namespace

Table 4–4 summarizes the actual commands that you typed to create the sample namespace. The prompt preceding each command indicates on which machine the command should be typed.

TABLE 4–4 Creating the Sample Namespace: Command Summary

Tasks	Commands

Set environment path to include `/usr/lib/nis`—C shell or Bourne shell.	`# setenv PATH $PATH:/usr/lib/nis`
	or
	`# PATH=$PATH:/usr/lib/nis; export PATH`

Optionally configure Diffie-Hellman key length.	`master1# nisauthconf dh640-0 des`
Create root master server for `doc.com.` domain.	`master1# nisserver -r -d doc.com.`

Populate the root master server’s NIS+ tables—from files or from NIS maps.	`master1# nispopulate -F -p /nis+files -d doc.com.`
	or
	`master1# nispopulate -Y -d doc.com. -h salesmaster -a \ 130.48.58.111 -y sales.doc.com.`

Add additional members to the admin group (2).	`master1# nisgrpadm -a admin. doc.com. topadmin.doc.com. \ secondadmin.doc.com.`

TABLE 4–4 Creating the Sample Namespace: Command Summary *(continued)*

Tasks	Commands
Make a checkpoint of the NIS+ database.	`master1#` `nisping -C org_dir. doc.com.`
Optionally configure Diffie-Hellman key length.	`client1#` `nisauthconf dh640-0 des`
Initialize a NIS+ client machine in the doc.com. domain.	`client1#` `nisclient -i -d doc.com. -h master1`
Initialize user as a NIS+ client.	`client1user1prompt%` `nisclient -u`

Convert NIS+ client to NIS+ server, without or with NIS compatibility or with NIS and DNS.	`client1#rpc.nisd`
	or
	`client1#` `rpc.nisd -Y`
	or
	`client1#` `rpc.nisd -Y -B`

Create a root replica.	`master1#` `nisserver -R -d doc.com. -h client1`
Convert a server to a non-root master server of the `sales.doc.com.` domain.	`master1#` `nisserver -M -d sales.doc.com. -h client2`

Populate the new master server's NIS+ tables—from files or from NIS maps.	`client2#` `nispopulate -F -p /nis+files -d sales.doc.com.`
	or
	`client2#` `nispopulate -Y -d sales.doc.com. -h \`
	`businessmachine -a 130.48.58.242 -y business.doc.com.`

Create a master server replica.	`client2#` `nisserver -R -d sales.doc.com. -h client3`
Initialize a NIS+ client in the `sales.doc.com.`. domain.	`subclient1#` `nisclient -i -d sales.doc.com. -h client2`
Initialize user as a NIS+ client.	`subclient1user2prompt%` `nisclient -u`

CHAPTER 5

Setting Up the Root Domain

This chapter provides step-by-step instructions for setting up the root domain and DES authentication using the NIS+ command set.

- "Introduction to Setting Up the Root Domain" on page 87
- "Standard Versus NIS-Compatible Configuration Procedures" on page 88
- "Establishing the Root Domain" on page 88
- "Root Domain Configuration Summary" on page 105

See *Solaris Naming Administration Guide* for information on how to take an existing root master server out of service and replace it with a new machine.

Introduction to Setting Up the Root Domain

This task describes how to configure the root domain with the root master server running at security level 2 (the normal level).

Note - It is much easier to perform this task with the NIS+ installation scripts as described in Part 1 than with the NIS+ command set as described here. The methods described in this chapter should be used only by those administrators who are very familiar with NIS+ and who require some nonstandard features or configurations not provided by the installation scripts.

Setting up the root domain involves three major tasks:

- Preparing the root master server

- Creating the root domain
- Creating credentials for the root domain

However, setting up the root domain is not as simple as performing these three tasks in order; they are intertwined with one another. For instance, you must specify some security parameters before you create the root directory, the rest, after. To make the root domain easier to configure, this chapter separates these tasks into individual steps and arranges them into their most efficient order.

Standard Versus NIS-Compatible Configuration Procedures

The steps in this chapter apply to both a standard NIS+ root domain and an NIS-compatible root domain. There are, however, some important differences. The NIS+ daemon for an NIS-compatible domain must be started with the –Y option, which allows the root master server to answer requests from NIS clients. This is described in Step 11 on page 94. The equivalent step for standard NIS+ domains is Step 12 on page 95.

An NIS-compatible domain also requires read rights to the `passwd` table for the nobody class, which allows NIS clients to access the information stored in the table's `passwd` column. This is accomplished with the –Y option to the `nissetup` command, in Step 14 on page 97. The standard NIS+ domain version uses the same command but without the –Y option.

Establishing the Root Domain

The procedure describes each step in detail and provides related information. For those who do not need detailed instructions, a summary listing of the necessary commands is provided on "Root Domain Configuration Summary" on page 105.

Summary of Steps

Here is a summary of the entire configuration process:

1. Log in as superuser to the root master server.
2. Check the root master server's domain name.
3. Check the root master server's switch-configuration file.

4. Optionally, configure the Diffie-Hellman key length.
5. Clean out leftover NIS+ material and processes.
6. Name the root domain's admin group.
7. Create the root directory and initialize the root master server.
8. [NIS-compatibility Only] Start the NIS+ daemon with –Y. [Standard NIS+ Only] Start the NIS+ daemon.
9. Verify that the daemon is running.
10. Create the root domain's subdirectories and tables.
11. Create DES credentials for the root master server.
12. Create the root domain's admin group.
13. Add the root master to the root domain's admin group.
14. Update the root domain's public keys.
15. Start the NIS+ cache manager.
16. Restart the NIS+ daemon with security level 2.
17. Add your LOCAL credentials to the root domain.
18. Add your DES credentials to the root domain.
19. Add credentials for other administrators.
20. Add yourself and other administrators to the root domain's admin group.

Establishing the Root Domain—Task Map

TABLE 5–1 Establishing the Root Domain

Task	Description	For Instructions, Go To
Establishing the Root Domain	Use the `domainname` command to establish the root domain. Optionally extend the Diffie-Hellman key length. Stop and start the `ncsd` daemon. Kill and restart `keyserv`. Clean out leftover NIS+ information.	"How to Configure a Root Domain" on page 90

Security Considerations

NIS+ provides preset security defaults for the root domain. The default security level is level 2. Operational networks with actual users should always be run at security level 2. Security levels 0 and 1 are for configuring and testing purposes only. Do not run an operational network at level 0 or 1.

Note - The NIS+ security system is complex. If you are not familiar with NIS+ security, you might want to review the security-related chapters of *Solaris Naming Administration Guide* before starting to configure your NIS+ environment.

Prerequisites

Before proceeding, make sure that:

- The `/etc/passwd` file on the root master server contains an entry for you and every other administrator whose credentials will be added to the root domain in this configuration process.
- If the server will operate in NIS-compatibility mode and support DNS forwarding for Solaris 1.x release clients, it must have a properly configured `/etc/resolv.conf` file, as described in "The Resolver" on page 217.
- The server must have a unique machine name that duplicates all user IDs.
- The server must have a machine name that does not contain any dots. For example, a machine named `sales.alpha` is not allowed. A machine named `sales-alpha` is allowed.

Information You Need

In order to complete this task you need to know:

- The superuser password of the workstation that will become the root master server
- The name of the root domain
- The name of the root domain's admin group
- Your UID and password
- The UID of any administrator whose credentials you will add to the root domain

▼ How to Configure a Root Domain

1. **Log in as superuser on the machine designated to be the root master server.**

   The examples in these steps use `rootmaster` as the root master server and `doc.com.` as the root domain.

2. **Check the root master server's domain name.**

   Use the `domainname` command to make sure the root master server is using the correct domain name. The `domainname` command returns a workstation's current domain name.

Caution - Domains and hosts should not have the same name. For example, if you have a sales domain you should not have a machine named sales. Similarly, if you have a machine named home, you do not want to create a domain named home. This caution applies to subdomains; for example, if you have a machine named west, you don't want to create a sales.west.myco.com subdirectory.

If the name is not correct, change it.

```
rootmaster# domainname
strange.domain
rootmaster# domainname doc.com
rootmaster# domainname
rootmaster# doc.com
rootmaster# rm -f /etc/defaultdomain
rootmaster# domainname > /etc/defaultdomain
```

(Do not include a trailing dot with the domainname command. The domainname command is not an NIS+ command, so it does not follow the NIS+ conventions of a trailing dot.)

The above example changes the domain name of the root master server from strange.domain to doc.com. When changing or establishing a domain name, make sure that it has at least two elements; for example, doc.com instead of doc. The final element should end in either an Internet organizational name (such as .com) or a geographical identifier (such as .jp or .uk).

3. **Check the root master server's switch-configuration file.**

   Make sure the root master server is using the NIS+ version of the nsswitch.conf file, even if it will run in NIS-compatibility mode. This step ensures that the primary source of information for the root master are NIS+ tables.

```
rootmaster# more /etc/nsswitch.conf
```

   This command displays the current nsswitch.conf file. The primary name service referenced by this file should be nisplus. If the root master server's configuration file does not use nisplus as the primary name service, exchange it for one that does, as explained in "Selecting a Different Configuration File" on page 25.

4. **Optionally, configure the Diffie-Hellman key length.**

If you are using DES authentication, you can elect to increase the Diffie-Hellman key length from the default 192 bits. For example, to allow both 640 and 192–bit keys type the following:

```
rootmaster# nisauthconf dh640-0 des
```

5. **If you made any changes at all to the** `nsswitch.conf` **file, stop and restart the** `nscd` **daemon.**

   Because `nscd` caches the contents of the `nsswitch.conf` file, it is necessary to stop and restart `nscd` after any change to the switch file.

   Complete instructions are provided in Chapter 1.

6. **Now kill and restart** `keyserv`, **as shown below.**

```
rootmaster# cp /etc/nsswitch.nisplus /etc/nsswitch.conf
rootmaster# sh /etc/init.d/nscd stop
rootmaster# sh /etc/init.d/nscd start
rootmaster# ps -e | grep keyserv
 root 145 1 67 16:34:44 ? keyserv
 .
 .
rootmaster# kill -9 145
rootmaster# rm -f /etc/.rootkey
rootmaster# keyserv
```

7. **Clean out leftover NIS+ material and processes.**

   If the workstation you are working on was previously used as an NIS+ server or client, remove any files that might exist in `/var/nis` and kill the cache manager, if it is still running. In this example, a cold-start file and a directory cache file still exist in `/var/nis`:

```
rootmaster# ls /var/nis
NIS_COLD_START NIS_SHARED_CACHE
rootmaster# rm -rf /var/nis/*
rootmaster# ps -ef | grep nis_cachemgr
 root 295 260 10 15:26:58 pts/0 0:00 grep nis_cachemgr
 root 286 1 57 15:21:55 ? 0:01 /usr/sbin/nis_cachemgr
rootmaster# kill -9 286
```

   This step makes sure files left in `/var/nis` or directory objects stored by the cache manager are completely erased so they do not conflict with the new information generated during this configuration process. If you have stored any admin scripts in /var/nis, you might want to consider temporarily storing them elsewhere, until you finish setting up the root domain.

8. **Kill server daemons**

   If the workstation you are working on was previously used as an NIS+ server, check to see if rpc.nisd or rpc.nispasswdd is running. If either of these daemons is running, kill them.

9. **Name the root domain's admin group.**

   Although you won't actually create the admin group until Step 16 on page 99, you must identify it now. Identifying it now ensures that the root domain's org_dir directory object, groups_dir directory object, and all its table objects are assigned the proper default group when they are created in Step 14 on page 97.

   To name the admin group, set the value of the environment variable NIS_GROUP to the name of the root domain's admin group. Here are two examples, one for csh users, and one for sh/ksh users. They both set NIS_GROUP to admin.doc.com..

   For C Shell

```
rootmaster# setenv NIS_GROUP admin.doc.com.
```

   For Bourne or Korn Shell

```
rootmaster# NIS_GROUP=admin.doc.com.
rootmaster# export NIS_GROUP
```

10. **Create the root directory and initialize the root master server.**

    This step creates the first object in the namespace—the root directory—and converts the workstation into the root master server. Use the nisinit –r command, as shown below. (This is the only instance in which you will create a domain's directory object and initialize its master server in one step. In fact, nisinit –r performs an automatic nismkdir for the root directory. In any case, except the root master, these two processes are performed as separate tasks.)

```
rootmaster# nisinit -r
This machine is in the doc.com. NIS+ domain
Setting up root server ...
All done.
```

(continued)

(Continuation)

A UNIX directory with the name `/var/nis/data` is created.

Within the `/var/nis` directory is a file named `root.object.`

```
rootmaster# ls -l /var/nis/data
-rw-rw-rw- 1 root other 384 date root.object
```

This is not the root directory object; it is a file that NIS+ uses to describe the root of the namespace for internal purposes. The NIS+ root directory object is created in Step 11 on page 94 or Step 12 on page 95.

In subsequent steps, other files are added beneath the directory created in this step. Although you can verify the existence of these files by looking directly into the UNIX directory, NIS+ provides more appropriate commands. They are called out where applicable in the following steps.

Caution - Do not rename the `/var/nis` or `/var/nis/data` directories or any of the files in these directories that were created by `nisinit` or any of the other NIS+ configuration procedures. In Solaris Release 2.4 and earlier, the `/var/nis` directory contained two files named *hostname*. It also contained a subdirectory named `/var/nis/`*hostname*. In Solaris Release 2.5, the two files are named `trans.log` and `data.dict`, and the subdirectory is named `/var/nis/data`. In Solaris Release 2.5, the content of the files has also been changed and they are not backward compatible with Solaris Release 2.4 or earlier. Thus, if you rename either the directories or the files to match the Solaris Release 2.4 patterns, the files will not work with either the Solaris Release 2.4 or the Solaris Release 2.5 version of `rpc.nisd`. Therefore, you should not rename either the directories or the files.

---

11. [NIS-Compatibility only] Start the NIS+ daemon with `–Y`**.**

Perform this step only if you are setting up the root domain in NIS-compatibility mode; if setting up a standard NIS+ domain, perform Step 12 on page 95 instead. This step includes instructions for supporting the DNS forwarding capabilities of NIS clients.

Substep *a* starts the NIS+ daemon in NIS-compatibility mode. Substep *b* makes sure that when the server is rebooted, the NIS+ daemon restarts in

NIS-compatibility mode. After substep *b* of Step 11 on page 95, go to Step 14 on page 97.

a. **Use** `rpc.nisd` **with the** –Y**,** –B**, and** –S 0 **options.**

```
rootmaster# rpc.nisd -Y -B -S 0 options
```

The –Y option invokes an interface that answers NIS requests in addition to NIS+ requests. The –B option supports DNS forwarding. The –S 0 flag sets the server's security level to 0, which is required at this point for bootstrapping. Because no `cred` table exists yet, no NIS+ principals can have credentials; if you used a higher security level, you would be locked out of the server.

b. **Edit the** `/etc/init.d/rpc` **file.**

Search for the string `EMULYP=''Y''` in the `/etc/init.d/rpc` file. Uncomment the line and, to retain DNS forwarding capabilities, add the –B flag.

An `rpc` file with DNS forwarding contains:

```
EMULYP=''-Y -B''
```

An `rpc` file without DNS forwarding contains:

```
EMULYP=''-Y''
```

If you do not need to retain DNS forwarding capabilities, uncomment the line but do not add the –B flag.

12. **[Standard NIS+ only] Start the NIS+ daemon.**

Use the `rpc.nisd` and be sure to add the –S 0 flag.

```
rootmaster# rpc.nisd -S 0
```

The –S 0 flag sets the server's security level to 0, which is required at this point for bootstrapping. Because no cred table exists yet, no NIS+ principals can have credentials, and if used with a higher security level, you would be locked out of the server.

13. **Verify that the root objects have been properly created.**

As a result of Step 11 on page 94 or Step 12 on page 95, your namespace should now have:

- A root directory object (`root.dir`)
- A root master server (`rootmaster`) running the NIS+ daemon (`rpc.nisd`)
- A cold start file for the master server (`NIS_COLD_START`)
- A transaction log file (`trans.log`)

- A table dictionary file (`data.dict`).

The root directory object is stored in the directory created in Step 10 on page 93. Use the `ls` command to verify that it is there.

```
rootmaster# ls -l /var/nis/data
-rw-rw-rw- 1 root other 384 date root.object
-rw-rw-rw- 1 root other 124 date root.dir
```

At this point, the root directory is empty; in other words, it has no subdirectories. You can verify this by using the `nisls` command.

```
rootmaster# nisls -l doc.com.
doc.com.:
```

However, it has several *object* properties, which you can examine using `niscat –o`:

```
rootmaster# niscat -o doc.com.
 Object Name : doc
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : Com.
Access Rights : r---rmcdrmcdr---
```

Notice that the root directory object provides full (read, modify, create, and destroy) rights to both the owner and the group, while providing only read access to the world and nobody classes. (If your directory object does not provide these rights, you can change them using the `nischmod` command.)

To verify that the NIS+ daemon is running, use the `ps` command.

```
rootmaster# ps -ef | grep rpc.nisd
root 1081 1 61 16:43:33 ? 0:01 rpc.nisd -S 0
root 1087 1004 11 16:44:09 pts/1 0:00 grep rpc.nisd
```

The root domain's NIS_COLD_START file, which contains the Internet address (and, eventually, public keys) of the root master server, is placed in /var/nis.

Although there is no NIS+ command that you can use to examine its contents, its contents are loaded into the server's directory cache (NIS_SHARED_DIRCACHE). You can examine those contents with the /usr/lib/nis/nisshowcache command.

Also created are a transaction log file (trans.log) and a dictionary file (data.dict). The transaction log of a master server stores all the transactions performed by the master server and all its replicas since the last update. You can examine its contents by using the nislog command. The dictionary file is used by NIS+ for internal purposes; it is of no interest to an administrator.

14. **Create the root domain's subdirectories and tables.**

This step adds the org_dir and groups_dir directories, and the NIS+ tables, beneath the root directory object. Use the nissetup utility. For an NIS-compatible domain, be sure to include the –Y flag. Here are examples for both versions:

For standard NIS+ only

```
rootmaster# /usr/lib/nis/nissetup
```

NIS-compatible only

```
rootmaster# /usr/lib/nis/nissetup -Y
```

Each object added by the utility is listed in the output:

```
rootmaster# /usr/lib/nis/nissetup
org_dir.doc.com. created
groups_dir.doc.com. created
auto_master.org_dir.doc.com. created
auto_home.org_dir.doc.com. created
bootparams.org_dir.doc.com. created
cred.org_dir.doc.com. created
ethers.org_dir.doc.com. created
group.org_dir.doc.com. created
hosts.org_dir.doc.com. created
mail_aliases.org_dir.doc.com. created
sendmailvars.org_dir.doc.com. created
netmasks.org_dir.doc.com. created
netgroup.org_dir.doc.com. created
```

(continued)

(Continuation)

```
networks.org_dir.doc.com. created
passwd.org_dir.doc.com. created
protocols.org_dir.doc.com. created
rpc.org_dir.doc.com. created
services.org_dir.doc.com. created
timezone.org_dir.doc.com. created
```

The –Y option creates the same tables and subdirectories as for a standard NIS+ domain, but assigns read rights to the `passwd` table to the `nobody` class so that requests from NIS clients, which are unauthenticated, can access the encrypted password in that column.

Recall that when you examined the contents of the root directory with `nisls` (in Step 12 on page 95), it was empty. Now, however, it has two subdirectories.

```
rootmaster# nisls doc.com.
doc.com.:
org_dir
groups_dir
```

You can examine the object properties of the subdirectories and tables by using the `niscat –o` command. You can also use the `niscat` option without a flag to examine the information in the tables, although at this point they are empty.

15. Create DES credentials for the root master server.

The root master server requires DES credentials so that its own requests can be authenticated. To create those credentials, use the `nisaddcred` command, as shown below. When prompted, enter the server's root password.

```
rootmaster# nisaddcred des
DES principal name: unix.rootmaster@doc.com
Adding key pair for unix.rootmaster@doc.com
 (rootmaster.doc.com.).
Enter login password:
Wrote secret key into /etc/.rootkey
```

If you enter a password that is different from the server's root password, you receive a warning message and a prompt to repeat the password:

```
Enter login password:
nisaddcred: WARNING: password differs from login password.
Retype password:
```

If you persist and retype the same password, NIS+ will still create the credential. The new password will be stored in /etc/.rootkey and be used by the keyserver when it starts up. To give the keyserver the new password right away, run keylogin –r, as described in the credentials chapter of *Solaris Naming Administration Guide.*

If you decide to use your login password after all, press Control-c and start the sequence over. If you were to retype your login password as encouraged by the server, you would get an error message designed for another purpose, but which in this instance could be confusing.

```
nisaddcred: WARNING: password differs from login password.
Retype password:
nisaddcred: password incorrect.
nisaddcred: unable to create credential.
```

As a result of this step, the root server's private and public keys are stored in the root domain's cred table (cred.org_dir.doc.com.) and its secret key is stored in /etc/.rootkey. You can verify the existence of its credentials in the cred table by using the niscat command. Since the default domain name is doc.com., you don't have to enter the cred table's fully qualified name; the org_dir suffix is enough. You can locate the root master's credential by looking for its secure RPC netname.

16. Create the root domain's admin group.

This step creates the admin group named in Step 9 on page 93. Use the nisgrpadm command with the –c option. The example below creates the admin.doc.com. group.

```
rootmaster# nisgrpadm -c admin.doc.com.
 Group admin.doc.com. created.
```

This step only creates the group—it does not identify its members. That is done in Step 17 on page 100. To observe the object properties of the group, use `niscat –o`, but be sure to append `groups_dir` in the group's name.

```
doc.com.
Object Name : admin
Directory : groups_dir.doc.com
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : groups_dir.doc.com.
Access Rights : ----rmcdr---r---
Time to Live : 1:0:0
Object Type : GROUP
Group Flags :
Group Members :
```

17. Add the root master to the root domain's admin group.

Since at this point the root master server is the only NIS+ principal that has DES credentials, it is the only member you should add to the admin group. Use the `nisgrpadm` command again, but with the `–a` option. The first argument is the group name, the second is the name of the root master server. This example adds `rootmaster. doc.com.` to the `doc.com` domain.

```
rootmaster# nisgrpadm -a admin.doc.com.
 rootmaster.doc.com.
Added rootmaster.doc.com. to group admin.doc.com.
```

To verify that the root master is indeed a member of the group, use the `nisgrpadm` command with the `–l` option (see the groups chapter of *Solaris Naming Administration Guide*).

Note - With group-related commands such as `nisgrpadm`, you don't have to include the `groups_dir` subdirectory in the name. You need to include that directory with commands like `niscat` because they are designed to work on NIS+ objects in general. The group-related commands are "targeted" at the `groups_dir` subdirectory.

```
rootmaster# nisgrpadm -l admin.doc.com.
 Group entry for admin.doc.com. group:
 Explicit members:
 rootmaster.doc.com.
 No implicit members
 No recursive members
 No explicit nonmembers
 No implicit nonmembers
 No recursive nonmembers
```

18. Update the root domain’s public keys.

Normally, directory objects are created by an NIS+ principal that already has DES credentials. In this case, however, the root master server could not acquire DES credentials until *after* it created the `cred` table (since there was no parent domain in which to store its credentials). As a result, three directory objects—`root`, `org_dir`, and `groups_dir`—do not have a copy of the root master server’s public key. (You can verify this by using the `niscat –o` command with any of the directory objects. Look for the public key field. Instructions are provided in the directories chapter of *Solaris Naming Administration Guide.*)

To propagate the root master server’s public key from the root domain’s cred table to those three directory objects, use the `/usr/lib/nis/nisupdkeys` utility for each directory object.

```
rootmaster# /usr/lib/nis/nisupdkeys doc.com.
rootmaster# /usr/lib/nis/nisupdkeys org_dir.doc.com.
rootmaster# /usr/lib/nis/nisupdkeys groups_dir.doc.com.
```

After each instance, you will receive a confirmation message such as this one:

```
Fetch Public key for server rootmaster.doc.com.
 netname = ’unix.rootmaster@doc.com.’
Updating rootmaster.doc.com.’s public key.
 Public key:
```

If you look in any of those directories (use `niscat –o`), you can find one or more entries like the following in the public key field:

```
Public key: Diffie-Hellman (192 bits)
```

19. Start the NIS+ cache manager.

The cache manager maintains a local cache of location information for an NIS+ client (in this case, the root master server). It obtains its initial set of information from the client's cold-start file (created in Step 11 on page 94 or Step 12 on page 95), and downloads it into a file named NIS_SHARED_DIRCACHE in /var/nis.

To start the cache manager, enter the nis_cachemgr command as shown below.

```
rootmaster# nis_cachemgr
```

After the cache manager has been started, you have to restart it only if you have explicitly killed it. You don't have to restart it if you reboot, since the NIS_COLD_START file in /var/nis starts it automatically when the client is rebooted. For more information about the NIS+ cache manager, see the directories chapter of *Solaris Naming Administration Guide*.

20. Restart the NIS+ daemon with security level 2.

Now that the root master server has DES credentials and the root directory object has a copy of the root master's public key, you can restart the root master with security level 2. First kill the existing daemon, then restart with security level 2.

For a standard NIS+ domain only:

```
rootmaster# ps -e | grep rpc.nisd
1081 ? 0:03 rpc.nisd -s 0
rootmaster# kill 1081
rootmaster# rpc.nisd
```

For an NIS-compatible root domain, be sure to use the –Y (and –B) flags:

For a NIS-compatible NIS+ domain:

```
rootmaster# ps -e | grep rpc.nisd
1081 ? 0:03 rpc.nisd -Y -B -s 0
rootmaster# kill 1081
rootmaster# rpc.nisd -Y -B
```

Since security level 2 is the default, you don't need to use an –S 2 flag.

Note - Operational networks with actual users should always be run at security level 2. Security levels 0 and 1 are for configuration and testing purposes only. Do not run an operational network at level 0 or 1.

21. Add your LOCAL credentials to the root domain.

Because you don’t have access rights to the root domain’s cred table, you must perform this operation as superuser. In addition, the root master’s /etc/passwd file must contain an entry for you. Use the nisaddcred command with the –p and –P flags as shown below.

```
nisaddcred -p uid -P principal-name local
```

The *principal-name* consists of the administrator’s login name and domain name. This example adds a LOCAL credential for an administrator with a UID of 11177 and an NIS+ principal name of topadmin.doc.com.

```
rootmaster# nisaddcred -p 11177 -P topadmin.doc.com. local
```

For more information about the nisaddcred command, see the credentials chapter of *Solaris Naming Administration Guide.*

22. Add your DES credentials to the root domain.

Use the nisaddcred command again, but with the following syntax:

```
nisaddcred -p secure-RPC-netname- P principal-name des
```

The *secure-RPC-netname* consists of the prefix unix followed by your UID, the symbol @, and your domain name, but *without* a trailing dot. The *principal-name* is the same as for LOCAL credentials: your login name followed by your domain name, *with* a trailing dot.

```
rootmaster# nisaddcred -p unix.11177@doc.com -P topadmin.doc.com. des
Adding key pair for unix.11177@doc.com (topadmin.doc.com.).
Enter login password:
```

If, after entering your login password, you get a password that differs from the login password warning, yet the password you entered is your correct login password, ignore the error message. The message appears because NIS+ cannot read the protected /etc/shadow file that stores the password, as expected. The message would not have appeared if you had no user password information stored in the /etc/passwd file.

23. Add credentials for the other administrators.

Add the credentials, both LOCAL and DES, of the other administrators who will work in the root domain. You can do this in the following ways.

- An easy way to create temporary credentials for the other administrators is to use Solstice AdminSuite (if you have it available) running in NIS+ mode.
- A second way is to ask them to add their own credentials. However, they will have to do this as superuser. Here is an example that adds credentials for an administrator with a UID of `33355` and a principal name of `miyoko.doc.com.`

```
rootmaster# nisaddcred -p 33355 -P miyoko.doc.com. local
rootmaster# nisaddcred -p unix.33355@doc.com -P miyoko.doc.com. des
Adding key pair for unix.33355@doc.com (miyoko.doc.com.).
 Enter login password:
```

- A third way is for you to create temporary credentials for the other administrators, using dummy passwords. (Note that the other administrator, in this example `miyoko`, must have an entry in the NIS+ `passwd` table. If no such entry exists, you must first create one with `nistbladm`. The example below includes that step.)

```
rootmaster# nistbladm -D owner=miyoko.doc.com. name=miyoko uid=33355 gcos=miyoko
home=/home/miyoko shell=/bin/tcsh passwd.org_dir
rootmaster# nisaddent -a -f /etc/passwd.xfr passwd
rootmaster# nisaddent -a -f /etc/shadow.xfr shadow
rootmaster# nisaddcred -p 33355 -P miyoko.doc.com. local
rootmaster# nisaddcred -p unix.33355@doc.com -P miyoko.doc.com. des
Adding key pair for unix.33355@doc.com (miyoko.doc.com.).
Enter miyoko's login password:
nisaddcred: WARNING: password differs from login passwd.
Retype password:
rootmaster# nischown miyoko.doc.com. '[name=miyoko],passwd.org_dir'
```

In this case, the first instance of `nisaddent` populates the `passwd` table—except for the password column. The second instance populates the shadow column. Each administrator can later change his or her network password using the `chkey` command. The credentials chapter of *Solaris Naming Administration Guide* describes how to do this.

24. Add yourself and other administrators to the root domain's admin group.

You don't have to wait for the other administrators to change their dummy passwords to perform this step. Use the `nisgrpadm` command with the `–a` option. The first argument is the group name, the remaining arguments are the names of the administrators. This example adds two administrators, `topadmin` and `miyoko`, to the `admin.doc.com.` group:

```
rootmaster# nisgrpadm -a admin.doc.com. topadmin.doc.com. miyoko.doc.com.
Added topadmin.doc.com. to group admin.doc.com.
Added miyoko.doc.com. to group admin.doc.com.
```

25. Allocate sufficient swap space to accommodate NIS+ tables.

Swap space should be double the size of the maximum size of rpc.nisd. To determine how much memory rpc.nisd is using, issue the following command:

```
rootmaster# /usr/lib/nis/nisstat
```

rpc.nisd will under certain circumstances fork a copy of itself. If there is not enough memory, rpc.nisd fails.

You can also calculate the memory and swap space requirements for NIS+ tables. For example, if you have 180,000 users and 180,000 hosts in your NIS+ tables, those two tables occupy approximately 190 Mbytes of memory. When you add credentials for 180,000 users and 180,000 hosts, the cred table has 540,000 entries (one entry for each local user credential, one entry for each DES user credential, and one entry for each host). The cred table occupies approximately 285 Mbytes of memory. In this example, rpc.nisd occupies at least 190 Mbytes + 285 Mbytes = 475 Mbytes of memory. So, you will require at least 1 Gbyte swap space. You will also want at least 500 Mbytes of memory to hold rpc.nisd entirely in memory.

Root Domain Configuration Summary

Table 5–2 summarizes the steps required to configure a root domain. The summary assumes a simple case. Be sure you are familiar with the complete task descriptions before you use this summary as a reference. This summary does not show the server's responses to each command.

TABLE 5–2 Setting Up a Root Domain: Command Summary

Tasks	Commands
Log in as superuser to rootmaster.	rootmaster% su Password:
Check domain name	# domainname
Check Switch file.	# more /etc/nsswitch.conf
Remove leftover NIS+ material.	# rm -rf /var/nis*
Name the admin group.	# NIS_GROUP=admin.doc.com.; export NIS_GROUP
Initialize the root master.	# nisinit -r
[NIS-compat only] Start daemon with –Y -B, S 0. Change to EMULYP=-Y -B.	# rpc.nisd -Y -B -S 0 # vi /etc/inet.d/rpc
[NIS+ Only] Start daemon with –S 0.	# rpc.nisd -S 0
Create org_dir and groups_dir tables.	# /usr/lib/nis/nissetup [-Y]
Create DES credentials for root master.	#nisaddcred des Enter login password:
Create admin group.	# nisgrpadm -c admin.doc.com.
Assign full group rights to root directory	# nischmod g+rmcd doc.com.
Add rootmaster to admin group.	# nisgrpadm -a admin.doc.com. rootmaster.doc.com.
Update root directory's keys. Update org_dir's keys. Update groups_dir's keys.	# /usr/lib/nis/nisupdkeys doc.com. # /usr/lib/nis/nisupdkeys org_dir.doc.com. # /usr/lib/nis/nisupdkeys groups_dir.doc.com.
Start NIS+ cache manager	# nis_cachemgr

TABLE 5–2 Setting Up a Root Domain: Command Summary *(continued)*

Tasks	Commands
Kill existing daemon.	`# ps -ef | grep rpc.nisd` `# kill -9` *process-id*
Restart the NIS+ daemon. (Use –Y for NIS compat and –B for DNS.)	`# rpc.nisd [-Y] [-B]`
Add your LOCAL credentials.	`# nisaddcred -p 11177 -P topadmin.doc.com. local`
Add your DES credentials.	`# nisaddcred -p unix.11177@doc.com -P topadmin.doc.com. des Enter login password:`
Add credentials for other admins. Add other admins to `admin` group.	`# nisaddcred ... nisgrpadm -a admin.doc.com` *members*

CHAPTER **6**

Configuring NIS+ Clients

This chapter gives step-by-step instructions for setting up NIS+ clients using the NIS+ command set and three different initialization methods. These instructions apply to clients in both the root domain and subdomains, whether all-NIS+ or NIS-compatible.

Introduction to Setting Up NIS+ Clients

This chapter describes how to configure clients in both standard NIS+ domains and NIS-compatible domains. The procedure describes each step in detail and provides related information. For those who do not need detailed instructions, a summary listing of the necessary commands is provided in Table 6–6.

Note - It is much easier to perform this task with the NIS+ installation scripts, as described in Part 1, than with the NIS+ command set as described here. The methods described in this chapter should only be used by those administrators who are very familiar with NIS+ and who require some non-standard features or configurations not provided by the installation scripts. If you have them available, the Solstice AdminSuite™ tools also provide easier methods of adding and setting up NIS+ client machines.

At Step 10 on page 114 in the client configuration instructions you must choose which of three methods to use: broadcast, host name, or cold-start file. Because each method is implemented differently, each has its own task description. After initializing a client by one of these methods, you can continue setting up the client by returning to Step 11 on page 114.

The last task in the chapter describes how to change a workstation's domain.

Configuring the Client

This section describes how to configure a typical NIS+ client in either the root domain or a non-root domain. This procedure applies to regular NIS+ clients and to those clients that will later become NIS+ servers. It applies, as well, to clients in a standard NIS+ domain and those in an NIS-compatible domain.

Caution - Domains and hosts should not have the same name. For example, if you have a `sales` domain you should not have a machine named `sales`. Similarly, if you have a machine named `home`, you do not want to create a domain named home. This caution applies to subdomains; for example, if you have a machine named `west` you do not want to create a `sales.west.myco.com` subdomain.

Setting up an NIS+ client involves the following tasks:

- Creating credentials for the client
- Preparing the workstation
- Initializing the workstation as an NIS+ client.

However, as with setting up the root domain, setting up a client is not as simple as carrying out these three tasks in order. To make the configuration process easier to execute, these tasks have been broken down into individual steps, and the steps have been arranged in the most efficient order:

1. Logging in to the domain's master server
2. Creating DES credentials for the new client workstation
3. Ascertaining the Diffie-Hellman key length used on the master server
4. Logging in as superuser to the client
5. Assigning the client its new domain name
6. Stopping and restarting `nscd`
7. Checking the client's `nsswitch.conf` file
8. Setting the client's Diffie-Hellman key
9. Cleaning out leftover NIS+ material and processes

10. Initializing the client
11. Killing and restarting the keyserv daemon
12. Running keylogin
13. Rebooting the client

Security Considerations

Setting up a client has two main security requirements: both the administrator and the client must have the proper credentials and access rights. Otherwise, the only way for a client to obtain credentials in a domain running at security level 2 is for the credentials to be created by an administrator with valid DES credentials and modify rights to the cred table in the client's home domain. The administrator can either have DES credentials in the client's home domain or in the administrator's home domain.

After an administrator creates the client's credentials, the client can complete the configuration process. However, the client still needs read access to the directory object of its home domain. If you configured the client's home domain according to the instructions in either Chapter 5 or Chapter 8, read access was provided to the world class by the NIS+ commands used to create the directory objects (nisinit and nismkdir, respectively).

You can check the directory object's access rights by using the niscat–o command. This command displays the properties of the directory, including its access rights:

```
rootmaster# niscat -o doc.com.
ObjectName : Doc
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : Com.
Access Rights : r---rmcdr---r---
```

You can change the directory object's access rights, provided you have modify rights to it yourself, by using the nischmod command, described in the rights chapter of *Solaris Naming Administration Guide.*

Prerequisites

The administrator setting up the client's credentials must have:

- A valid DES credential
- Modify rights to the cred table in the client's home domain

The client must have:

- Read rights to the directory object of its home domain

- The client's home domain must already be configured and running NIS+
- An entry in either the master server's `/etc/hosts` or `/etc/inet/ipnodes` file or in its domain's hosts or ipnodes table
- A unique machine name that does duplicate any user ID
- A machine name that does not contain any dots. (For example, a machine named `sales.alpha` is not allowed; a machine named `sales-alpha` is allowed.)

Information You Need

- The name of the client's home domain
- The superuser password of the workstation that will become the client
- The IP address of an NIS+ server in the client's home domain

Configuring the Client—Task Map

TABLE 6–1 Configuring the Client

Task	Description	For Instructions, Go To
Configuring the Client"	Create credentials fpr the client. Prepare the client workstation and initialize it as an NIS+ client.	"How to Configure an NIS+ Client" on page 112

▼ How to Configure an NIS+ Client

1. **Log into the domain's master server.**

   You can log in as superuser or as yourself, depending on which NIS+ principal has the proper access rights to add credentials to the domain's cred table.

2. **Create DES credentials for the new client workstation.**

   Use the `nisaddcred` command with the `–p` and `–P` arguments. Here is the syntax:

   ```
   nisaddcred -p secure-RPC-netname  principal-name des [domain]
   ```

   The secure-RPC-netname consists of the prefix `unix` followed by the client's host name, the symbol `@` and the client's domain name, but without a trailing dot. The *principal-name* consists of the client's host name and domain name, with a trailing dot. If the client belongs to a different domain than the server from which you enter the command, append the client’s domain name after the second argument.

This example adds a DES credential for a client workstation named `client1` in the `doc.com.` domain:

```
rootmaster% nisaddcred -p unix.client1@doc.com -P client1.doc.com. des
Adding key pair for unix.client1@doc.com (client1.doc.com.).
Enter client1.doc.com.’s root login passwd:
Retype password:
```

For more information about the `nisaddcred` command, see the credentials chapter of the *Solaris Naming Administration Guide*.

3. **Ascertain the Diffie-Hellman key length used on the master server.**

   For example:

   ```
   rootmaster% nisauthconf dh640-0 des
   ```

4. **Log in as superuser to the client.**

   Now that the client workstation has credentials, you can log out of the master server and begin working from the client itself. You can do this locally or remotely.

5. **Assign the client its new domain name.**

   See “Changing a Workstation’s Domain Name” on page 115 for information on how to assign (or change) a client’s domain name, then return to Step 6 on page 113.

6. **Check the client’s `nsswitch.conf` file.**

   Make sure the client is using a NIS+ version of the `nsswitch.conf` file. This ensures that the primary source of information for the client will be NIS+ tables. See “Default NIS+ Version of Switch File” on page 22 for a description of a NIS+ switch file.

7. **If you made any changes to the `nsswitch.conf` file (or copied over a new file), you must now stop and restart `nscd`, as shown below.**

```
client1# cp /etc/nsswitch.nisplus /etc/nsswitch.conf
client1# sh /etc/init.d/nscd stop
client1# sh /etc/init.d/nscd start
```

(Although the instructions in Chapter 1 tell you to kill and restart the keyserver at this point, you do not need to do that here, since you will do so in Step 11 on page 114.)

8. **Set the Diffie-Hellman key length on the client, using the information from step 3.**

   For example:

   ```
   client# nisauthconf dh640-0 des
   ```

9. **Clean out leftover NIS+ material and processes.**

   If the workstation you are working on was previously used as an NIS+ server or client, remove any files that might exist in `/var/nis` and kill the cache manager, if it is still running. In this example, a cold-start file and a directory cache file still exist in `/var/nis`.

```
client1# ls /var/nis
NIS_COLD_START NIS_SHARED_CACHE
client1# rm -rf /var/nis/*
client1# ps -ef | grep nis_cachemgr
 root 295 260 10 15:26:58 pts/0 0:00 grep nis_cachemgr
 root 286 1 57 15:21:55 ? 0:01 /usr/sbin/nis_cachemgr
client1# kill -9 286
```

   This step makes sure that files left in `/var/nis` or directory objects stored by the cache manager are completely erased so that they do not conflict with the new information generated during this configuration process. If you have stored any admin scripts in `/var/nis`, you might want to consider temporarily storing them elsewhere, until you finish setting up the root domain.

10. **Initialize the client.**

    You can initialize a client in three different ways: by host name, by cold-start file, or by broadcast. Choose and perform one of those methods. After initializing the client, proceed with Step 11 on page 114.

11. **Kill and restart the** `keyserv` **daemon.**

    This step stores the client’s secret key on the keyserver.

    a. **Kill the** `keyserv` **daemon.**

       This also has the side effect of updating the key server’s switch information about the client.

    b. **Remove the** `/etc/.rootkey` **file.**

c. **Restart the keyserver.**

This example shows the complete procedure in Step 11 on page 114.

```
client1# ps -e | grep keyserv
 root 145 1 67 16:34:44 ? keyserv
client1# kill 145
client1# rm -f /etc/.rootkey
client1# keyserv
```

d. **Run `keylogin–r`.**

This step stores the client's secret key with the keyserver. It also saves a copy in `/etc/.rootkey`, so that the superuser on the client does not have to run `keylogin` to use NIS+. Use `keylogin` with the `–r` option. When prompted for a password, type the client's superuser password. It must be the same as the password supplied to create the client's DES credentials:

```
client1# keylogin -r
Password:
Wrote secret key into /etc/.rootkey
```

e. **Reboot the client.**

Changing a Workstation's Domain Name

This task changes a workstation's domain name. Since a workstation's domain name is usually set during installation, you should check it (type `domainname` without an argument) before you decide to perform this task.

Security Considerations

You must perform this task as superuser on the workstation whose domain name you are changing.

Information You Need

- The workstation's superuser password
- The new domain name

Changing a Workstation's Domain—Task Map

TABLE 6–2 Configuring the Client

Task	Description	For Instructions, Go To
Changing a Workstation's Domain	Use the `domainname` command to change the client workstation domain	"How to Change a Client's Domain Name" on page 116

▼ How to Change a Client's Domain Name

1. Log in to the workstation and become superuser.

The examples in this task use `client1` as the workstation and `doc.com.` as the new domain name.

```
client1% su
Password:
```

2. Change the workstation's domain name.

Type the new name after the `domainname` command. Do not use a trailing dot. For example, to change a workstation's domain to the `doc.com` domain, you enter:

```
client1# domainname doc.com
```

If the workstation had been an NIS client, it may no longer be able to get NIS service.

3. **Verify the result.**

   Run the domainname command again, this time without an argument, to display the server's current domain.

```
client1# domainname
doc.com
```

4. **Save the new domain name.**

   Redirect the output of the domainname command into the /etc/defaultdomain file.

```
client1# domainname > /etc/defaultdomain
```

5. **At a convenient time, reboot the workstation.**

   Even after entering the new domain name into the /etc/defaultdomain file, some processes may still operate with the old domain name. To ensure that all processes are using the new domain name, reboot the workstation.

   Because you may be performing this task in a sequence of many other tasks, examine the work remaining to be done on the workstation before rebooting. Otherwise, you might find yourself rebooting several times instead of just once.

   Although restarting individual daemons, such as mountd may solve an NFS problem, it is strongly recommended that you reboot to synchronize configuration changes across daemons. This minimizes application failures caused by unknown changes to the configuration.

Initializing an NIS+ Client

The three different ways to initialize a NIS+ client are:

- Broadcast method (see "Broadcast Initialization" on page 118)
- Host-name method (see "Initializing a Client by Host Name" on page 119)
- Cold-start file method (see "Initializing Client Using a Cold-Start File" on page 120)

Broadcast Initialization

This method *initializes* an NIS+ client by sending an IP broadcast on the client's subnet.

This is the simplest way to configure a client but is also the least secure. The NIS+ server that responds to the broadcast sends the client all the information that the client needs in its cold-start file, including the server's public key. Presumably, only an NIS+ server will respond to the broadcast. However, the client has no way of knowing whether the workstation that responded to the broadcast is indeed a trusted server. As a result, this method is only recommended for sites with small, secure networks.

Security Considerations

You must perform this task as superuser on the client.

Prerequisites

At least one NIS+ server must exist on the same subnet as the client. The client must use the same Diffie-Hellman key lengths as those on the master server. See `nisauthconf`(1M).

Information You Need

You need the superuser password to the client.

Initializing an NIS+ Client—Task Map

TABLE 6–3 Initializing an NIS+ Client

Task	Description	For Instructions, Go To
Initializing an NIS+ Client	Use `nisclient` command to initialize an NIS+ client	"How to Initialize a Client—Broadcast Method" on page 119

How to Initialize a Client—Broadcast Method

♦ **Initialize the client.**

This step initializes the client and creates a `NIS_COLD_START` file in its `/var/nis` directory. Use the `nisinit` command with the –`c` and –`B` options.

```
client1# nisinit -c -B
This machine is in the doc.com. NIS+ domain.
Setting up NIS+ client ...
All done.
```

An NIS+ server on the same subnet will reply to the broadcast and add its location information into the client's cold-start file.

Initializing a Client by Host Name

Initializing a client by host name consists of explicitly identifying the IP address of its trusted server. This server's name, location information, and public keys are then placed in the client's cold-start file.

This method is more secure than the broadcast method because it actually specifies the IP address of the trusted server, rather than relying on a server to identify itself. However, if a router exists between the client and the trusted server, it could intercept messages to the trusted IP address and route them to an untrusted server.

Security Considerations

You must perform this operation as superuser on the client.

Prerequisites

- The NIS+ service must be running in the client's domain.
- The client must have an entry in its `/etc/hosts` or `/etc/inet/ipnodes` file for the trusted server.
- The client must use the same Diffie-Hellman key lengths as those on the master server. See `nisauthconf`(1M).

Information You Need

You need the name and IP address of the trusted server.

Initializing an NIS+ Client—Task Map

TABLE 6–4 Initializing an NIS+ Client

Task	Description	For Instructions, Go To
Initializing a Client by Host Name	Use `nisinit` command to initialize an NIS+ client by host name.	"How to Initialize a Client—Host-name Method" on page 120

How to Initialize a Client—Host-name Method

1. **Check the client's** `/etc/hosts` **or** `/etc/inet/ipnodes` **file.**

   Make sure the client has an entry for the trusted server.

2. **Initialize the client.**

   This step initializes the client and creates a `NIS_COLD_START` file in its `/var/nis` directory. Use the `nisinit` command with the –`c` and –`H` options. This example uses `rootmaster` as the trusted server.

```
Client1# nisinit -c -H rootmaster
This machine is in the doc.com. NIS+ domain.
Setting up NIS+ client ...
All done.
```

   The `nisinit` utility looks for the server's address in the client's `/etc/hosts` or `/etc/inet/ipnodes` file, so do not append a domain name to the server. If you do, the utility will not be able to find its address.

Initializing Client Using a Cold-Start File

This task initializes an NIS+ client by using the cold-start file of another NIS+ client, preferably one from the same domain. This is the most secure method of setting up an NIS+ client. It ensures that the client obtains its NIS+ information from a trusted server, something that cannot be guaranteed by the host-name or broadcast method.

Security Considerations

You must perform this task as superuser on the client.

Prerequisites

The servers specified in the cold-start file must already be configured and running NIS+.

The client must use the same Diffie-Hellman key lengths as those on the master server. See `nisauthconf`(1M).

Information You Need

You need the name and location of the cold-start file you will copy.

Initializing an NIS+ Client—Task Map

TABLE 6–5 Initializing an NIS+ Client

Task	Description	For Instructions, Go To
InitializingClient via Cold-Start File	Use `nisinit` command to initialize an NIS+ client via a cold-start file	"How to Initialize a Client—Cold-Start Method" on page 121

How to Initialize a Client—Cold-Start Method

1. **Copy the other client's cold-start file.**

   Copy the other client's cold-start file into a directory in the new client. This may be easier to do while logged on as yourself rather than as superuser on the client. Be sure to switch back to superuser before initializing the client.

   Don't copy the `NIS_COLD_START` file into `/var/nis`, because that file gets overwritten during initialization. This example copies the cold-start file of previously initialized `client1` into the `/tmp` directory of uninitialized `client2`.

```
client2# exit
client2% rcp client1:/var/nis/NIS_COLD_START /tmp
```

(Continuation)

```
client2% su
```

2. **Initialize the client from the cold-start file.**

   Use the `nisinit` command with the `–c` and `–C` options.

```
client2# nisinit -c  -C /tmp/NIS_COLD_START
This machine is in the doc.com. NIS+ domain.
Setting up NIS+ client ...
All done.
```

NIS+ Client Configuration Summary

Table 6–6 shows a summary of the steps required to configure a client named `client1` in the `doc.com` domain. It assumes the simplest case, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. For the sake of brevity, this summary does not show the responses to each command.

TABLE 6–6 Setting Up a Client: Command Summary

Tasks	**Commands**
Log in to domain's master.	`rootmaster%`
Create DES credentials for client.	`rootmaster% nisaddcred -p unix.client1.doc.com -P client1.doc.com. des`
Ascertain the Diffie-Hellman .key length.	`rootmaster% nisauthconf`

TABLE 6–6 Setting Up a Client: Command Summary *(continued)*

Tasks	Commands

Log in, as superuser, to the client.	`client1%` `su`
	`Password:`

Assign the client a domain name.	`client1#` `domainname doc.com`
	`client1#` `domainname > /etc/defaultdomain`

Check that the client's switch configuration file has the correct settings.	`client1#` `more /etc/nsswitch.conf`
Set the Diffie-Hellman key length	`client1#` `nisauthconf dh640-0 des`
Clean out `/var/nis`.	`client1#` `rm -rf /var/nis/*`
Initialize the client.	`client1#` `nisinit -c -H rootmaster`

Kill and restart the keyserver.	`client1#` `ps -ef | grep keyserv`
	`client1#` `kill -9` *process-id*
	`client1#` `keyserv`

Run `keylogin` on the client.	`client1#` `keylogin -r password:`
Reboot the client.	`client1#` `init 6`

CHAPTER **7**

Configuring NIS+ Servers

This chapter provides step-by-step procedures for using the NIS+ commands to set up NIS+ servers (except the root master) and add replica servers to existing NIS+ domains.

- "Setting Up an NIS+ Server" on page 125
- "Adding a Replica to an Existing Domain" on page 129

This section applies to any NIS+ server except the root master; that is, it applies to root replicas, non-root masters, and non-root replicas, whether running in NIS-compatibility mode or not. A summary of each task is provided at the end of the chapter.

See *Solaris Naming Administration Guide* for information on how to take an existing server out of service and replace it with a new machine.

Setting Up an NIS+ Server

It is much easier to perform this task with the NIS+ installation scripts, as described in Part 1, than with the NIS+ command set described here. The methods described in this chapter should be used only by those administrators who are very familiar with NIS+ and who require some nonstandard features or configurations not provided by the installation scripts.

Standard Versus NIS-Compatible Configuration Procedures

The differences between setting up an NIS-compatible and a standard NIS+ server are the same as the differences between setting up standard and NIS-compatible root master servers (see "Standard Versus NIS-Compatible Configuration Procedures" on page 88). The NIS+ daemon for an NIS-compatible server must be started with the –Y option (and the –B option for DNS forwarding), which allows the server to answer requests from NIS clients. This is described in Step 2 on page 127 (the equivalent step for standard NIS+ servers is Step 3 on page 128).

Note - Whenever `rpc.nisd` is started with either the –Y or –B option, a secondary daemon named `rpc.nisd_resolv` is spawned to provide name resolution. This secondary daemon must be separately killed whenever you kill the primary `rpc.nisd` daemon.

Here is a summary of the entire configuration process:

1. Log in as superuser to the new replica server.
2. [NIS-Compatibility Only] Start the NIS+ daemon with –Y.
3. [Standard NIS+ Only] Start the NIS+ daemon.

Security Considerations

Note - The NIS+ security system is complex. If you are not familiar with NIS+ security, you might want to review the security-related chapters of *Solaris Naming Administration Guide* before starting to configure your NIS+ environment.

The security level at which you start the server determines the credentials that its clients must have. For instance, if the server is configured with security level 2 (the default), the clients in the domain it supports must have DES credentials. If you have configured the client according to the instructions in this book, the client has DES credentials in the proper domain, and you can start the server with security level 2.

Note - Security level 0 is for administrator configuration and testing purposes only. Security level 1 is not supported. Do not use level 0 or 1 in any environment where ordinary users are doing their normal work. Operating networks should always be run at security level 2.

Prerequisites

- The root domain must already be configured (see Chapter 5).
- The server must have already been initialized as an NIS+ client (see Chapter 6).

- To configure a server you must be logged in as superuser on that machine.
- For the server to run in NIS-compatibility mode and support DNS forwarding, it must have a properly configured `/etc/resolv.conf` file (described in Chapter 1).

Information You Need

You need the superuser password of the client that you will convert into a server.

Setting Up an NIS+ Server — Task Map

TABLE 7–1 Setting Up an NIS+ Server

Task	Description	For Instructions, Go To
Setting Up an NIS+ Server	Set up an NIS+ server for NIS compatibility or NIS+ only.	"How to Configure an NIS+ Server" on page 127

▼ How to Configure an NIS+ Server

While it is possible to have a master or replica server serving more than one domain, doing so is not recommended.

1. **Log in as superuser to the new replica server.**

   The following steps assume that you rebooted the workstation after you set it up as a NIS+ client, as instructed in "Configuring the Client" on page 110. Rebooting starts the cache manager, which is a recommended prerequisite to the following step. If you did not reboot the workstation, restart the cache manager now, using `nis_cachemgr`.

2. **[NIS-Compatibility Only] Start the NIS+ daemon with** `–Y`**.**

   Perform this step only if you are setting up the server in NIS-compatibility mode; if setting up a standard NIS+ server, perform Step 3 on page 128 instead. This step also includes instructions for supporting the DNS forwarding capabilities of NIS clients.

   This step has two parts. The first part starts the NIS+ daemon in NIS-compatibility mode. The second part makes sure that when the server is rebooted, the NIS+ daemon restarts in NIS-compatibility mode.

a. Run `rpc.nisd` **with the** –Y **and** –B **flags.**

```
compatserver# rpc.nisd -Y -B
```

The –Y option invokes an interface that answers NIS requests in addition to NIS+ requests. The –B option supports DNS forwarding.

b. Edit the `/etc/init.d/rpc` **file.**

Search for the string `EMULYP=-Y` in the `/etc/init.d/rpc` file and uncomment that line.

To retain DNS forwarding capabilities, add a –B flag to the `EMULYP=-Y` line. (If you don't need to retain DNS forwarding capabilities, uncomment the line, but don't add the –B flag.)

This step creates a directory called `/var/nis/data` and a transaction log file called `trans.log`, which is placed in `/var/nis`.

```
compatserver# ls -F /var/nis
NIS_COLD_START data/ trans.log data.dict
```

The `trans.log` file is a transaction log. You can examine the contents of the transaction log by using the `nislog` command, described in the directories chapter of *Solaris Naming Administration Guide.*

Caution - Do not move or rename the `/var/nis` or `/var/nis/data` directories. Do not move or rename the `/var/nis/trans.log` or `/var/nis/data.dict` files. If you are upgrading from Solaris Release 2.4 or earlier, the older `/hostname` subdirectory is automatically converted to `/var/nis/data` and the relevant files are converted as necessary. Do *not* change these new names after the conversion has occurred.

Now this server is ready to be designated a master or replica of a domain, as described in Chapter 8. This step completes this task. A task summary is provided on "Server Configuration Summary" on page 136.

3. [Standard NIS+ Only] Start the NIS+ daemon.

Run the `rpc.nisd` command.

```
server# rpc.nisd
```

To verify that the NIS+ daemon is indeed running, use the `ps` command.

```
server# ps -ef | grep rpc.nisd
root 1081 1 16:43:33 ? 0:01 rpc.nisd
root 1087 1004 11 16:44:09 pts/1 0:00 grep rpc.nisd
```

This step creates a directory called `/var/nis/data` and a transaction log file called `trans.log` which is placed in `/var/nis`.

```
compatserver# ls -F /var/nis
NIS_COLD_START data/ trans.log data.dict
```

The `compatserver.log` file is a transaction log. You can examine the contents of the transaction log by using the `nislog` command, described in the directories chapter of *Solaris Naming Administration Guide.*

Caution - Do not move or rename the `/var/nis` or `/var/nis/data` directories. Do not move or rename the `/var/nis/trans.log` or `/var/nis/data.dict` files. If you are upgrading from Solaris Release 2.4 or earlier, the older `/hostname` subdirectory will be automatically converted to `/var/nis/data` and the relevant files will also be converted as necessary. Do *not* change these new names after the conversion has occurred.

Now this server is ready to be designated a master or replica of a domain, as described in Chapter 8. This step completes this task. A task summary is provided on "Server Configuration Summary" on page 136.

Adding a Replica to an Existing Domain

To have regularly available NIS+ service, you should always create one or more replica servers for each domain. Having replicas can also speed network-request resolution because multiple servers are available to handle requests.

For performance reasons, you should have no more than a few replicas per domain. If your network includes multiple subnets or different sites connected by a Wide Area Network (WAN), you might need additional replicas:

- *Subnets*. If you have a domain that spans multiple subnets, it is a good idea to have at least one replica server within each subnet so that, if the connection between nets is temporarily out of service, each subnet can continue to function until the connection is restored.
- *Remote sites*. If you have a domain spanning multiple sites linked over a WAN, it is a good idea to have at least one replica server on each side of the WAN link. For example, it may make sense from an organizational point of view to have two physically distant sites in the same NIS+ domain. If the domain's master server and all of its replicas are at the first site, there will be much NIS+ network traffic between the first and second sites. Creating an additional replica at the second site should reduce network traffic. See *NIS+ Transition Guide* for more information on replica distribution.

See *Solaris Naming Administration Guide* for additional information on how to determine the optimum number of replicas. To add a replica to an existing domain you must first configure the new replica, then load the NIS+ data set for your namespace.

The two ways to configure and load a new replica server are:

- *Scripts*. You can use the `nisserver` script, as described in "Creating a Root Replica Server" on page 70. This method automatically performs a full re-sync to load the NIS+ data set on to the new replica server. This is the preferred method because it is easiest, but it might be slower than using the NIS+ command set and backup / restore.
- *NIS+ command set*. You can use the NIS+ command set to configure a replica, as described in "Using NIS+ Commands to Configure a Replica Server" on page 131. This requires more knowledge of NIS+ than using `nisserver`. One advantage of this method is that it gives you the maximum amount of control and monitoring. Another advantage is that you can bring up a replica by manually creating the domain directories, then loading the NIS+ data set using `nisbackup` and `nisrestore`. Using the NIS+ backup and restore capability loads data faster than that used by `nisserver`.

  The two ways to load the NIS+ data set on to the newly configured replica server are:

  - `nisping`. When you configure a new replica server with either the `nisserver` script or the NIS+ command set, the master server automatically begins to load the namespace data set on to the new replica over the network using `nisping`. If your namespace is large, this could take a long time, during which requests for naming information could be delayed. See " Using `nisping` to Load Data on to a Replica Server" on page 135 for details.
  - *Backup and restore*. You can interrupt the transfer of data via `nisping`, and use the NIS+ backup and restore capabilities to load your namespace data on to a newly configured replica server, as described in "Using `nisrestore` to Load Data on to a Replica Server" on page 133. This is the preferred method because

the replica's data set is downloaded on to the replica, which is much faster than having the master transfer the data set to the replica over the network.

Using NIS+ Commands to Configure a Replica Server

This section describes how to add a replica server to an existing domain using the NIS+ command.

Security Considerations

The NIS+ principal performing this operation must have modify rights to the domain's directory object.

Prerequisites

- The domain must have already been configured and have a master server up and running.
- The new replica server must already be configured as an NIS+ server, as described in "Setting Up an NIS+ Server" on page 125.

Information You Need

- Name of the server
- Name of the domain

Using NIS+ Commands to Configure a Replica Server-Task Map

TABLE 7–2 Using NIS+ Commands to Configure a Replica Server

Task	**Description**	**For Instructions, Go To**
Setting Up an NIS+ Server	Use NIS+ commands to set up an NIS+ server	"How to Configure a Replica Server With NIS+ Commands" on page 132

TABLE 7–2 Using NIS+ Commands to Configure a Replica Server *(continued)*

▼ How to Configure a Replica Server With NIS+ Commands

In this example, the master server is named `master1`, and the new replica is named `replica2`.

1. **Log in to the domain's master server.**

2. **Make sure that `rpc.nisd` is running.**

3. **Add the replica to the domain.**

   Run the `nismkdir` command with the `–s` option. The example below adds the replica machine named `replica2` to the `doc.com.` domain.

```
master1# nismkdir -s replica2 doc.com.
master1# nismkdir -s replica2 org_dir.doc.com.
master1# nismkdir -s replica2 groups_dir.doc.com.
```

When you run the `nismkdir` command on a directory object that already exists, it does not recreate the directory but modifies it, according to the flags you provide. In this case, the `–s` flag assigns the domain an additional replica server. You can verify that the replica was added by examining the directory object's definition, using the `niscat -o` command.

Caution - Never run `nismkdir` on the replica machine. Running `nismkdir` on a replica creates communications problems between the master and the replicas.

Your new replica is now configured. You can now load your NIS+ data set on to the replica. You can do this in two ways:

- `nisping`. If you do nothing, your master server will use the `nisping` command to load your namespace data on to your newly configured replica server. If your namespace is large, this process can take hours. During this process, requests for naming information can be delayed. See " Using `nisping` to Load Data on to a Replica Server" on page 135 for details.
- *Backup and restore*. You can interrupt the transfer of data via `nisping` and use the NIS+ backup and restore capabilities to load your namespace data on to a

newly configured replica server, as described in “Using nisrestore to Load Data on to a Replica Server” on page 133. Because it is so much faster and more efficient, this is the preferred method.

Using nisrestore to Load Data on to a Replica Server

This section describes how to use the NIS+ backup and restore utilities to load namespace data on to a newly configured replica. This is the preferred method of loading data on to a replica.

Security Considerations

The NIS+ principal performing this operation must have modify rights to the domain’s directory object.

Prerequisites

- The domain must have already been configured and have a master server up and running.
- The new replica server must already be configured as an NIS+ server, as described in “Setting Up an NIS+ Server” on page 125.
- The new replica server must be configured as a replica, as described in “Using NIS+ Commands to Configure a Replica Server” on page 131.

Using nisrestore to Load Data on to a Replica Server — Task Map

TABLE 7–3 Using nisrestore to Load Data on to a Replica Server

Task	Description	For Instructions, Go To
Using nisrestore to Load Data on to a Replica Server	Use the nisrestore command to load data on to a replica server	“How to Load Namespace Data—nisrestore Method” on page 134

How to Load Namespace Data—`nisrestore` Method

In this example, the master server is named `master1`, and the new replica is named `replica2`.

1. **Kill `rpc.nisd` on the new replica server.**

   This interrupts the automatic transfer of namespace data from the master to the replica with the `nisping` command.

2. **Perform an NIS+ backup on the master server.**

   This step is described in more detail in *Solaris Naming Administration Guide*. The example below shows how to use the `nisbackup` command to backup up the `master1` server to the `/var/master1_bakup` directory.

   ```
   master1# nisbackup -a /var/master1_bakup
   ```

   The most convenient method of using `nisrestore` to configure a new replica is to back up the master's data to an NFS mounted directory that the new replica can access. This example assumes that both the master and the new replica server have access to the `/var/master1_bakup` directory.

   Another method is to use the `tar` command to copy the data from the `/var/master1_bakup` directory to some transferable storage media, such as a tape cartridge, then copy the data from storage media into a directory mounted on the new replica, then use that directory as the source for the `nisrestore` command, as described in Step 3 on page 134.

3. **Download the NIS+ data set onto the new replica using the `nisrestore` command.**

   This step is described in more detail in *Solaris Naming Administration Guide*. The example below shows how to use the `nisrestore` command to down load NIS+ data on to the `client2` replica from the `/var/master1_bakup` directory.

   ```
   replica2# nisrestore -a /var/master1_bakup
   ```

   If the replica you are creating is for the root domain, or if you get an error message that `nisrestore` cannot verify or look up needed data, then use the `nisrestore –f` option. For example:

   ```
   replica2# nisrestore -f -a /var/master1_bakup
   ```

4. **Restart `rpc.nisd` on the new replica**

   See "How to Configure an NIS+ Server" on page 127 for details.

Using nisping to Load Data on to a Replica Server

This section describes how to use the nisping command to load namespace data onto a newly configured replica. In most cases, it is not necessary to actually run the nisping command because the process should begin automatically.

The problem with the nisping method is that it requires a full resync of data from the master to the replica over the network using NIS+ protocols. If your namespace is large, this process can take hours, during which requests for naming information can be delayed.

Security Considerations

The NIS+ principal performing this operation must have modify rights to the domain's directory object.

Prerequisites

- The domain must have already been configured and have a master server up and running.
- The new replica server must already be configured as an NIS+ server, as described in "Setting Up an NIS+ Server" on page 125.
- The new replica server must by configured as a replica, as described in "Using NIS+ Commands to Configure a Replica Server" on page 131.

Using nisping to Load Data on to a Replica Server — Task Map

TABLE 7–4 Using nisping to Load Data on to a Replica Server

Task	Description	For Instructions, Go To
Using nisping to Load Data on to a Replica Server	Use nisping to load data on to a replica server	"How to Load Namespace Data—nisping Method" on page 136

▼ How to Load Namespace Data—`nisping` Method

Normally, the loading for namespace data is automatically initiated by the master server. If that does not occur, run the `nisping` command as described below.

♦ **Run** `nisping` **on the directories**

This step sends a message (a “ping”) to the new replica, telling it to ask the master server for an update. If the replica does not belong to the root domain, be sure to specify its domain name. (The example below includes the domain name only for completeness. Since the example used throughout this task adds a replica to the root domain, the `doc.com.` domain name in the example below is not necessary.)

```
master1# nisping doc.com.
master1# nisping org_dir.doc.com.
master1# nisping groups_dir.doc.com.
```

You should see results similar to these:

```
master1# nisping doc.com.
Pinging replicas serving directory doc.com. :
Master server is master1.doc.com.
 No last update time
Replica server is replica1.doc.com.
 Last update seen was Wed Nov 18 11:24:32 1992
 Pinging ... replica2.doc.com.
```

If your namespace is large, this process can take a significant amount of time. For more information about `nisping`, see the directories chapter of *Solaris Naming Administration Guide.*

Server Configuration Summary

Table 7–6 and Table 7–5 provide a summary of the tasks described in this chapter. They assume the simplest cases, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. This summary does not show the server’s responses to each command.

TABLE 7–5 Adding a Replica Named `replica2` to `doc.com.`: Command Summary

Tasks	Commands
Log in as superuser to domain master.	`master1% su`

Designate the new replica.	`# nismkdir -s replica2 doc.com.`
	`# nismkdir -s replica2 org_dir.doc.com.`
	`# nismkdir -s replica2 groups_dir.doc.com.`

Ping the replica.	`# /usr/lib/nis/nisping doc.com.`
	`# /usr/lib/nis/nisping org_dir.doc.com.`
	`# /usr/lib/nis/nisping groups_dir.doc.com.`

Note - Rather than using `nisping` to transfer data to the new replica, as shown in the example above, an easier method is to use the NIS+ backup and restore capability, as described in "Using `nisrestore` to Load Data on to a Replica Server" on page 133 and *Solaris Naming Administration Guide* .

TABLE 7–6 Starting Up a Non-root Master Server: Command Summary

Tasks	Commands
Log in to the server as root.	`server%su`
NIS-compat only: Start daemon with –`Y` –`B`	`server# rpc.nisd -Y - B`
Change to `EMULYP= -Y -B`.	`server# vi /etc/inet.d/rpc`
NIS+-Only: Start daemon.	`server# rpc.nisd`

CHAPTER **8**

Configuring a Non-Root Domain

This chapter provides step-by-step instructions for using the NIS+ command set to configure a subdomain domain (also known as a non-root domain) including designating its master and replica servers.

- "Setting Up a Non-Root Domain" on page 139
- "Subdomain Configuration Summary" on page 147

A summary of this task is provided by Table 8–2.

Setting Up a Non-Root Domain

Note - It is much easier to perform this task with the NIS+ installation scripts, as described in Part 1, than with the NIS+ command set as described here. The methods described in this chapter should be used only by those administrators who are very familiar with NIS+ and who require some nonstandard features or configurations not provided by the installation scripts.

You should not configure a non-root domain until *after* you have configured its servers.

Setting up a non-root domain involves the following tasks:

- Establishing security for the domain
- Creating the domain's directories
- Creating the domain's tables
- Designating the domain's servers

As with setting up the root domain, these tasks cannot be performed sequentially. To make the configuration process easier to execute, they have been broken down into individual steps and the steps have been arranged into the most efficient order.

Standard Versus NIS-Compatible Configuration Procedures

The differences between NIS-compatible and standard NIS+ servers in subdomains are the same as they are for servers in the root domain (see "Standard Versus NIS-Compatible Configuration Procedures" on page 88).

The NIS+ daemon for each server in an NIS-compatible domain should have been started with the –Y option, as instructed in Chapter 7. An NIS-compatible domain also requires its tables to provide read rights for the nobody class, which allows NIS clients to access the information stored in them. This is accomplished with the –Y option to the `nissetup` command, shown in Step 4 on page 143. (The standard NIS+ domain version uses the same command but without the –Y option, so it is described in the same step.)

Here is a summary of the entire configuration process:

1. Log in to the domain's master server.
2. Name the domain's administrative group.
3. Create the domain's directory and designate its servers.
4. Create the domain's subdirectories and tables.
5. Create the domain's admin group.
6. Assign full group access rights to the directory object.
7. Add the servers to the domain's admin group.
8. Add credentials for other administrators.
9. Add the administrators to the domain's admin group.

Security Considerations

Note - The NIS+ security system is complex. If you are not familiar with NIS+ security, you might want to review the security-related chapters of *Solaris Naming Administration Guide* before starting to configure your NIS+ environment.

At most sites, to preserve the security of the parent domain, only the parent domain's master server or an administrator who belongs to the parent domain's admin group is allowed to create a domain beneath it. Although this is a policy decision and not a requirement of NIS+, the instructions in this chapter assume that you are following that policy. Of course, the parent domain's admin group must have create rights to the parent directory object. To verify this, use the `niscat -o` command.

```
rootmaster# niscat -o doc.com.
Object Name : Doc
Owner : rootmaster
Group : admin.doc.com.
Domain : Com.
Access Rights : r---rmcdrmcdr---
:
```

If you are more concerned about convenience than security, you can make the new domain’s master server a member of its parent domain’s admin group, then perform the entire procedure from the server. Use the `nisgrpadm` command, described in the groups chapter of *Solaris Naming Administration Guide*.

Prerequisites

- The parent domain must be configured and running.
- The server that will be designated as this domain’s master must be initialized and running NIS+.
- If you will designate a replica server, the master server must be able to obtain the replica’s IP address through its `/etc/hosts` or `/etc/inet/ipnodes` file or from its NIS+ hosts table.

Information You Need

- The name of the new domain (for Step 3 on page 142)
- The name of the new domain’s master and replica servers
- The name of the new domain’s admin group (for Step 2 on page 142)
- User IDs (UID) of the administrators who will belong to the new domain’s admin group (for Step 8 on page 145)

Setting Up a Non-root Domain — Task Map

TABLE 8–1 Setting Up a Non-root Domain

Task	Description	For Instructions, Go To
Setting Up a Non-root Domain	Use NIS+ commands to set up a non-root domain	"How to Configure a Non-root Domain" on page 142

▼ How to Configure a Non-root Domain

1. **Log in to the domain's master server.**

   Log in to the server that you will designate as the new domain's master. The steps in this task use the server named `smaster`, which belongs to the `doc.com.` domain, and will become the master server of the `sales.doc.com.` subdomain. The administrator performing this task is `nisboss.doc.com.`, a member of the `admin.doc.com.` group. That group has full access rights to the `doc.com.` directory object.

2. **Name the domain's administrative group.**

   Although you won't actually create the admin group until Step 5 on page 144, you need to identify it now. This enables the `nismkdir` command used in the following step to create the directory object with the proper access rights for the group. It does the same for the `nissetup` utility used in Step 4 on page 143.

   Set the value of the environment variable `NIS_GROUP` to the name of the domain's admin group. Here are two examples, one for C shell users and one for Bourne or Korn shell users. They both set `NIS_GROUP` to `admin.sales.doc.com.`

   For C Shell

   ```
   smaster# setenv NIS_GROUP admin.sales.doc.com.
   ```

   For Bourne or Korn Shell

```
smaster# NIS_GROUP=admin.sales.doc.com.
smaster# export NIS_GROUP
```

3. **Create the domain's directory and designate its servers.**

   The `nismkdir` command, in one step, creates the new domain's directory and designates its supporting servers. It has the following syntax:

```
nismkdir -m master -s replica domain
```

The –m flag designates its master server, and the –s flag designates its replica.

```
smaster# nismkdir -m smaster -s salesreplica sales.doc.com.
```

Caution - Always run nismkdir on the master server. Never run nismkdir on the replica machine. Running nismkdir on a replica creates communications problems between the master and the replica.

The directory is loaded into /var/nis. Use the niscat –o command to view it (do not use cat or more).

```
smaster# niscat -o sales.doc.com.
Object Name : Sales
Owner : nisboss.doc.com.
Group : admin.sales.doc.com.
Domain : doc.com.
Access Rights : ----rmcdr---r---
.
```

Unlike the root directory, this directory object *does* have the proper group assignment. As a result, you won't have to use nischgrp.

4. **Create the domain's subdirectories and tables.**

   This step adds the org_dir and groups_dir directories and the NIS+ tables beneath the new directory object. Use the nissetup utility, but be sure to add the new domain name. And, for an NIS-compatible domain, include the –Y flag.

   NIS compatible

```
smaster# /usr/lib/nis/nissetup -Y sales.doc.com.
```

   NIS+

```
smaster# /usr/lib/nis/nissetup sales.doc.com.
```

   Each object added by the utility is listed in the output:

```
smaster# /usr/lib/nis/nissetup
org_dir.sales.doc.com. created
groups_dir.sales.doc.com. created
auto_master.org_dir.sales.doc.com. created
auto_home.org.dir.sales.doc.com. created
bootparams.org_dir.sales.doc.com. created
cred.org_dir.sales.doc.com. created
ethers.org_dir.sales.doc.com. created
group.org_dir.sales.doc.com. created
hosts.org_dir.sales.doc.com. created
mail_aliases.org_dir.sales.doc.com. created
sendmailvars.org_dir.sales.doc.com. created
netmasks.org_dir.sales.doc.com. created
netgroup.org_dir.sales.doc.com. created
networks.org_dir.sales.doc.com. created
passwd.org_dir.sales.doc.com. created
protocols.org_dir.sales.doc.com. created
rpc.org_dir.sales.doc.com. created
services.org_dir.sales.doc.com. created
timezone.org_dir.sales.doc.com. created
```

The –Y option creates the same tables and subdirectories as for a standard NIS+ domain, but assigns read rights to the nobody class so that requests from NIS clients, which are unauthenticated, can access information in the NIS+ tables.

You can verify the existence of the `org_dir` and `groups_dir` directories by looking in your master server's `/var/nis/data` directory. They are listed along with the root object and other NIS+ tables. The tables are listed under the `org_dir` directory. You can examine the contents of any table by using the `niscat` command, described in Chapter 9 (although at this point the tables are empty).

5. **Create the domain's admin group.**

   This step creates the admin group named in Step 2 on page 142. Use the `nisgrpadm` command with the –c option. This example creates the `admin.sales.doc.com.` group

```
smaster# nisgrpadm -c admin.sales.doc.com.
 Group admin.sales.doc.com. created.
```

   This step only creates the group—it does not identify its members. That is done in Step 9 on page 146.

6. **Assign full group access rights to the directory object.**

By default, the directory object grants only its group read access, which makes the group no more useful than the world class. To make the configuration of clients and subdomains easier, change the access rights that the directory object grants its group from read to read, modify, create, and destroy. Use the `nischmod` command.

```
smaster# nischmod g+rmcd sales.doc.com.
```

Complete instructions for using the `nischmod` command are provided in the rights chapter of *Solaris Naming Administration Guide.*

7. Add the servers to the domain's admin group.

At this point, the domain's group has no members. Add the master and replica servers, using the `nisgrpadm` command with the –`a` option. The first argument is the group name; the others are the names of the new members. This example adds `smaster.doc.com.` and `salesreplica.doc.com.` to the `admin.sales.doc.com.` group:

```
smaster# nisgrpadm -
a admin.sales.doc.com. smaster.doc.com. salesreplica.doc.com.
Added smaster.doc.com. to group admin.sales.doc.com.
Added salesreplica.doc.com. to group admin.sales.doc.com.
```

To verify that the servers are indeed members of the group, use the `nisgrpadm` command with the –`l` option (see the groups chapter of *Solaris Naming Administration Guide*).

```
smaster# nisgrpadm -l admin.sales.doc.com.
 Group entry for admin.sales.doc.com. group:
 Explicit members:
 smaster.doc.com.
 salesreplica.doc.com.
 No implicit members
 No recursive members
 No explicit nonmembers
 No implicit nonmembers
 No recursive nonmembers
```

8. Add credentials for other administrators.

Add the credentials of the other administrators who will work in the domain.

For administrators who already have DES credentials in another domain, add LOCAL credentials. Use the `nisaddcred` command with both the –`p` and the –`P` flags.

```
smaster# nisaddcred -p 33355 -P nisboss.doc.com. local
```

For administrators who do not yet have credentials, you can proceed in two different ways.

- One way is to ask them to add their own credentials. However, they will have to do this as superuser. Here is an example in which an administrator with a UID of 22244 and a principal name of `juan.sales.doc.com.` adds his own credentials to the `sales.doc.com.` domain.

```
smaster# nisaddcred -p 22244 -P juan.sales.doc.com. local
smaster# nisaddcred -p unix.22244@sales.doc.com -P juan.sales.doc.com. des
Adding key pair for unix.22244@sales.doc.com.
Enter login password:
```

- The other way is for you to create temporary credentials for the other administrators, using dummy passwords (each administrator must have an entry in the NIS+ passwd table).

```
smaster# nisaddcred -p 22244 -P juan.sales.doc.com. local
smaster# nisaddcred -p unix.22244@sales.doc.com -P juan.sales.doc.com. des
Adding key pair for unix.22244@sales.doc.com.
Enter juan’s login password:
nisaddcred: WARNING: password differs from login passwd.
Retype password:
```

Each administrator can later change his or her network password by using the `chkey` command. The credentials and keys chapters of *Solaris Naming Administration Guide* describe how to do this.

Note - In the two examples shown in Step 8 on page 145, the domain name following the lower case `–p` flag must *never* end in a trailing dot, while the domain name following the upper case `–P` flag must *always* end in a trailing dot.

---

9. Add the administrators to the domain’s admin group.

You don’t have to wait for the other administrators to change their dummy passwords to perform this step. Use the `nisgrpadm` command with the `–a` option. The first argument is the group name, and the remaining arguments are the names of the administrators. This example adds the administrator `juan` to the `admin.sales.doc.com.` group:

```
smaster# nisgrpadm -a admin.sales.doc.com. juan.sales.doc.com.
Added juan.sales.doc.com. to group admin.sales.doc.com.
```

10. **Allocate sufficient swap space to accommodate NIS+ tables.**

    Swap space should be double the size of the maximum size of rpc.nisd. To determine how much memory rpc.nisd is using, issue the following command:

    ```
    rootmaster# /usr/lib/nis/nisstat
    ```

    rpc.nisd will under certain circumstances fork a copy of itself. If there is not enough memory, rpc.nisd fails.

    You can also calculate the memory and swap space requirements for NIS+ tables. For example, if you have 180,000 users and 180,000 hosts in your NIS+ tables, those two tables occupy approximately 190 Mbytes of memory. When you add credentials for 180,000 users and 180,000 hosts, the cred table has 540,000 entries (one entry for each local user credential, one entry for each DES user credential, and one entry for each host). The cred table occupies approximately 285 Mbytes of memory. In this example, rpc.nisd occupies at least 190 Mbytes + 285 Mbytes = 475 Mbytes of memory. So, you will require at least 1 Gbyte swap space. You will also want at least 500 Mbytes of memory to hold rpc.nisd entirely in memory.

Subdomain Configuration Summary

Table 8–2 is a summary of the steps required to configure a non-root domain. It assumes the simplest case, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. This summary does not show the server’s responses to each command.

TABLE 8–2 Setting Up a Subdomain Command Summary

Tasks	Commands
Log in as superuser to domain master.	`smaster% su`

Name the domain's admin group.	`# NIS_GROUP=admin.sales.doc.com.`
	`# export NIS_GROUP`

Create the domain's directory and designate its servers.	`# nismkdir -m smaster -s salesreplica sales.doc.com.`
Create `org_dir`, `groups_dir`, and tables. (For NIS-compatibility, use `-Y`.)	`# /usr/lib/nis/nissetup sales.doc.com.`
Create the admin group.	`# nisgrpadm -c admin.sales.doc.com.`
Assign full group rights to the domain's directory.	`# nischmod g+rmcd sales.doc.com.`
Add servers to admin group.	`# nisgrpadm -a admin.sales.doc.com. smaster.doc.com. sreplica.doc.com.`

Add credentials for other admins.	`# nisaddcred -p 22244 -P juan.sales.doc.com. local`
	`# nisaddcred -p unix.22244@sales.doc.com. juan.sales.doc.com. DES`

Add admins to domain's admin group.	`# nisgrpadm -a admin.sales.doc.com. juan.sales.doc.com.`

CHAPTER **9**

Setting Up NIS+ Tables

This chapter provides step-by-step instructions for using the NIS+ command set to populate NIS+ tables on a master server from `/etc` files or NIS maps, how to transfer information back from NIS+ tables to NIS maps, how to limit access to the passwd column of the `passwd` table.

- "Populating Tables—Options" on page 150
- "Populating NIS+ Tables From Files" on page 151
- "Populating NIS+ Tables From NIS Maps" on page 157
- "Transferring Information From NIS+ to NIS" on page 163
- "Limiting Access to the `Passwd` Column to Owners and Administrators" on page 164

Setting Up Tables

Note - It is much easier to perform this task with the NIS+ installation scripts, as described in Part 1, than with the NIS+ command set, as described here. The methods described in this chapter should be used only by those administrators who are very familiar with NIS+ and who require some nonstandard features or configurations not provided by the installation scripts. Also, if you have them available, the Solstice AdminSuite tools provide easier methods of working with NIS+ tables.

You can populate NIS+ tables in four ways:

- From files, as described in "Populating NIS+ Tables From Files" on page 151
- From NIS maps, as described in "Populating NIS+ Tables From NIS Maps" on page 157

- With the `nispopulate` script, as described in “Populating NIS+ Tables” on page 55 and “Populating the New Subdomain’s Tables” on page 78
- With Solstice AdminSuite tools, if you have them available

When populating tables from maps or files, the tables should have already been created in the process of setting up a root or subdomain, as explained in Chapter 5 and Chapter 8. Although you can populate a domain’s tables at any time after they are created, it is recommended that you do so immediately after setting up the domain. This enables you to add clients more easily, since the required information about the clients should already be available in the domain’s tables.

Populating Tables—Options

When you populate a table—whether from a file or an NIS map—you can use any of these options:

- *Replace* - With the replace option, NIS+ first deletes all existing entries in the table and then adds the entries from the source. In a large table, this adds a large set of entries into the master server’s `/var/nis/trans.log` file (one set for removing the existing entries, another for adding the new ones), taking up space in `/var/nis`. Thus, propagation to replicas will take longer.
- *Append* - The append option adds the source entries to the NIS+ table. Existing entries are not touched.
- *Merge* — The merge option produces the same results as the replace option but uses a different process. The Merge option updates existing entries rather than deleting and then replacing them. With the merge option, NIS+ handles three types of entries differently:

  - Entries that exist only in the source are added to the table.
  - Entries that exist in both the source and the table are updated in the table.
  - Entries that exist only in the NIS+ table are deleted from the table.

  When updating a large table with a file or map whose contents are not vastly different from those of the table, the merge option can spare the server a great many operations. Because it deletes only the entries that are not duplicated in the source (the replace option deletes *all* entries, indiscriminately), it saves one delete and one add operation for every duplicate entry. Therefore, this is the preferred option.

Populating NIS+ Tables From Files

This task transfers the contents of an ASCII file, such as `/etc/hosts`, into an NIS+ table.

Here is an outline of the procedure:

1. Check the content of each file that you will be transferring data from.
2. Make a copy of each file. Use this copy for the actual transfer. In this guide, copies of files to be transferred are given names ending in `xfr` (for example, `hosts.xfr`).
3. Log in to an NIS+ client. (You must have credentials and permissions allowing you to update the tables. See "Files Security Considerations" on page 151, below.)
4. Add `/usr/lib/nis` to the search path for this shell (if not already done).
5. Use `nisaddent` to transfer any of these files one at a time: `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`, `shadow`, and `ipnodes`.

Note - The new `/etc/inet/ipnodes` file contains IPv4 and IPv6 addresses. Use `nisaddent` to transfer the `/etc/inet/ipnodes` file into the `ipnodes.org_dir` table.

6. Transfer the `publickey` file.
7. Transfer the automounter information.
8. Ping any replicas.
9. Checkpoint the tables.

Files Security Considerations

You can populate NIS+ tables from any NIS+ client or from the root master server. You do not have to be logged in as superuser (root) to populate NIS+ tables, but you do have to have the proper credentials and access rights. If you are going to replace or merge the entries in the table with the entries from the text file, you must have create and destroy rights to the table. If you are going to append the new entries, you only need create rights.

Note - The NIS+ security system is complex. If you are not familiar with NIS+ security, you may want to review the security-related chapters of *Solaris Naming Administration Guide* before starting to set up your NIS+ environment.

After you complete this operation, the table entries will be owned by the NIS+ principal that performed the operation and the group specified by the `NIS_GROUP` environment variable.

Prerequisites

- The domain must have already been set up and its master server must be running.
- The domain's servers must have enough swap space to accommodate the new table information. See *NIS+ Transition Guide* for information on NIS+ requirements.
- The information in the file must be formatted appropriately for the table into which it will be loaded. See "Prerequisites to Running nispopulate" on page 56 for information concering what format a text file must have to be transferred into its corresponding NIS+ table. Local /etc files are usually formatted properly, but may have several comments that you need to remove.
- Machine and user names cannot be duplicated. All users and all machines must have unique names. You cannot have a machine with the same name as a user.
- Machine names cannot contain dots (periods) or underscores. For example, a machine named sales.alpha is not allowed. Hyphens, however, are allowed. For example, a machine name such as sales-alpha is allowed.

Information You Need

You need the name and location of the text files that will be transferred.

Populating NIS+ Tables From Files — Task Map

TABLE 9–1 Populating NIS+ Tables From Files

Task	Description	For Instructions, Go To
Populating NIS+ Tables From Files	Populate NIS+ tables from files	"How to Populate NIS+ Tables From Files" on page 152

▼ How to Populate NIS+ Tables From Files

1. **Check each file that you will be transferring data from.**

   Make sure that there are no spurious or incorrect entries. Make sure that the right data is in the correct place and formatted properly. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. (It is easier to add incomplete entries after setup than to try transferring incomplete or damaged entries from the file.)

2. **Make a working copy of each file you will be transferring.**

   Use this working copy for the actual file transfer steps described in this section. Give each working copy the same filename extension (for example, .xfr).

   ```
   rootmaster% cp /etc/hosts /etc/hosts.xfr
   ```

   For safety reasons, you might also copy all of the files you plan to use to some directory other than /etc. The nisaddent and nispopulate commands allow you to specify the file source directory.

3. **Log in to an NIS+ client.**

   You can perform this task from any NIS+ client—just be sure that the client belongs to the same domain as the tables into which you want to transfer the information. The examples in this task use the root master server. Because the administrator in these examples is logged on as superuser, the NIS+ principal actually performing this operation (and therefore needing the proper credentials and access rights) is the root master server.

   However, it is *not necessary* to be a superuser (root) or to be logged in on the root master server to update NIS+ tables. Regular users or superusers on any machine can update NIS+ tables, so long as they have the proper credentials, authorization, file permissions.

4. **Add /usr/lib/nis to the search path for this shell.**

   Since you will be using the /usr/lib/nis/nisaddent command once per table, adding its prefix to the search path will save you the trouble of typing it each time. Here are two examples, one for C shell users and one for Bourne or Korn shell users:

   For C Shell

   ```
   rootmaster# setenv PATH $PATH:/usr/lib/nis
   ```

   For Bourne or Korn Shell

```
rootmaster# PATH=$PATH:/usr/lib/nis
rootmaster# export PATH
```

5. **Use nisaddent to transfer any of these files, one at a time:**

```
aliases
bootparams
ethers
group
hosts
ipnodes
netgroup
netmasks
networks
protocols
rpc, services
```

The `publickey`, automounter, `passwd`, and `shadow` files require slightly different procedures; for the `publickey` file, go to Step 6 on page 155; for the automounter files, go to Step 7 on page 156; for the `passwd` and `shadow` files, go to Step 8 on page 156.

By default, `nisaddent` *appends* the information. To replace or merge instead, use the `–r` or `–m` options.

To replace:

```
rootmaster# nisaddent -r -f filename table[domain]
```

To append:

```
rootmaster# nisaddent -a -f filename table [domain]
```

To merge:

```
rootmaster# nisaddent -m -f filename table [domain]
```

The best option for populating the tables for the first time is the `–a` option, the default. The best option to synchronize the NIS+ tables with NIS maps or `/etc` files is the `–m` (merge) option.

- *filename* is the name of the file. The common convention is to append `.xfr` to the end of these file names to identify them as transfer files created with `nisaddent`.
- *table* is the name of the NIS+ table. The *domain* argument is optional; use it only to populate tables in a different domain. Here are some examples, entered from

the root domain's master server. The source files are edited versions of the `/etc` files:

```
rootmaster# nisaddent -m -f /etc/hosts.xfr hosts
rootmaster# nisaddent -m -f /etc/groups.xfr groups
```

If you perform this operation from a non-root server, keep in mind that a non-root server belongs to the domain above the one it supports; therefore, it is a client of another domain. For example, the `sales.doc.com.` master server belongs to the `doc.com.` domain. To populate tables in the `sales.doc.com.` domain from that master server, you must append the `sales.doc.com.` domain name to the `nisaddent` statement.

```
salesmaster# nisaddent -f /etc/hosts.xfr hosts Sales.doc.com.
```

If you perform this operation as a client of the `sales.doc.com.` domain, you do not need to append the domain name to the syntax. For more information about `nisaddent`, see the tables chapter of *Solaris Naming Administration Guide*.

To verify that the entries were transferred into the NIS+ table, use the `niscat` command, as described more fully in the tables chapter of *Solaris Naming Administration Guide*.

```
rootmaster# niscat groups.org_dir
root::0:root
other::1::
bin::2:root,bin,daemon
.
```

Troubleshooting tip: If `niscat` does not now show the updates immediately, it could be because the changes have not been sent by the master server to one or more of the replica servers. In this case, you can either wait and try again in five or ten minutes or use `niscat`'s `–M` option, which specifies that `niscat` obtain its data from the master server.

6. **Transfer the `publickey` file.**

   Because the domain's `cred` table already stores some credentials, you need to make sure they are not overwritten by the contents of the `publickey` text file that you transfer into the cred table. You can avoid this by removing those credentials from the `publickey` text file. For `rootmaster`, there might be one or more lines like the following, all of which should be removed:

   ```
   unix.rootmaster@doc.com public-key:private-key [alg-type]
   ```

Then you can transfer the contents of the publickey file to the cred table. Use nisaddent, with the –a (add) option.

```
rootmaster# nisaddent -a -f /etc/publickey.xfr -t cred.org_dir publickey [domain]
```

Note, however, that this operation transfers only DES credentials into the cred table. You still need to create their LOCAL credentials to the cred table.

7. **Transfer the automounter information.**

   Although the nissetup utility creates auto_master and auto_home tables, they are not considered standard NIS+ tables. Therefore, transferring information into them requires a slightly different syntax; in particular, you must use the –t flag and specify that the table is of type key-value.

```
rootmaster# nisaddent -f auto.master.xfr -t auto_master.org_dir key-value
rootmaster# nisaddent -f auto.home.xfr -t auto_home.org_dir key-value
```

8. **Build the NIS+ passwd table.**

   The NIS+ passwd table is composed of data drawn from both the /etc/passwd and /etc/shadow files. Thus, you must run nisaddent twice to build the passwd table: once for the /etc/passwd file, using passwd as the target table, and once for the /etc/shadow file, using shadow as the target table. (Note that when running nisaddent on the shadow file, you specify shadow as the target table, even though there is no shadow table and the data is actually being placed in the shadow column of the passwd table.)

```
rootmaster# nisaddent -m -f /etc/passwd.xfr passwd
rootmaster# nisaddent -m -f /etc/shadow.xfr shadow
```

9. **Transfer your updates to your replica servers by running nisping.**

   Running nisping updates any replica servers with your changes.

```
master1# nisping domain
master1# nisping org_dir.domaincom.
master1# nisping groups_dir.domain
```

10. **Checkpoint the tables.**

Now that you have updated the in-memory copies of the NIS+ data set on your master and replica servers, you should write those changes into the table files on disk. This is called *checkpointing*. (Checkpoint is not *mandatory* at this time, so long as you have regularly scheduled checkpoints that will take care of it later. But if your changes have been significant, it is a good idea to get them written to disk as soon as convenient.)

This step ensures that all the servers supporting the domain transfer the new information from their `.log` files to the disk-based copies of the tables. If you have just set up the root domain, this step affects only the root master server, since the root domain does not yet have replicas. To checkpoint, use the `nisping` command with the `–C` (uppercase) option.

```
rootmaster# nisping -C org_dir
Checkpointing replicas serving directory org_dir.doc.com. :
Master server is rootmaster.doc.com.
 Last update occurred at July 14, 1997
Master server is rootmaster.doc.com.
checkpoint succeeded.
```

If you do not have enough swap space, the server is unable to checkpoint properly, but it will not notify you. One way to make sure that you have sufficient swap space is to list the contents of a table with the `niscat` command. If you do not have enough swap space, you will see this error message:

```
can’t list table: Server busy, Try Again.
```

Even though it doesn’t *seem* to, this message indicates that you don’t have enough swap space. Increase the swap space and checkpoint the domain again.

Populating NIS+ Tables From NIS Maps

This task transfers the contents of an NIS map into an NIS+ table. Here is a list of the steps:

1. Check the content of each NIS map that you will be transferring data from.
2. Log in to an NIS+ client.
3. Add `/usr/lib/nis` to the search path for this shell.

4. Use `nisaddent` to transfer any of these maps, one at a time: `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`.
5. Transfer the `publickey` map.
6. Transfer the automounter information.
7. Update your replicas with your changes by running `nisping`.
8. Checkpoint the tables.

Maps Security Considerations

You can perform this task from any NIS+ client as long as you (or superuser on the client) have the proper credentials and access rights. If you are going to replace or merge the entries in the table with the entries from the NIS map, you must have create and destroy rights to the table. If you are going to append the new entries, you need only create rights.

After you complete this operation, the table entries are owned by the NIS+ principal that performed the operation (either you or, if logged on as superuser, the client) and the group specified by the `NIS_GROUP` environment variable.

Prerequisites

- The domain must have already been set up and its master server must be running.
- The `dbm` files (`.pag` and `.dir` files) for the NIS maps you are going to load into the NIS+ tables must already be in a subdirectory of `/var/yp`.
- Machine and user names cannot be duplicated. All users and all machines must have unique names. You cannot have a machine with the same name as a user.
- Machine names cannot contain dots (periods). For example, a machine named `sales.alpha` is not allowed. A machine named `sales-alpha` is allowed.

Information You Need

You need the name and location of the NIS maps.

Populating NIS+ Tables From NIS Maps — Task Map

TABLE 9–2 Populating NIS+ Tables From NIS Maps

Task	Description	For Instructions, Go To
Populating NIS+ Tables From NIS Maps	Populate NIS+ tables from NIS maps	"How to Populate Tables From Maps" on page 159

How to Populate Tables From Maps

1. **Check each NIS map that you will be transferring data from.**

   Make sure that there are no spurious or incorrect entries. Make sure that the right data is in the correct place and format properly. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. (It is easier to add incomplete entries after setup than to try transferring incomplete or damages entries from the map.)

2. **Log in to an NIS+ client.**

   You can perform this task from any NIS+ client—so long as that client belongs to the same domain as the tables into which you want to transfer the information. The examples in this task use the root master server. Since the administrator in these examples is logged in as superuser, the NIS+ principal actually performing this operation (and therefore needing the proper credentials and access rights) is the root master server.

3. **Add `/usr/lib/nis` to the search path for this shell.**

   Because you will be using the `/usr/lib/nis/nisaddent` command once for each table, adding its prefix to the search path will save you the trouble of typing it each time. Here are two examples, one for C shell users and one for Bourne or Korn shell users:

   For C Shell

   ```
   rootmaster# setenv PATH $PATH:/usr/lib/nis
   ```

   For Bourne or Korn Shell

```
rootmaster# PATH=$PATH:/usr/lib/nis
rootmaster# export PATH
```

4. **Use `nisaddent` to transfer any of these maps, one at a time:**

aliases, bootparams, ethers, group, hosts, netgroup, netmasks, networks, passwd, protocols, rpc, services.

The –publickey and automounter maps require slightly different procedures; for the publickey file, go to Step 6 on page 155, and for the automounter files, go to Step 7 on page 156.

By default, nisaddent *appends* the file information to the table information. To replace or merge instead, use the –r or –m options:

To replace:

```
rootmaster# nisaddent -r -y nisdomain table
```

To append:

```
rootmaster# nisaddent -a -y nisdomain table
```

To merge:

```
rootmaster# nisaddent -m -y nisdomain table
```

The best option for populating the tables for the first time is the –a option, which is the default. The best option to synchronize the NIS+ tables with NIS maps or /etc files is the –m (merge) option.

The –y (lowercase) option indicates an NIS domain instead of a text file. The *nisdomain* argument is the name of the NIS domain whose map you are going transfer into the NIS+ table. You don't have to name the actual map; the nisaddent utility automatically selects the NIS map that corresponds to the *table* argument. Here are some examples:

```
rootmaster# nisaddent -m -y olddoc hosts
rootmaster# nisaddent -m -y olddoc passwd
rootmaster# nisaddent -m -y olddoc groups
```

The first example transfers the contents of the hosts.byname and hosts.byaddr maps in the olddoc (NIS) domain to the NIS+ hosts table in the root domain (NIS+). The second transfers the NIS maps that store password-related information into the NIS+ passwd table. The third does the same with group-related information. For more information about the nisaddent command, see the tables chapter of *Solaris Naming Administration Guide.*

5. **Transfer the** publickey **map.**

Since the domain's cred table already stores some credentials, you need to make sure they are not overwritten by the contents of the publickey map that you transfer into the cred table.

a. **First, dump the publickey map to a file, then open that file with your text editor.**

```
rootmaster# makedbm -u /var/yp/olddoc/publickey.byname /etc/publickey.xfr
rootmaster# vi /tmp/publickey.tmp
```

b. **Now remove the credentials of the workstation you are logged in to from the publickey map.**

For rootmaster, there might be one or lines like the following, all of which should be removed:

```
unix.rootmaster@doc.com public-key:private-key [alg-type]
```

c. **Now you can transfer the contents of the *file*—not the map—into the cred table. Use nisaddent, with the –a (add) option.**

```
rootmaster# nisaddent -a -f /etc/publickey.xfr -t cred.org_dir Publickey
```

Notice that this operation transfers only DES credentials into the cred table. You still need to create their LOCAL credentials to the cred table.

6. **Transfer the automounter information.**

Although the nissetup utility creates auto_master and auto_home tables, they are not considered standard NIS+ tables. Therefore, transferring information into them requires a slightly different syntax:

```
rootmaster# nisaddent -y olddoc -Y auto.master -t auto_master.org_dir key-value
rootmaster# nisaddent -y olddoc -Y auto.home -t auto_home.org_dir key-value
```

The –m and –y options are still required, as is the NIS domain name (in this instance, olddoc). However, you must precede the name of the NIS map (for example, auto.master) with a –Y (uppercase). Then, as is required when transferring automounter *text files*, you must use the –t option, which indicates that this is a nonstandard NIS+ table. Its arguments are the name of the NIS+ directory object (auto_master.org_dir) and the type of table (key-value). Be sure to append the org_dir suffixes to the NIS+ table names.

7. **Transfer your updates to your replica servers by running `nisping`.**

   Running `nisping` updates any replica servers with your changes.

```
master1# nisping domain
master1# nisping org_dir.domaincom.
master1# nisping groups_dir.domain
```

8. **Checkpoint the tables.**

   This step ensures that all the servers supporting the domain transfer the new information from their `.log` files to the disk-based copies of the tables. If you just finished setting up the root domain, this step affects only the root master server, since the root domain does not yet have replicas. Use the `nisping` command with the `–C` (uppercase) option.

```
rootmaster# nisping -C org_dir
Checkpointing replicas serving directory org_dir.doc.com. :
Master server is rootmaster.doc.com.
 Last update occurred at July 14, 1994
Master server is rootmaster.doc.com.
checkpoint succeeded.
```

If you do not have enough swap space, the server is unable to checkpoint properly, but it does not notify you. One way to make sure you have sufficient swap space is to use list the contents of a table with the `niscat` command. If you do not have enough swap space, you will see this error message:

```
can’t list table: Server busy, Try Again.
```

Even though it does not *seem* to, this message indicates that you do not have enough swap space. Increase the swap space and checkpoint the domain again.

Transferring Information From NIS+ to NIS

This task transfers the contents of NIS+ tables into NIS maps on a Solaris 1.x NIS master server. Here is an outline of the procedure:

1. Log in to the NIS+ server.
2. Transfer the NIS+ tables in to output files.
3. Transfer the contents of the output files to the NIS maps.

NIS to NIS+ Security Considerations

To perform this task, you must have read access to each table whose contents you transfer.

Prerequisites

The maps must already have been built on the NIS server.

Transferring Information From NIS+ to NIS — Task Map

TABLE 9–3 Transferring Information From NIS+ to NIS

Task	Description	For Instructions, Go To
Transferring Information From NIS+ to NIS	Transfer information from NIS+ tables to NIS maps on a Solaris 1.x NIS master server	"How to Transfer Information From NIS+ to NIS" on page 163

▼ How to Transfer Information From NIS+ to NIS

1. **Log in to the NIS+ server.**

   This example uses the server named `dualserver`.

2. **Transfer the NIS+ tables to output files.**

Use the `nisaddent` command with the `–d` option, once for each table.

```
dualserver% /usr/lib/nis/nisaddent -d -t table tabletype > filename
```

The `–d` option transfers the contents of *table* to *filename*, converting the contents back to standard `/etc` file format.

3. **Transfer the contents of the output files in to the NIS maps.**

   The NIS+ output files are ASCII files that you can use as input files for the NIS maps. Copy them into the NIS master's `/etc` directory, then use `make` as usual.

```
dualserver# cd /var/yp
dualserver# make
```

Limiting Access to the `Passwd` Column to Owners and Administrators

This task describes how to limit read access to the password-related columns of the `passwd` table to the entry owner and the table administrators, without affecting the read access of other authenticated principals (including applications) to the remaining columns of the passwd table.

This task establishes the following rights:

```
                          Nobody  Owner   Group  World
Table Level Rights:       ----    rmcd    rmcd   ----
Passwd Column Rights:     ----    rm--    rmcd   ----
Shadow Column Rights:     ----    rm--    rmcd   ----
```

`Passwd` Column Security Considerations

- The domain must *not* be running in NIS-compatibility mode.
- All clients of the domain must have DES credentials.
- All clients of the domain must be running Solaris Release 2.3 or a later release.
- Users' network passwords (used to encrypt their DES credentials) must be the same directory as their login passwords.

Prerequisites

- The passwd table must have already been set up. It need not have any information in it, however.
- The NIS+ principal performing this task must have modify rights to the passwd table.

Information You Need

All you need is the name of the passwd table.

Limiting Access to the Passwd Column to Owners and Administrators — Task Map

TABLE 9–4 Limiting Access to the Passwd Column to Owners and Administrators

Task	Description	For Instructions, Go To
Limiting Access to the Passwd Column to Owners and Administrators	Modify passwd.org_dir, via NIS+ commands, to restrict access to the passwd column for owners and administrators.	"How to Limit Read Access to the Passwd Column" on page 165

▼ How to Limit Read Access to the Passwd Column

1. **Log in to the domain's master server.**

   The examples in this task use the root master server, rootmaster.

2. **Check the current table and column permissions.**

   Use the niscat –o command.

```
rootmaster# niscat -o passwd.org_dir
```

   This task assumes the existing permissions are:

```
Access Rights    : ----rmcdrmcdr---
Columns          :
                     [0]  Name              : name
```

(continued)

(Continuation)

```
      Access Rights : r-----------r---
[1]  Name              : passwd
      Access Rights : -----m----------
[2]  Name              : uid
      Access Rights : r-----------r---
[3]  Name              : gid
      Access Rights : r-----------r---
[4]  Name              : gcos
      Access Rights : r----m------r---
[5]  Name              : home
      Access Rights : r-----------r---
[6]  Name              : shell
      Access Rights : r-----------r---
[7]  Name              : shadow
      Access Rights : r-----------r---
```

If your permissions are different, you may need to use a different syntax. For instructions, see the rights chapter of *Solaris Naming Administration Guide*.

3. **Change the table permissions.**

   Use the `nischmod` command to change the table's object-level permissions to `---- rmcdrmcd ----`

```
rootmaster# nischmod og=rmcd,nw= passwd.org_dir
```

4. **Change the column permissions.**

   Use the `nistbladm` command with the `–u` option to change the permissions of the passwd and shadow columns to:

```
passwd ---- rm-- ---- ----
shadow ---- r--- ---- ----
rootmaster# nistbladm -u passwd=o+r, shadow=o+r passwd.org_dir
```

5. **Verify the new permissions.**

   Use the `niscat –o` command, as you did in Step 2 on page 165. The permissions should look the same as they do in that step's output.

Table Population Summaries

Following are summaries of the steps required to populate NIS+ tables. They assume the simplest case, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. For brevity, these summaries do not show the server's responses to each command.

TABLE 9–5 Transferring Files Into NIS+ Tables: Command Summary

Tasks	Commands
Log in to an NIS+ client.	`rootmaster%`
Create working copies of the files to be transferred.	`% cp /etc/hosts /etc/hosts.xfr`
Add `/usr/lib/nis` to search path.	`% PATH=$PATH:/usr/lib/nis; export PATH`
Transfer each file, one at a time.	`% nisaddent -m -f /etc/hosts.xfr hosts`
Remove old server credentials from `publickey` file.	`% vi /etc/publickey.xfer`
Transfer it to the cred table.	`% nisaddent -a -f /etc/publickey.xfr cred`

Transfer the automounter files.	`% nisaddent -f auto.master.xfr -t auto_master.org_dir key-value`
	`% nisaddent -f auto.home.xfr -t auto_home.org_dir key-value`

Checkpoint the table directory.	`% nisping -C org_dir`

TABLE 9–6 Transferring Maps Into NIS+ Tables: Command Summary

Tasks	Commands
Log in to an NIS+ client.	`rootmaster%`
Add `/usr/lib/nis` to search path.	`% PATH=$PATH:/usr/lib/nis; export PATH`
Transfer each map, one at a time.	`% nisaddent -m -y olddoc hosts`
Dump `publickey` map to a file.	`% makedbm -u /var/yp/olddoc/publickey.byname > /etc/ publickey.xfr`
Remove new credentials.	`% vi /etc/publickey.xfr`
Transfer the `publickey` file.	`% nisaddent -a -f /etc/publickey.xfr -t cred.ortg_dir publickey`

Transfer the automounter maps.	`% nisaddent -y olddoc -Y auto.master -t auto_master.org_dir key-value`
	`% nisaddent -y olddoc -Y auto.home -t auto_home.org_dir key-value`

Checkpoint the table directory.	`% nisping -C org_dir`

TABLE 9–7 Transferring NIS+ Tables to NIS Maps: Command Summary

Tasks	Commands
Log in to NIS+ server.	`dualserver%`
Transfer NIS+ tables to files.	`% /usr/lib/nis/nisaddent -d [-t` *table*`]` *tabletype* `>` *filename*
Transfer files to NIS maps.	`% makedbm` *flags output-file NIS-dbm-file*

TABLE 9–8 Limiting Access to Passwd Column: Command Summary

Tasks	Commands
Log into the domain's master server.	`rootmaster#`
Check the table's existing rights.	`# niscat -o passwd.org_dir`
Assign the table new rights.	`# nischmod og=rmcd,nw= passwd.org_dir`
Assign the columns new rights	`# nistbladm -u passwd=o+r, shadow=n+r passwd.org_dir`
Verify the new rights.	`# niscat -o passwd.org_dir`

PART III NIS Setup and Configuration

This part describes how to set up and configure the Network Information Service (NIS) service. It has one chapter.

CHAPTER **10**

Configuring NIS Service

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

See *Solaris Naming Administration Guide* for a general description and overview of NIS.

NIS in the Solaris 8 Operating Environment

The following NIS features are new or different in Solaris 8 Operating Envirnment.

NIS and IPv6

New ipnodes maps (ipnodes.byaddr and ipnodes.byname)are added to NIS. They store both IPv4 and IPv6 addresses (see the ipnodes(4) man page. NIS clients and servers can communicate using either IPv4 or IPv6 RPC transports.

Before You Begin Configuring NIS

Before configuring your NIS namespace, you must:

- Install properly configured nsswitch.conf files on all the machines that will be using NIS. See Chapter 1 for details.
- Plan your NIS domain "Planning Your NIS Domain" on page 174.

Planning Your NIS Domain

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

Planning the Domain

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

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

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

Identify Your NIS Servers

Decide which machines will be NIS servers. Select one machine to be the master server (you can always change this at a later date). Decide which machines, if any, will be slave servers. (See *Solaris Naming Administration Guide* for a general overview of NIS and NIS requirements.)

Identify Your NIS Client Machines

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

NIS Configuration Steps

After your Solaris 7 release software and `nsswitch.conf` files are installed, and your domain planned, you must perform the following steps to configure NIS:

1. Prepare the master server (see "Preparing the Master Server" on page 175).
2. Configure the NIS master server (see "How to Set Up the Master Server With `ypinit`" on page 188).
3. Start the NIS daemons on the master server (see "Starting NIS Service on the Master Server" on page 190).
4. Configure your slave servers (see "Setting Up NIS Slave Servers" on page 191).
5. Configure NIS client machines (see "Setting Up NIS Clients" on page 193).

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

The following sections explain these steps in detail.

Preparing the Master Server

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

Source Files Directory

The source files may be located either in the /etc directory on the master server or in some other directory. Having them in /etc might be undesirable because the contents of the maps are then the same as the contents of the local files on the master server. This is a special problem for passwd and shadow files, because all users would have access to the master server maps and the root password would be passed to all YP clients through the passwd map. (See "Passwd Files and Namespace Security" on page 176 for additional information on this subject).

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

Passwd Files and Namespace Security

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

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

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

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

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

Preparing the Master Server — Task Map

TABLE 10–1 Preparing the Master Server

Task	Description	For Instructions, Go To
Preparing the Master Server	Convert files to NIS maps	"How To Prepare Source Files for Conversion to NIS Maps" on page 177
Preparing the Master Server	Set up with `ypinit`	"How to Set Up the Master Server With `ypinit`" on page 188

▼ How To Prepare Source Files for Conversion to NIS Maps

Prepare the source files for conversion to NIS maps.

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

   Check the following files:

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

- shadow

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

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

4. **Check the /etc/mail/aliases file.**

   Unlike other source files, the /etc/mail/aliases file cannot be moved to another directory. This file must reside in the /etc/mail directory. Make sure the /etc/mail/aliases source file is complete by verifying that it contains all the mail aliases that you want to have available throughout the domain. Refer to the aliases man page for more information.

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

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

6. **Check to make sure that the data in all the source files is correctly formatted**

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

Preparing the Makefile

After checking the source files and copying them into the source file directory, you now need to convert those source files into the ndbm format maps that the NIS service uses. This is done automatically for you by ypinit when called on the master server, as explained in the next section, “How to Set Up the Master Server With ypinit” on page 188.

The ypinit script calls the program make, which uses the Makefile located in the /var/yp directory. A default Makefile similar to Code Example 10–1 is provided for you in the /var/yp directory and contains the commands needed to transform the source files into the desired ndbm format maps.

You can use the default Makefile as it is, or modify it if you want. (If you do modify the default Makefile, be sure to first copy and store the original default Makefile in case you need it for future use.) You might need to make one or more of the following modifications to the Makefile:

- *Nondefault maps.* If you have created your own non-default source files and want to convert them to NIS maps, you must add those source files to the Makefile.
- DIR *value.* If you want the Makefile to use source files stored in some directory other than /etc, as explained in “Source Files Directory” on page 176, you must

change the value of DIR in the Makefile to the directory that you want to use. When changing this value in the Makefile, do not indent the line.

- PWDIR *value*. If you want the Makefile to use passwd, shadow, and/or adjunct source files stored in some directory other than /etc, you must change the value of PWDIR in the Makefile to the directory that you want to use. When changing this value in the Makefile, do not indent the line.
- *Domain name resolver*. If you want the NIS server to use the domain name resolver for machines not in the current domain, comment out the Makefile line B=, and uncomment (activate) the line B= -b.

CODE EXAMPLE 10–1 Default Makefile Before Modification

```
#
# Copyright (c) 1996-1999, by Sun Microsystems, Inc.
# All rights reserved.
#
#pragma ident "@(#)Makefile 1.25 99/06/01 SMI"
#
#----
# It is somewhat confusing to note that Solaris 2.x uses /etc/auto_master
# instead of the 4.x /etc/auto.master file name because of NIS+ treating a
# "." in a special way.
#
# Set the following variable to "-b" to have NIS servers use the domain name
# resolver for hosts not in the current domain.
#B=-b
B=
DIR =/etc
#
# If the ipnodes (IPv6 hosts file) lives in a directory other than
# /etc/inet, then you’ll need to change the following line.
#
INETDIR=/etc/inet
#
# If the passwd, shadow and/or adjunct files used by rpc.yppasswdd
# live in directory other than /etc then you’ll need to change the
# following line.
# DO NOT indent the line, however, since /etc/init.d/yp attempts
# to find it with grep "^PWDIR" ...
#
PWDIR =/etc
DOM = ‘domainname‘
NOPUSH = ""
ALIASES = /etc/mail/aliases
YPDIR=/usr/lib/netsvc/yp
SBINDIR=/usr/sbin
YPDBDIR=/var/yp
YPPUSH=$(YPDIR)/yppush
MAKEDBM=$(SBINDIR)/makedbm
MULTI=$(YPDIR)/multi
REVNETGROUP=$(SBINDIR)/revnetgroup
STDETHERS=$(YPDIR)/stdethers
STDHOSTS=$(YPDIR)/stdhosts
```

(continued)

(Continuation)

```
MKNETID=$(SBINDIR)/mknetid
MKALIAS=$(YPDIR)/mkalias

CHKPIPE=  || (  echo "NIS make terminated:" $@ 1>&2; kill -TERM 0 )


k:
 @if [ ! $(NOPUSH) ]; then $(MAKE)  $(MFLAGS) -k all; \
 else $(MAKE) $(MFLAGS) -k all NOPUSH=$(NOPUSH);fi

all: passwd group hosts ipnodes ethers networks rpc services protocols \
 netgroup bootparams aliases publickey netid netmasks c2secure \
 timezone auto.master auto.home \
 auth.attr exec.attr prof.attr user.attr audit.user

c2secure:
 -@if [ -f $(PWDIR)/security/passwd.adjunct ]; then \
  if [ ! $(NOPUSH) ]; then $(MAKE)  $(MFLAGS) -k \
  passwd.adjunct.time group.adjunct.time; \
  else $(MAKE) $(MFLAGS) -k NOPUSH=$(NOPUSH) \
  passwd.adjunct.time group.adjunct.time; \
  fi; \
 fi

passwd.time: $(PWDIR)/passwd $(PWDIR)/shadow
 -@if [ -f $(PWDIR)/security/passwd.adjunct ]; then \
  (nawk ’BEGIN { FS=":"; OFS=":" } /^[a-zA-Z0-9_]/ { $$2 = "##" $$1; \
    printf "%s\t%s\n", $$1, $$0 }’ $(PWDIR)/passwd $(CHKPIPE)) | \
    $(MAKEDBM) - $(YPDBDIR)/$(DOM)/passwd.byname; \
  (nawk ’BEGIN { FS=":"; OFS=":"  } /^[a-zA-Z0-9_]/ { $$2 = "##" $$1; \
    printf "%-10d\t%s\n", $$3, $$0 }’ $(PWDIR)/passwd $(CHKPIPE)) | \
    $(MAKEDBM) - $(YPDBDIR)/$(DOM)/passwd.byuid; \
    elif [ -f $(PWDIR)/shadow ]; then \
  (nawk ’BEGIN { FS=":"; OFS=":"; while ( getline < "$(PWDIR)/shadow" > 0) \
    shadow[$$1] = $$2; } /^[a-zA-Z0-9_]/ { $$2 = shadow[$$1]; \
    printf "%s\t%s\n",$$1,$$0 }’ $(PWDIR)/passwd $(CHKPIPE))| \
    $(MAKEDBM) - $(YPDBDIR)/$(DOM)/passwd.byname; \
  (nawk ’BEGIN { FS=":"; OFS=":"; while ( getline < "$(PWDIR)/shadow" > 0) \
    shadow[$$1] = $$2; } /^[a-zA-Z0-9_]/ { $$2 = shadow[$$1]; \
    printf "%-10d\t%s\n",$$3,$$0 }’ $(PWDIR)/passwd $(CHKPIPE))| \
    $(MAKEDBM) - $(YPDBDIR)/$(DOM)/passwd.byuid; \
 else \
  (awk ’BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { print $$1, $$0 }’ \
    $(PWDIR)/passwd  $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/passwd.byname; \
  (awk ’BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { printf("%-10d ", $$3); \
    print $$0 }’ $(PWDIR)/passwd $(CHKPIPE))| $(MAKEDBM) - \
    $(YPDBDIR)/$(DOM)/passwd.byuid; \
 fi
 @touch passwd.time;
 @echo "updated passwd";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) passwd.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) passwd.byuid; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed passwd"; fi
```

(continued)

(Continuation)

```
group.time: $(DIR)/group
 @(awk ’BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { print $$1, $$0 }’ \
    $(DIR)/group $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/group.byname;
     @(awk ’BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { printf("%-10d ", $$3); \
     print $$0 }’ $(DIR)/group $(CHKPIPE)) | $(MAKEDBM) - \
     $(YPDBDIR)/$(DOM)/group.bygid;
 @touch group.time;
 @echo "updated group";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) group.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) group.bygid; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed group"; fi

ipnodes.time: $(INETDIR)/ipnodes
 @($(MULTI) -n $(B) -l $(INETDIR)/ipnodes);
 @($(STDHOSTS) -n $(INETDIR)/ipnodes $(CHKPIPE))| \
 (awk ’BEGIN { OFS="\t"; } $$1 !~ /^#/ { print $$1, $$0 }’ $(CHKPIPE)) | \
 $(MAKEDBM) $(B) - $(YPDBDIR)/$(DOM)/ipnodes.byaddr;
 @touch ipnodes.time;
 @echo "updated ipnodes";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) ipnodes.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) ipnodes.byaddr; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed ipnodes"; fi

hosts.time: $(DIR)/hosts
 @($(MULTI) $(B) -l $(DIR)/hosts);
 @($(STDHOSTS) $(DIR)/hosts $(CHKPIPE))| \
 (awk ’BEGIN { OFS="\t"; } $$1 !~ /^#/ { print $$1, $$0 }’ $(CHKPIPE)) | \
 $(MAKEDBM) $(B) - $(YPDBDIR)/$(DOM)/hosts.byaddr;
 @touch hosts.time;
 @echo "updated hosts";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) hosts.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) hosts.byaddr; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed hosts"; fi

ethers.time: $(DIR)/ethers
 @($(STDETHERS) $(DIR)/ethers $(CHKPIPE)) \
 |(awk ’{print $$1, $$0; for (i = 3;i <= NF;i++) print $$i,$$0}’ $(CHKPIPE)) \
 | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/ethers.byaddr

 @(awk ’BEGIN { OFS="\t"; } $$1 !~ /^#/ { print $$2, $$0 }’ \
    $(DIR)/ethers $(CHKPIPE)) | \
 $(MAKEDBM) - $(YPDBDIR)/$(DOM)/ethers.byname;
 @touch ethers.time;
 @echo "updated ethers";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) ethers.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) ethers.byaddr; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed ethers"; fi

networks.time: $(DIR)/networks
 @(sed -e "/^#/d" -e s/#.*$$// $(DIR)/networks $(CHKPIPE)) |( awk \
     ’{print $$1, $$0; for (i = 3;i <= NF;i++) print $$i,$$0}’ \
     $(CHKPIPE) )| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/networks.byname;
 @(awk ’BEGIN { OFS="\t"; } $$1 !~ /^#/ { print $$2, $$0 }’ \
    $(DIR)/networks $(CHKPIPE)) | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/networks.byaddr;
```

(continued)

(Continuation)

```
 @touch networks.time;
 @echo "updated networks";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) networks.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) networks.byaddr; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed networks"; fi

services.time: $(DIR)/services
 @(awk ’BEGIN { OFS="\t"; } $$1 !~ /^#/ { print $$2, $$0 }’ \
     $(DIR)/services $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/services.byname;
 @(awk ’BEGIN { OFS="\t"; } \
 $$1 !~ /^#/ { split($$2,pp,"/"); printf("%s/%s %s\n", $$1, pp[2], $$0);\
  if (seen[$$1] == "") {\
   printf("%s %s\n", $$1, $$0); seen[$$1]=$$1;} \
  for (i = 3; i <= NF && $$i !~ /^#/; i++) \
   printf("%s/%s %s\n", $$i, pp[2], $$0)}’ \
  $(DIR)/services $(CHKPIPE)) | \
 $(MAKEDBM) $(B) - $(YPDBDIR)/$(DOM)/services.byservicename

 @touch services.time;
 @echo "updated services";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) services.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) services.byservicename; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed services"; fi

rpc.time: $(DIR)/rpc
 @(awk ’BEGIN { OFS="\t"; } $$1 !~ /^#/ { print $$2, $$0 }’ \
     $(DIR)/rpc $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/rpc.bynumber;
 @touch rpc.time;
 @echo "updated rpc";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) rpc.bynumber; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed rpc"; fi

protocols.time: $(DIR)/protocols
 @(awk ’BEGIN { OFS="\t"; } $$1 !~ /^#/ { print $$2, $$0 }’ \
     $(DIR)/protocols $(CHKPIPE)) | $(MAKEDBM) - \
     $(YPDBDIR)/$(DOM)/protocols.bynumber;

 @(sed -e "/^#/d" -e s/#.*$$// $(DIR)/protocols $(CHKPIPE)) |( awk \
     ’{print $$1,$$0; for (i = 3;i <= NF;i++) print $$i, $$0}’ \
     $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/protocols.byname;

 @touch protocols.time;
 @echo "updated protocols";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) protocols.byname; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) protocols.bynumber; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed protocols"; fi

netgroup.time: $(DIR)/netgroup
 @$(MAKEDBM) $(DIR)/netgroup $(YPDBDIR)/$(DOM)/netgroup
 @($(REVNETGROUP) < $(DIR)/netgroup -u $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/
netgroup.byuser
 @($(REVNETGROUP) < $(DIR)/netgroup -h $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/
netgroup.byhost
 @touch netgroup.time;
```

(continued)

(Continuation)

```
 @echo "updated netgroup";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) netgroup; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) netgroup.byuser; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) netgroup.byhost; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed netgroup"; fi

bootparams.time: $(DIR)/bootparams
 @(sed -e '/^#/d' -e s/#.*$$// -e 's/[  ][  ]*$$//' \
     -e '/\\$$/s/\\$$/ /' $(DIR)/bootparams $(CHKPIPE))\
 |( awk '/ $$/ {printf "%s", $$0} !/ $$/ {print}' $(CHKPIPE))\
 |( sed -e 's/[  ][  ]*/ /g' $(CHKPIPE))\
 | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/bootparams;
 @touch bootparams.time;
 @echo "updated bootparams";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) bootparams; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed bootparams"; fi

aliases.time: $(ALIASES)
 @cp $(ALIASES) $(YPDBDIR)/$(DOM)/mail.aliases;
 @/usr/lib/sendmail -bi -oA$(YPDBDIR)/$(DOM)/mail.aliases;
 $(MKALIAS) $(YPDBDIR)/$(DOM)/mail.aliases $(YPDBDIR)/$(DOM)/mail.byaddr;
 @rm $(YPDBDIR)/$(DOM)/mail.aliases;
 @touch aliases.time;
 @echo "updated aliases";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) mail.aliases; fi
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) mail.byaddr; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed aliases"; fi

netmasks.time: $(DIR)/netmasks
 $(MAKEDBM) $(DIR)/netmasks $(YPDBDIR)/$(DOM)/netmasks.byaddr;
 @touch netmasks.time;
 @echo "updated netmasks";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) netmasks.byaddr; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed netmasks"; fi


publickey.time: $(DIR)/publickey
 @(sed "/^#/d" < $(DIR)/publickey $(CHKPIPE))| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/
publickey.byname;
 @touch publickey.time;
 @echo "updated publickey";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) publickey.byname; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed publickey"; fi

netid.time: $(PWDIR)/passwd $(DIR)/group $(DIR)/hosts $(DIR)/netid
 @$(MKNETID) -q -p $(PWDIR)/passwd -g $(DIR)/group -h $(DIR)/hosts -m $(DIR)/
netid > .ypjunk;
 @$(MAKEDBM) .ypjunk $(YPDBDIR)/$(DOM)/netid.byname;
 @rm -f .ypjunk;
 @touch netid.time;
 @echo "updated netid";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) netid.byname; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed netid"; fi
```

(continued)

(Continuation)

```
# Old way.  Could be restored by PSARC decision.
#
#passwd.adjunct.time: $(PWDIR)/security/passwd.adjunct
# @(awk ’BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { print $$1, $$0 }’ $(PWDIR)/security/
passwd.adjunct $(CHKPIPE)) | \
#  $(MAKEDBM) -s - $(YPDBDIR)/$(DOM)/passwd.adjunct.byname;
# @chmod 600 $(YPDBDIR)/$(DOM)/passwd.adjunct.byname.dir;
# @chmod 600 $(YPDBDIR)/$(DOM)/passwd.adjunct.byname.pag;
# @touch passwd.adjunct.time
# @echo "updated passwd.adjunct";
# @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) passwd.adjunct.byname; fi
# @if [ ! $(NOPUSH) ]; then echo "pushed passwd.adjunct"; fi

passwd.adjunct.time: $(PWDIR)/security/passwd.adjunct $(PWDIR)/shadow
 -@if [ -f $(PWDIR)/shadow ]; then \
  (nawk ’BEGIN { FS=":"; while (getline < "$(PWDIR)/shadow" > 0) \
    shadow[$$1] = $$2; } /^[a-zA-Z0-9_]/ { $$2 = shadow[$$1]; OFS=":"; \
    printf "%s\t%s\n", $$1, $$0 }’ $(PWDIR)/security/passwd.adjunct $(CHKPIPE)) | \
    $(MAKEDBM) -s - $(YPDBDIR)/$(DOM)/passwd.adjunct.byname; \
 else \
  (awk ’BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { print $$1, $$0 }’ \
    $(PWDIR)/security/passwd.adjunct $(CHKPIPE)) | \
  $(MAKEDBM) -s - $(YPDBDIR)/$(DOM)/passwd.adjunct.byname; \
 fi
 @chmod 600 $(YPDBDIR)/$(DOM)/passwd.adjunct.byname.dir;
 @chmod 600 $(YPDBDIR)/$(DOM)/passwd.adjunct.byname.pag;
 @touch passwd.adjunct.time
 @echo "updated passwd.adjunct";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) passwd.adjunct.byname; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed passwd.adjunct"; fi

group.adjunct.time: $(PWDIR)/security/group.adjunct
 @(awk ’BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { print $$1, $$0 }’ $(PWDIR)/security/
group.adjunct $(CHKPIPE)) | \
 $(MAKEDBM) -s - $(YPDBDIR)/$(DOM)/group.adjunct.byname;
 @chmod 600 $(YPDBDIR)/$(DOM)/group.adjunct.byname.dir;
 @chmod 600 $(YPDBDIR)/$(DOM)/group.adjunct.byname.pag;
 @touch group.adjunct.time
 @echo "updated group.adjunct";
 @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) group.adjunct.byname; fi
 @if [ ! $(NOPUSH) ]; then echo "pushed group.adjunct"; fi

timezone.time:  $(DIR)/timezone
 -@if [ -f $(DIR)/timezone ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/timezone \
  | awk ’{for (i = 2; i<=NF; i++) print $$i, $$0}’ \
  | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/timezone.byname; \
  touch timezone.time; \
  echo "updated timezone"; \
  if [ ! $(NOPUSH) ]; then \
   $(YPPUSH) timezone.byname; \
   echo "pushed timezone"; \
  else \
  : ; \
```

(continued)

(Continuation)

```
  fi \
 else \
  echo "couldn’t find $(DIR)/timezone"; \
 fi

auto.master.time:  $(DIR)/auto_master
 -@if [ -f $(DIR)/auto_master ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/auto_master \
  | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/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

auto.home.time:  $(DIR)/auto_home
 -@if [ -f $(DIR)/auto_home ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/auto_home \
  | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto.home; \
  touch auto.home.time; \
  echo "updated auto.home"; \
  if [ ! $(NOPUSH) ]; then \
   $(YPPUSH) auto.home; \
   echo "pushed auto.home"; \
  else \
  : ; \
  fi \
 else \
  echo "couldn’t find $(DIR)/auto_home"; \
 fi


auth.attr.time:  $(DIR)/auth_attr
 -@if [ -f $(DIR)/auth_attr ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/auth_attr \
  |sed -e ’/\\$$/{:l’ -e ’N;s/\\\n//;t h’ -e ’:h’ \
  -e ’s/\\$$/\\/;t l’ -e } \
  | (nawk ’BEGIN { FS=":"; OFS=":" } /^[a-zA-Z0-9_]/ \
  {printf "%s:%s\n", $$1, $$0 }’ $(CHKPIPE)) \
  | $(MAKEDBM) -S ":" -E - $(YPDBDIR)/$(DOM)/auth_attr; \
  touch auth.attr.time; \
  echo "updated auth_attr"; \
  if [ ! $(NOPUSH) ]; then \
   $(YPPUSH) auth_attr; \
   echo "pushed auth_attr"; \
  else \
  : ; \
  fi \
```

(continued)

(Continuation)

```
 else \
  echo "couldn’t find $(DIR)/auth_attr"; \
 fi

exec.attr.time:  $(DIR)/exec_attr
 -@if [ -f $(DIR)/exec_attr ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/exec_attr \
  |sed -e ’/\\$$/{:l’ -e ’N;s/\\\n//;t h’ -e ’:h’ \
  -e ’s/\\$$/\\/;t l’ -e } \
  | (nawk ’BEGIN { FS=":"; OFS=":" } /^[a-zA-Z0-9_]/ \
  {printf "%s:%s:%s\n", \
  $$1, $$6, $$0 }’ $(CHKPIPE)) \
  | $(MAKEDBM) -S ":" -E -D 1 - $(YPDBDIR)/$(DOM)/exec_attr; \
  touch exec.attr.time; \
  echo "updated exec_attr"; \
  if [ ! $(NOPUSH) ]; then \
   $(YPPUSH) exec_attr; \
   echo "pushed exec_attr"; \
  else \
  : ; \
  fi \
 else \
  echo "couldn’t find $(DIR)/exec_attr"; \
 fi

prof.attr.time:  $(DIR)/prof_attr
 -@if [ -f $(DIR)/prof_attr ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/prof_attr \
  |sed -e ’/\\$$/{:l’ -e ’N;s/\\\n//;t h’ -e ’:h’ \
  -e ’s/\\$$/\\/;t l’ -e } \
  | (nawk ’BEGIN { FS=":"; OFS=":" } /^[a-zA-Z0-9_]/ \
  {printf "%s:%s\n", $$1, $$0 }’ $(CHKPIPE)) \
  | $(MAKEDBM) -S ":" -E - $(YPDBDIR)/$(DOM)/prof_attr; \
  touch prof.attr.time; \
  echo "updated prof_attr"; \
  if [ ! $(NOPUSH) ]; then \
   $(YPPUSH) prof_attr; \
   echo "pushed prof_attr"; \
  else \
  : ; \
  fi \
 else \
  echo "couldn’t find $(DIR)/prof_attr"; \
 fi

user.attr.time:  $(DIR)/user_attr
 -@if [ -f $(DIR)/user_attr ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/user_attr \
  |sed -e ’/\\$$/{:l’ -e ’N;s/\\\n//;t h’ -e ’:h’ \
  -e ’s/\\$$/\\/;t l’ -e } \
  | (nawk ’BEGIN { FS=":"; OFS=":" } /^[a-zA-Z0-9_]/ \
  {printf "%s:%s\n", $$1, $$0 }’ $(CHKPIPE)) \
  | $(MAKEDBM) -S ":" -E - $(YPDBDIR)/$(DOM)/user_attr; \
  touch user.attr.time; \
```

(continued)

(Continuation)

```
  echo "updated user_attr"; \
  if [ ! $(NOPUSH) ]; then \
   $(YPPUSH) user_attr; \
   echo "pushed user_attr"; \
  else \
  : ; \
  fi \
 else \
  echo "couldn’t find $(DIR)/user_attr"; \
 fi

audit.user.time:  $(DIR)/audit_user
 -@if [ -f $(DIR)/audit_user ]; then \
  sed -e "/^#/d" -e s/#.*$$// $(DIR)/audit_user \
  |sed -e ’/\\$$/{:l’ -e ’N;s/\\\n//;t h’ -e ’:h’ \
  -e ’s/\\$$/\\/;t l’ -e } \
  | (nawk ’BEGIN { FS=":"; OFS="\t" } /^[a-zA-Z0-9_]/ \
  {print $$1, $$0 }’ $(CHKPIPE)) \
  | $(MAKEDBM) -s - $(YPDBDIR)/$(DOM)/audit_user; \
  touch audit.user.time; \
  echo "updated audit_user"; \
  if [ ! $(NOPUSH) ]; then \
   $(YPPUSH) audit_user; \
   echo "pushed audit_user"; \
  else \
  : ; \
  fi \
 else \
  echo "couldn’t find $(DIR)/audit_user"; \
 fi

passwd: passwd.time
group: group.time
hosts: hosts.time
ipnodes: ipnodes.time
ethers: ethers.time
networks: networks.time
rpc: rpc.time
services: services.time
protocols: protocols.time
netgroup: netgroup.time
bootparams: bootparams.time
aliases: aliases.time
publickey: publickey.time
netid: netid.time
passwd.adjunct: passwd.adjunct.time
group.adjunct: group.adjunct.time
netmasks: netmasks.time
timezone: timezone.time
auto.master: auto.master.time
auto.home: auto.home.time
auth.attr:auth.attr.time
exec.attr:exec.attr.time
prof.attr:prof.attr.time
```

(continued)

(Continuation)

```
user.attr:user.attr.time
audit.user:audit.user.time
$(DIR)/netid:
$(DIR)/timezone:
$(DIR)/auto_master:
$(DIR)/auto_home:
$(PWDIR)/shadow:
$(DIR)/auth_attr:
$(DIR)/exec_attr:
$(DIR)/prof_attr:
$(DIR)/user_attr:
$(DIR)/audit_user:
```

The function of the `Makefile` is to create the appropriate NIS maps for each of the databases listed under `all`. After passing through `makedbm` the data is collected in two files, `mapname.dir` and `mapname.pag`, both in the `/var/yp/`*domainname* directory on the master server.

The `Makefile` builds `passwd` maps from the `/PWDIR/passwd`, `/PWDIR/shadow`, and `/PWDIR/security/passwd.adjunct` files, as appropriate.

▼ How to Set Up the Master Server With `ypinit`

The `/usr/sbin/ypinit` shell script sets up master and slave servers and clients to use NIS. It also initially runs `make` to create the maps on the master server.

To use `ypinit` to build a fresh set of NIS maps on the master server, follow these steps:

1. **Become root on the master server and ensure that the name service gets its information from the `/etc` files, not from NIS, by typing:**

   ```
   # cp /etc/nsswitch.files /etc/nsswitch.conf
   ```

2. **Edit the `/etc/hosts` or `/etc/inet/ipnodes` file to add the name and IP address of each of the NIS servers.**

3. **To build new maps on the master server, type:**

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

4. `ypinit` **prompts for a list of other machines to become NIS slave servers. Type the name of the server you are working on, along with the names of your NIS slave servers.**

5. `ypinit` **asks whether you want the procedure to terminate at the first nonfatal error or continue despite nonfatal errors. Type** `y`**.**

   When you choose `y`, `ypinit` exits upon encountering the first problem; you can then fix it and restart `ypinit`. This is recommended if you are running `ypinit` for the first time. If you prefer to continue, you can try to manually fix all problems that occur, and then restart `ypinit`.

   Note - A nonfatal error can appear when some of the map files are not present. This is not an error that affects the functionality of NIS. You might need to add maps manually if they were not created automatically. Refer to Table 10–3 for a description of all default NIS maps.

   ---

6. `ypinit` **asks whether the existing files in the** `/var/yp/`***domainname*** **directory can be destroyed.**

   This message is displayed only if NIS has been previously installed. You must answer `yes` to install the new version of NIS.

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

   ```
   # make
   ```

   This program uses the instructions contained in the `Makefile` (either the default one or the one you modified) located in `/var/yp`. The `make` command cleans any remaining comment lines from the files you designated and runs `makedbm` on them, creating the appropriate maps and establishing the name of the master server for each map.

   If the map or maps being pushed by the `Makefile` correspond to a domain other than the one returned by the command `domainname` on the master, you can make sure that they are pushed to the correct domain by starting `make` in the `ypinit` shell script with a proper identification of the variable `DOM`, as follows:

   ```
   # make DOM=domainname password
   ```

   This pushes the `password` map to the intended domain, instead of the domain to which the master belongs.

8. **To enable NIS as the naming service, type:**

   ```
   # cp /etc/nsswitch.nis /etc/nsswitch.conf
   ```

This replaces the current switch file with the default NIS-oriented switch file. You can edit this file as necessary.

Master Supporting Multiple NIS Domains

Normally, a NIS master server supports only one NIS domain. However, if you are using a master server to support multiple domains, you must modify the steps slightly, as described in the section above, when setting up the server to serve the additional domains.

Run the `domainname` command on the server. The domain name returned by the command is the server's default domain. The steps described in the section above will work properly for setting up service for that domain. To configure service for any *other* domain, you must modify the `ypinit` shell script as follows:

```
# make DOM=correct-domain passwd
```

Where *correct-domain* is the name of the other domain that you are setting up service for, and `passwd` is the `make` target. This command pushes the `password` map to the intended domain, instead of the domain to which the master belongs.

Starting NIS Service on the Master Server

Now that the master maps are created, you can start the NIS daemons on the master server and begin service. To do this, you have to start `ypserv` on the server and run `ypbind`. When a client requests information from the server, `ypserv` is the daemon that answers information requests from clients after looking them up in the NIS maps.

There are two ways that NIS service can be started on a server:

- By automatically invoking the `/usr/lib/netsvc/yp/ypstart` script during the boot process.
- Using `ypstart` from the command line.

Starting NIS Service Automatically

After the NIS master server has been configured by running `ypinit`, `ypstart` is automatically invoked to start up `ypserve` when the machine is booted. (See "How to Set Up the Master Server With `ypinit`" on page 188.)

Starting NIS From the Command Line

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

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

Note - Because there is a slight delay before ypserv is ready to respond to calls after startup, you should issue a three to five second sleep after ypstart when calling it from inside a program or script.

DNS Forwarding

In the Solaris 7 release, if the /etc/resolv.conf file is present ypstart will start up ypserve with DNS forwarding. If you do not want the DNS forwarding option set, edit the /usr/lib/netsvc/yp/ypstart script to remove the –d option from the ypserv command. You must then reboot the machine.

For more information about using NIS with DNS, refer to Chapter 13 and *Solaris Naming Administration Guide*.

Stopping NIS with ypstop

To stop NIS service, run the ypstop command:

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

Setting Up NIS Slave Servers

Your network can have one or more slave servers. Having slave servers ensures the continuity of NIS services when the master server is not available.

Preparing a Slave Server

Before actually running ypinit to create the slave servers, you should run the domainname command on each NIS slave to make sure the domain name is consistent with the master server.

Note - Domain names are case-sensitive.

Make sure that the network is working properly before you configure an NIS slave server. In particular, check to be sure you can use `rcp` to send files from the master NIS server to NIS slaves.

Setting Up NIS Slave Servers — Task Map

TABLE 10–2 Setting Up NIS Slave Servers

Task	Description	For Instructions, Go To
Setting Up NIS Slave Servers	Set up NIS slave servers	"How to Set Up a Slave Server" on page 192

▼ How to Set Up a Slave Server

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

1. **As root, edit the** `/etc/hosts` **or** `/etc/inet/ipnodes` **file on the slave server to add the name and IP addresses of all the other NIS servers.**

2. **Change directory to** `/var/yp` **on the slave server.**

3. **To initialize the slave server as a client, type the following:**

```
# /usr/sbin/ypinit -c
```

The `ypinit` command prompts you for a list of NIS servers. Enter the name of the local slave you are working on first, then the master server, followed by the other NIS slave servers in your domain in order from the physically closest to the furthest (in network terms).

Note - You must first configure the new slave server as an NIS client so that it can get the NIS maps from the master for the first time. (See "Setting Up NIS Clients" on page 193 for details.)

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

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

If a listing is displayed, ypbind is running.

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

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

6. **Type the following to restart** ypbind**:**

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

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

```
# /usr/sbin/ypinit -s master
```

Where *master* is the machine name of the existing NIS master server.

Repeat the procedures described in this section for each machine you want configured as an NIS slave server.

Starting NIS Service on a Slave Server

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

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

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

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

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

Setting Up NIS Clients

You must perform two tasks to allow a machine to use NIS:

- Select the correct nsswitch.conf file, as described in Chapter 1.
- Configure the machine to use NIS, as explained below.

Configuring a Machine to Use NIS

The two methods for configuring a machine to use NIS as its name service are explained below.

- ypinit. The recommended method for configuring a client machine to use NIS is to login to the machine as root and run ypinit -c.

```
# ypinit -c
```

You will be asked to name NIS servers from which the client obtains name service information. You can list as many master or slave servers as you want. The servers that you list can be located anywhere in the domain. It is a better practice to first list the servers closest (in net terms) to the machine, than those that are on more distant parts of the net.

- *Broadcast method.* An older method of configuring a client machine to use NIS to log in to the machine as root, set the domain name with the domainname command, then run ypbind.

```
# domainname doc.com
# ypbind -broadcast
```

When you run ypbind, it searches the local subnet for an NIS server. If it finds one, it binds to it. This search is referred to as *broadcasting*. If there is no NIS server on the client’s local subnet, it fails to bind and the client machine is not able to obtain namespace data from the NIS service.

NIS Maps

The namespace data set used by NIS is stored in a set of NIS maps. NIS maps are essentially two-column tables.

Default NIS Maps

Table 10–3 describes the default NIS maps, information they contain, and whether the operating system consults the corresponding administrative files when NIS is running.

TABLE 10–3 NIS Map Descriptions

Map Name	Corresponding NIS Admin File	Description
bootparams	bootparams	Contains path names of files clients need during boot: root, swap, possibly others.
ethers.byaddr	ethers	Contains machine names and Ethernet addresses. The Ethernet address is the key in the map.
ethers.byname	ethers	Same as ethers.byaddr, except the key is machine name instead of the Ethernet address.
group.bygid	group	Contains group security information with group ID as key.
group.byname	group	Contains group security information with group name as key.
hosts.byaddr	hosts	Contains machine name, and IPv4 address, with IPv4 address as key.
hosts.byname	hosts	Contains machine name and IPv4 address, with machine (host) name as key.
ipnodes.byaddr	ipnodes	Contains machine name and IP address (both IPv4 and IPv6 addresses) with IP address as key.
ipnodes.byname	ipnodes	Contains machine name and IP address (both IPv4 and IPv6 addresses) with machine (host) name as key.
mail.aliases	aliases	Contains aliases and mail addresses, with aliases as key.
mail.byaddr	aliases	Contains mail address and alias, with mail address as key.
netgroup.byhost	netgroup	Contains group name, user name, and machine name.
netgroup.byuser	netgroup	Same as netgroup.byhost, except that key is user name.

TABLE 10–3 NIS Map Descriptions *(continued)*

Map Name	Corresponding NIS Admin File	Description
`netgroup`	`netgroup`	Same as `netgroup.byhost`, except that key is group name.
`netid.byname`	`passwd, hosts, group`	Used for UNIX-style authentication. Contains the netname database. If there is a `netid` file available it is consulted in addition to the data available through the other files.
`netmasks.byaddr`	`netmasks`	Contains network mask to be used with IP subnetting, with the address as the key.
`networks.byaddr`	`networks`	Contains names of networks known to your system and their IP addresses, with the address as the key.
`networks.byname`	`networks`	Same as `networks.byaddr`, except the key is the name of the network.
`passwd.adjunct. byname`	`passwd` and `shadow`	Contains auditing information and the hidden password information for C2 clients.
`passwd.byname`	`passwd` and `shadow`	Contains password information with user name as key.
`passwd.byuid`	`passwd` and `shadow`	Same as `passwd.byname`, except that the key is user ID.
`protocols.byname`	`protocols`	Contains network protocols known to your network.
`protocols.bynumber`	`protocols`	Same as `protocols.byname`, except that key is protocol number.
`rpc.bynumber`	`rpc`	Contains program number and name of RPCs known to your system. Key is RPC program number.
`services.byname`	`services`	Lists Internet services known to your network. Key is port and protocol.

TABLE 10–3 NIS Map Descriptions *(continued)*

Map Name	Corresponding NIS Admin File	Description
`services.byservice`	`services`	Lists Internet services known to your network. Key is service name.
`ypservers`	N/A	Lists NIS servers known to your network.

Modifying NIS Maps

See *Solaris Naming Administration Guide* for information on modifying NIS maps after they are created.

NIS Administration, Problem Solving, and Error Messages

See *Solaris Naming Administration Guide* for information on NIS administration, problem solving, and error messages.

PART IV FNS Setup and Configuration

This part describes how to initially set up and configure the Federated Naming Service (FNS) in an NIS+, NIS, or /etc namespace environment.

CHAPTER **11**

FNS Setup and Configuration

This chapter describes how to initially set up and configure the Federated Naming Service (FNS) in an NIS+, NIS, or files-based naming environment. (*Files-based* naming refers to name services that obtain their data from `/etc` files rather than NIS+ or NIS.)

- "Setting Up FNS—Overview" on page 201
- "Determining Resource Requirements" on page 202
- "Preparing the Namespace for FNS" on page 203
- "Creating Global FNS Namespace Contexts" on page 206
- "Replicating FNS Service" on page 209
- "FNS Administration, Problem Solving, and Error Messages" on page 212

See *Solaris Naming Administration Guide* for a general description and overview of FNS.

Note - The *Solaris Naming Administration Guide* contains an "FNS Quickstart" chapter that provides a summary overview of FNS, a brief description of setup and configuration steps, and a programming example. Experienced administrators may find that this quick start chapter is all they need.

Setting Up FNS—Overview

After your Solaris 7 release software is installed, you must perform the following tasks to set up FNS:

1. Make sure that your servers can handle FNS. See "Determining Resource Requirements" on page 202.

2. Prepare your namespace for FNS. See “Preparing the Namespace for FNS” on page 203.
3. Set up the FNS namespace contexts. There are two ways to do this:
   a. Globally create all contexts in one process. See “Creating Global FNS Namespace Contexts” on page 206.
   b. Individually create your FNS contexts. See *Solaris Naming Administration Guide*.
4. Set up FNS replica servers. See “Replicating FNS Service” on page 209.

Depending on the size of the organization, you should allow several hours for the FNS setup to be completed, plus additional time for namespace preparation.

Determining Resource Requirements

Before proceeding with any installation procedure, you must first ensure that the servers supporting FNS have sufficient memory and disk storage. Space for FNS is in addition to the space needed for your enterprise-level name service (NIS+, NIS, or files).

As a general rule-of-thumb, you will need approximately 17 Kbytes of disk storage for each user and host, plus adequate swap space. Where this disk storage space is located and how it is calculated varies according to your underlying enterprise-level naming service:

- *NIS+*. The disk storage must be mounted on the machine that will function as the FNS server for the domain or subdomain. In an NIS+ environment, a server hosting the FNS `ctx_dir` directory does not have to be the same server hosting the standard NIS+ directories, such as `org_dir`. In order to more evenly distribute server load, many large installations choose to use separate machines for NIS+ and FNS servers. The amount of space needed on an FNS server in an NIS+ environment is determined by the number of users and hosts in the domain, or subdomain, for which the server provides naming.
- *NIS*. The disk storage must be mounted on the machine that will function as the FNS server for the domain. In an NIS environment, a server hosting FNS does not have to be the same server hosting NIS. In order to more evenly distribute server load, many large installations choose to use separate machines for NIS and FNS servers. The amount of space needed on an FNS server in a NIS environment is determined by the number of users and hosts in the domain.
- *Files-based*. When your enterprise-level name service is files-based, the amount of disk storage needed by FNS is determined by the number of users and hosts in `/etc/users` and `/etc/hosts` files of the machine mounting `/var/fn`. If every machine has its own `/var/fn` directory, then the amount of space needed is determined by each machine’s user and host files. If `/var/fn` is mounted on one

machine and exported to the rest of the machines on the network by NFS, the space needed by the machine hosting `/var/fn` is determined by the number of users and hosts in that machine's `/etc/users` and `/etc/hosts` files.

For example, to support an FNS environment in an NIS+ domain with 1200 users and hosts, you will need:

- A minimum of 20 Mbytes of disk space beyond the space needed by your underlying enterprise namespace (NIS+, NIS, or files-based).
- An additional 40 Mbytes of swap space

Preparing the Namespace for FNS

This section describes the preparations you need to make before running `fncreate` to set up your FNS contexts. The preparations vary according to your enterprise-level naming service.

Preparing the Namespace for FNS — Task Map

TABLE 11–1 Preparing the Namespace for FNS

Task	Description	For Instructions, Go To
Preparing the Namespace for FNS	Convert files to NIS maps	"How To Prepare Source Files for Conversion to NIS Maps" on page 177
Preparing the Namespace for FNS	Prepare NIS service	"How to Prepare NIS Service for FNS" on page 205
Preparing the Namespace for FNS	Prepare files-based naming	"Preparing Files-Based Naming for FNS" on page 205

▼ How to Prepare NIS+ Service for FNS

Before setting up the FNS namespace, do the following:

1. **Make sure that the NIS+ domain is properly set up.**

   The NIS+ domain and associated subdomains must already be set up before configuring FNS. In other words, NIS+ standard tables, such as `hosts` and `passwd`, must already exist and be populated.

2. **Make sure that the domain's** `hosts.org_dir` **and** `passwd.org_dir` **tables are fully populated with the names of every host and user.**

   You can use the `niscat` or `nismatch` commands to check the contents of these tables.

3. **Set the** `NIS_GROUP` **environment variable to the name of the group that will be administering the FNS objects.**

   The `fncreate` command will not let you complete the FNS setup without setting this variable first. When `fncreate` creates user and host contexts, they are owned by those hosts and users, and not by the administrator who executed the command. Setting `NIS_GROUP` allows the administrators who are members of the group to subsequently modify these contexts, even though they do not own the objects.

   Assuming a C-Shell, the example below sets `NIS_GROUP` to `fns_admins.doc.com`.

   ```
   rootmaster# setenv NIS_GROUP fns_admins.doc.com
   ```

4. **[Optional] Specify that FNS run on a machine other than the NIS+ master server.**

   All NIS+ objects used by FNS are kept under the `ctx_dir` directory of an NIS+ domain, at the same level as the domain's `org_dir` directory. For large domains, such as those with more than 5000 users and hosts, it is recommended (though not required) that the `ctx_dir` used by FNS be supported by a server different from the one supporting the standard NIS+ directories, such as `groups_dir`. Using separate servers avoids placing too much load on one server. It also allows you to keep separate the administration of FNS's use of NIS+ and the administration of NIS+ itself.

   To specify that FNS be hosted by a machine that is not the NIS+ master server for the domain, you must manually create a `ctx_dir` directory object on the machine that will serve as the FNS host for the domain. (If you omit this step, FNS will be installed on the domain's NIS+ root master server.)

   To specify the machine that will become the FNS master server:

   a. **Create the** `ctx_dir` **directory for the NIS+ domain.**

      For example, to create a `ctx_dir` directory on a machine named `fns_server` in the `doc.com` domain, run the following command on the domain's master server (note the trailing dot at the end of the domain name, as shown):

      ```
      nismaster# nismkdir -m fns_server ctx_dir.doc.com.
      ```

(See *Solaris Naming Administration Guide* for more information on creating NIS+ directory objects with the nismkdir command.)

Note - If you are creating an FNS ctx_dir directory for a *subdomain*, the machine you specify as the FNS server hosting ctx_dir must reside *in* the subdomain, it cannot be a machine in the parent domain. (By contrast, a subdomain's NIS+ master server always resides in the domain *above* the one it serves.) In other words, when configuring FNS for an NIS+ subdomain, if you use the *same* server for both NIS+ and FNS, that server resides in the domain above the subdomain; but if you use *different* servers for NIS+ and FNS, the NIS+ master server resides in the domain above and the FNS server resides in the subdomain that it serves.

---

b. **Use the** nisls **command to verify that the** ctx_dir **directory has been created.**

```
rootmaster # nisls doc.com.ctx_dir
```

c. **Run** nisping **to checkpoint the directory**

```
# /usr/lib/nis/nisping -C ctx_dir.doc.com.
```

▼ How to Prepare NIS Service for FNS

Before setting up the FNS namespace, do the following:

- **Make sure that the** hosts.byname**,** user.byname**, and** printer.conf.byname **maps are complete, correct, and up to date.**

Note - You can assign a different master server for FNS maps, using the same procedure that you would to assign a different master for any other NIS map. See *Solaris Naming Administration Guide* for details.

Preparing Files-Based Naming for FNS

Files-based naming refers to name services that obtain their data from /etc files rather than NIS+ or NIS.

If you are going to install a `/var/fn` directory on each machine, as is normally the case, the steps below must be performed on each machine. If you decide to mount and export the `/var/fn` directory from one machine, the steps below need to be performed on the machine that exports `/var/fn`.

- **Make sure that the** `/etc/hosts` **and** `/etc/passwd` **files are complete and contain the names of all users and hosts.**

Creating Global FNS Namespace Contexts

This section describes how to create your namespace globally for a given enterprise or NIS+ domain.

The FNS namespace is created by the `fncreate` command.

```
# fncreate -t org org//
```

Or, alternatively:

```
# fncreate -t org org/domain/
```

Where *domain* is the name of an NIS+ domain or subdomain.

The `fncreate` command creates the default contexts for the specified organization and all its subcontexts, including contexts and subcontexts for users and hosts in the organization.

For information on how to manually create individual FNS contexts, see *Solaris Naming Administration Guide.*

Creating Global FNS Namespace Contexts — Task Map

TABLE 11–2 Globally Creating FNS Namespace Contexts

Task	Description	For Instructions, Go To
Globally Creating FNS Namespace Contexts	Create FNS namespace under NIS+	"How to Create Namespace Contexts Under NIS+" on page 207
Globally Creating FNS Namespace Contexts	Create FNS namespace under NIS	"How to Create Namespace Contexts Under NIS" on page 208
Globally Creating FNS Namespace Contexts	Create FNS namespace under files	"How to Create Namespace Contexts Under Local Files" on page 209

▼ How to Create Namespace Contexts Under NIS+

When your primary enterprise-level name service is NIS+, namespace contexts must be created separately for each NIS+ domain or subdomain in your enterprise.

- The NIS+ domain or subdomain must already exist.
- If you intend to use the same server for both NIS+ and FNS, you must run the `fncreate` command on the domain's (or subdomain's) master server. If you intend to use different servers for NIS+ and FNS, you must run the `fncreate` command on the machine that will function as the FNS server. (If you are going to use different machines, you must first prepare the FNS server, as explained in Step 4 on page 204.)
- You must have full NIS+ administration authorization (see *Solaris Naming Administration Guide* for information on NIS+ authorization).

For example, to create the contexts for the `manf.doc.com` subdomain on the `submaster` machine that is the NIS+ master server for that domain:

1. **On the subdomain master, run `fncreate` as shown below:**

```
submaster# fncreate -t org org/manf.doc.com./
```

This creates the organization context for the NIS+ `manf.doc.com.` subdomain, and contexts and associated subcontexts for all users found in that subdomain's passwd.org_dir table and all hosts found in the subdomain's hosts.org_dir table.

(If you want to use different machines for NIS+ and FNS servers, run the above command on the machine you want to use as the FNS server. See Step 4 on page 204 for information on how to prepare a non-NIS+ server to be an FNS server.)

1. **Use** nisping **to checkpoint the** ctx_dir **directory:**

```
# /usr/lib/nis/nisping -C ctx_dir.manf.doc.com.
```

Note - For a large organization with several thousand users and hosts, the initial fncreate operation can take several hours; the subsequent checkpoint can also take several hours.

▼ How to Create Namespace Contexts Under NIS

When your primary enterprise-level name service is NIS, there is only one domain for the enterprise. Namespace contexts are created for that enterprise-wide domain.

- The NIS domain must already exist.
- The fncreate command must be run by root on the FNS master server. (Normally, this would be the NIS master server, but you could choose to use a different server.)

For example, create the contexts for the doc.com domain, on the machine named fns_master, which is also the NIS master server:

♦ **On the domain master, run** fncreate **as shown below:**

```
fns_master# fncreate -t org org//
```

This creates the organization context for the NIS domain doc.com, and contexts and associated subcontexts for all users found in NIS servers's passwd map and all hosts found in the server's hosts map.

Note - After you have created your context maps, you can assign the same machine to be the master server, using the same procedure that you would to assign a different master for any other NIS map. The FNS maps all have names starting with fns_ and ending with either .ctx or .attr. See *Solaris Naming Administration Guide* for details.

How to Create Namespace Contexts Under Local Files

When your primary enterprise-level name service is files-based, namespace contexts are created for the system.

- The /etc/passwd and /etc/hosts files on the machine where the /var/fn directory resides must be clean and fully populated.

- The fncreate command must be run by root on the machine where the /var/fn directory resides.

For example, to create the contexts for the system:

- **On the machine hosting the /var/fn directory, run fncreate, as shown below:**

```
server1# fncreate -t org org//
```

This creates the organization context for the system and contexts and associated subcontexts for all users found in machine's /etc/passwd file, and all hosts found in the machine's /etc/hosts file.

Replicating FNS Service

On large or mission-critical networks where performance and reliability of FNS naming is of vital importance, FNS service should be replicated.

Replicating FNS Service — Task Map

TABLE 11–3 Replicating FNS Service

Task	Description	For Instructions, Go To
Replicating FNS Service	Replicate FNS service under NIS+	"How to Replicate FNS Under NIS+" on page 210
Replicating FNS Service	Replicate FNS service under NIS	"How to Replicate FNS Under NIS" on page 210
Replicating FNS Service	Replicate FNS service under files	"How to Replicate FNS Under Files-Based Naming" on page 212

▼ How to Replicate FNS Under NIS+

After the FNS namespace has been set up on the master server, additional replicas can be added in each domain to serve the domain's `ctx_dir` directory. Replicas enhance availability and performance of the servers.

1. **Run the `nismkdir` command on the FNS master server to add a replica for the `ctx_dir` directory.**

   For example, establish the machine `fnsrserver` as an FNS replica for the `doc.com.` domain:

   ```
   # nismkdir -s fnsrserver ctx_dir.doc.com.
   ```

2. **Checkpoint the `ctx_dir` directory with the `nisping` command.**

   ```
   # /usr/lib/nis/nisping -C ctx_dir.doc.com.
   ```

   FNS replicas should be checkpointed at regular intervals. The recommended period is every few days. The period you choose depends on how frequently changes are made to the FNS namespace.

▼ How to Replicate FNS Under NIS

After the FNS namespace has been set up on the domain master server, additional slave servers can be added to enhance availability and performance of the servers.

1. **As root, edit the** `/etc/hosts` **file on the slave server to add the name and IP addresses of all the other NIS servers.**

2. **Change directory to** `/var/yp` **on the slave server.**

3. **To initialize the slave server as a client, type the following:**

```
# /usr/sbin/ypinit -c
```

The `ypinit` command prompts you for a list of NIS servers. Enter the name of the local slave you are working on first, then the master server, followed by the other NIS slave servers in your domain in order, from the physically closest to the furthest (in network terms).

Note - You must first configure the new slave server as an NIS client so that it can get the NIS maps from the master for the first time. (See “Setting Up NIS Clients” on page 193 for details.)

---

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

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

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

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

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

6. **Type the following to restart** `ypbind`:

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

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

```
# /usr/sbin/ypinit -s master
```

Where *master* is the machine name of the existing NIS master server.

8. **Stop** `yp` **processes on the Slave Server:**

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

9. **Restart** `yp` **service:**

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

Alternatively, you can reboot the slave server and allow daemons to start automatically.

▼ How to Replicate FNS Under Files-Based Naming

There is no server replication when your primary naming service is files-based.

FNS Administration, Problem Solving, and Error Messages

See *Solaris Naming Administration Guide* for information on FNS administration, problem solving, and error messages.

PART V DNS Setup and Configuration

This part gives an overview of the Domain Name System (DNS) and describes how to setup DNS clients and servers. It has two chapters.

CHAPTER **12**

Setting Up DNS Clients

This chapter describes how to set up Domain Name System (DNS) service on client machines.

Note - One of the most common, and important, uses of DNS is connecting your network to the global Internet. In order to connect to the Internet, your network IP address must be registered with whomever is administering your parent domain. Who that administrator is varies according to your geographic location and type of parent domain. This manual does not describe how to register networks with domain administrators.

For more detailed information, see *DNS and Bind,* by Cricket Liu and Paul Albitz (O'Reilly, 1992).

Solaris DNS BIND Implementation

For your convenience, the Solaris 7 release supplies a compiled version of Berkeley Internet Name Domain (BIND) version 4.9.4, Patch-Level 1. In compiling this software, options and choices were made to meet the needs of the greatest number of sites. If this pre-compiled version of BIND does not meet your requirements, you can recompile your own version of BIND from the publicly available source code.

In compiling the BIND version supplied with the Solaris 7 release, the following choices were made:

- *RFC1535*. Not implemented because doing so would remove implicit search lists.
- *Inverse Queries*. Enabled because SunOS 4.x `nslookup` does not work without them.
- *Bogus Name Logging*. Logging of bogus name servers is not implemented because it produces too many unimportant messages.
- *Default Domain Name*. If the DNS domain name is not set in `/etc/resolv.conf`, or via the `LOCALDOMAIN` environment variable, `libresolv` derives it from the NIS or NIS+ domain name, provided that the `/etc/nsswitch.conf` file contains `nisplus` or `nis` as the first element in the `hosts` line.
- *Utility Scripts*. The BIND utility scripts are not included in this Solaris release.
- *Test Programs*. The BIND test programs `dig`, `dnsquery`, and `host` are not included in this Solaris release because their purpose is similar to that of `nslookup` and `nstest`.

Setting Up DNS Service

Setting up DNS service is accomplished in two basic steps:

1. Set up DNS service on your client machines. This chapter describes how to do this.
2. Set up your DNS servers as described in Chapter 13.

Client Set Up

Setting up DNS on a client machine involves two tasks:

- Creating the `/etc/resolv.conf` file, as described in“Creating the `resolv.conf` File” on page 217.
- Modifying the `/etc/nsswitch.conf` file, as described in “Enabling a Machine to Use DNS” on page 27.

If you are setting up DNS service on a host that will function as a DNS server, you also need to set up boot and data files, as described in Chapter 13.

The Resolver

DNS clients use the dynamic library routines, collectively called the *resolver*, to locate a remote host. The resolver queries the DNS database on a name server, which eventually returns the host name or IP address of the machine requested by the resolver. Because DNS name servers are clients of servers outside their local domains, they must also run the resolver.

The DNS name server uses several files to load its database. At the resolver level, it needs the file `/etc/resolv.conf` listing the addresses of the servers where it can obtain its information. The resolver reads this `resolv.conf` file to find the name of the local domain and the location of name servers. It sets the local domain name and instructs the resolver routines to query the listed name servers for information. Normally, each DNS client system on your network has a `resolv.conf` file in its `/etc` directory. (If a client does not have a `resolv.conf` file, it defaults to using a server at IP address `127.0.0.1`.)

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

Creating the `resolv.conf` File

A simple example `resolv.conf` file for a client (non-server) machine in the `doc.com` domain is shown in Code Example 12–1:

CODE EXAMPLE 12–1 Sample `resolv.conf` File

```
; Sample resolv.conf file for the machine polaris
domain doc.com
; try local name server
nameserver 127.0.0.1
; if local name server down, try these servers
nameserver 123.45.6.1
nameserver 111.22.3.5
; sort the addresses returned by gethostbyname(3c)
sortlist
130.155.160.0/255.255.240.0
```

(continued)

(Continuation)

```
130.155.0.0
```

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

```
domain domainname
```

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

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

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

```
nameserver 127.0.0.1
```

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

```
nameserver IP_address
```

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

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

```
sortlist
addresslist
```

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

Modifying the `/etc/nsswitch.conf` File

How you enable a machine to use DNS depends on your underlying enterprise-level name service:

- *NIS+*. If your primary enterprise-level name service is NIS+, follow the directions described in "Enabling a Machine to Use DNS" on page 27.
- *NIS*. If your primary enterprise-level name service is NIS, and it is correctly set up, do nothing. With proper configuration, NIS is ready to use DNS from the time it is installed.
- *Files-based*. If your primary enterprise-level name service is based on `/etc` files, follow the directions described in "Enabling a Machine to Use DNS" on page 27.

For additional information on the `nsswitch.conf` file, see *Solaris Naming Administration Guide*.

CHAPTER **13**

Setting Up DNS Servers

This chapter describes how to set up a Domain Name System (DNS) name server.

Setting Up DNS Servers

▼ How to Set Up DNS Servers

To set up a DNS server:

1. **Set the server up as a DNS client (this includes setting up the server's** `resolv.conf` **file). See Chapter 12.**

2. **Set up the boot file. See "The `named.conf` File" on page 227.**

3. **Set up the data files See "Setting Up the Data Files" on page 233. You need to set up four data files:**

   a. **The `named.ca` file. See "Setting Up the `named.ca` File" on page 234.**

   b. **The `hosts` file. See "Setting Up the `hosts` File" on page 237.**

   c. **The `hosts.rev` file. See "Setting Up the `hosts.rev` File" on page 238.**

   d. **The `named.local` file. See "Setting Up the `named.local` File" on page 239.**

4. **Initialize the server. See "Initializing the Server" on page 239.**

5. **Test the server. See "Testing Your Installation" on page 240.**

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

Server Configuration and Data File Names

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

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

Configuration File

The master server configuration file is `/etc/named.conf.` (See "The `named.conf` File" on page 227.) The configuration file contains a list of domain names and the file names containing host information. (See *Solaris Naming Administration Guide* for additional information on the `named.conf` file.)

Names of DNS Data Files

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

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

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

For reference purposes, Table 13–1 compares BIND file names from these three sources:

TABLE 13–1 File Name Examples

Solaris Names	O'Reilly Names or other names	U.C. Berkeley Names	Content and Purpose of File
`/etc/named.conf`, same file name for all three sources			BIND 8.1 adds a new `named.conf` file to replace the earlier `named.boot` file. This configuration file adds security, startup options, logging. It specifies the type of server it is running on and selectively applies options on a per-zone or per-server basis, rather than all zones or servers. It contains a list of domain names and the names of the data files.
`/etc/resolv.conf`, same file name for all three sources			This file resides on every DNS client (including DNS servers) and designates the servers that the client queries for DNS information.

TABLE 13–1 File Name Examples *(continued)*

Solaris Names	O'Reilly Names or other names	U.C. Berkeley Names	Content and Purpose of File
`named.ca`	`db.cache` `db.root`	`root.cache`	This file establishes the names of root servers and lists their addresses.
Generic: `hosts` Examples: `db.doc`, `db.sales`	Generic: `db.domain` Examples: `db.movie`, `db.fx`	Generic: `hosts` Example: `ucbhosts`	This file contains all the data about the machines in the local zone that the server serves.
Generic: `hosts.rev` Examples: `doc.rev`	Generic: `db.ADDR` Examples `db.192.249.249` `db.192.249.253`	`hosts.rev`	This file specifies a zone in the `in-addr.arpa.` domain, a special domain that allows reverse (address-to-name) mapping.
`named.local`	Generic: `db.cache` Example: `db.127.0.0`	`named.local`	This file specifies the address for the local loopback interface, or local host.
`$INCLUDE` files, same convention for all three sources			Any file identified by an `$INCLUDE()` statement in a data file.

Data Files

The required data files are:

- `/var/named/named.ca`. (See "Setting Up the `named.ca` File" on page 234 and *Solaris Naming Administration Guide* for additional information on the `named.ca` file.) So long as you are internally consistent, you can name this file anything you want.
- `/var/named/hosts`. (See "Setting Up the `hosts` File" on page 237 and *Solaris Naming Administration Guide* for additional information on `hosts` files.)

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

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

- /var/named/hosts.rev. See "Setting Up the hosts.rev File" on page 238 and *Solaris Naming Administration Guide* for additional information on the hosts.rev file.)

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

- /var/named/named.local. See "Setting Up the named.local File" on page 239 and *Solaris Naming Administration Guide* for additional information on the named.local file.) So long as you are internally consistent, you can name this file anything you want.

$INCLUDE Files

An include file is any file named in a $INCLUDE( ) statement in a DNS data file. $INCLUDE files can be used to separate different types of data into multiple files for your convenience. (See *Solaris Naming Administration Guide* for additional details.)

Domain Names

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

Default Domain Name

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

- If the resolv.conf file is not available, or does not identify a default domain, and if your enterprise-level name service is either NIS+ or NIS, the Sun implementation of DNS obtains the default domain name from those services.
- If resolv.conf is not available or does not provide a domain name and you are *not* running either NIS+ or NIS, you must either provide a resolv.conf file on each machine that does specify the domain (see "The Resolver" on page 217), or set the LOCALDOMAIN environment variable.

Trailing Dots in Domain Names

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

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

`resolv.conf` File

The following discussion describes how to set up the `resolv.conf` file.

▼ How to Set Up the `resolv.conf` File

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

CODE EXAMPLE 13–1 Sample `resolv.conf` File for DNS Server

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

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

```
domain domainname
```

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

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

The second line identifies the server itself in the form:

```
nameserver 0.0.0.0
```

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

```
nameserver IP_address
```

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

The `named.conf` File

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

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

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

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

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

TABLE 13–2 `named.conf` Statements

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

CODE EXAMPLE 13–2 Example Master Configuration File for a Primary Server

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

logging {
        category queries { default_syslog; };
};
```

(continued)

(Continuation)

```
include "/var/named/abcZones.conf"


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

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

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

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


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

Migration From BIND 4.9.x to BIND 8.1

Become superuser and run the Korn shell script, `/usr/sbin/named-bootconf`, to convert a BIND 4.9.x `named.boot` file to a BIND 8.1 `named.conf` file. See `named-bootconf`(1M).

Note - The `named.boot` and `named.conf` files cannot coexist in the same server.

Specifying Server Function

DNS servers perform one or more functions:

- *Zone primary master server*. Each zone has one server that is designated as the *primary* master server for that zone. A zone's primary master server is the *authoritative* server for that zone. (See "Specifying a Primary Master Server" on page 230.)
- *Zone secondary master server*. A zone can have one or more *secondary* master servers. Secondary master servers obtain their DNS from the zone's primary master server. You do not modify data files on a secondary server; you modify the data files on the zone's primary server and the secondary servers update their files from the primary. (See "Specifying a Secondary Master Server" on page 232.)
- *Caching-Only Server*. All servers are caching servers in the sense that they all maintain a cache of DNS data. A caching-only server is a server that is not a master server for any zone other than the `in-addr.arpa.` domain. (See "Specifying a Cache-Only Server" on page 233.)
- *Root Domain servers*. If your network is connected to the Internet, your root domain servers are out on the Internet itself and all you have to do is provide their Internet IP addresses in the `named.ca` file, as explained in "Setting Up the `named.ca` File" on page 234. If your network is not connected to the Internet, you have to set up your own root domain server, as explained in "Setting Up a Non-Internet Root Master" on page 243.

These various server functions can be performed by the same machine. For example, a machine can be a primary master server for one zone and a secondary master server for another zone. When this manual refers to a primary, secondary, or caching-only server, it is not referring to a particular machine, but the role that machine plays for a given zone.

Refer to *Solaris Naming Administration Guide* for additional information on these different server functions.

Specifying a Primary Master Server

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

1. **Create the primary record for the zone.**

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

- The first field designates the server as “primary.”
- The second field identifies the zone it serves.
- The third field identifies the hosts file.

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

```
primary    doc.com    db.doc
```

2. **Create a primary record for the zone’s reverse map.**

This record designates the server as a primary server for the zone’s reverse address map (that is, the reverse address domain for doc.com), and tells the server where to find the authoritative hosts file. This record has three fields; the first field designates the server as “primary,” the second field identifies the zone, and the third field identifies the hosts.rev file.

The reverse address domain for a zone contains the zone’s IP address in reverse order followed by in-addr.arpa. For example, suppose that the doc.com zone’s IP address is 123.45.6. In that case, the reverse address domain would be 6.45.123.in-addr.arpa.

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

```
primary   6.45.123 .   in-addr.arpa    doc.rev
```

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

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

Note - Loopback hosts are always identified as 0.0.127.in-addr.arpa.

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

```
primary  0.0.127.in-addr.arpa   named.local
```

Specifying a Secondary Master Server

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

A "secondary" record has three required fields:

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

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

- *Secondary servers*. After the IP address of the primary server, you can add IP addresses of other secondary servers. These provide additional sources from which the secondary server can obtain data. Adding IP addresses of secondary servers may under some circumstances reduce performance, unless those IP addresses are additional network addresses of a multihome primary server.
- *Backup file*. After the IP address of the primary (and optional secondary) server(s), you can add the name of a backup hosts file. If a backup file name is present, the secondary server loads its data from that file, then checks with the primary (and optional secondary) servers to make sure that the data in the backup file is up to date. If the backup file is not up to date, it is brought up to date, based on the information received from the primary server.

For example, the following lines in a boot file specify that the server is the secondary server for the doc.com zone and its reverse address domain; that it obtains its authoritative data from the primary server with an IP address of 129.146.168.119, that it uses the server 192.146.168.38 as a secondary source of zone data, and initially loads its data from the file doc.com.bakup:

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

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

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

Specifying a Cache-Only Server

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

Code Example 13–3 is a sample boot file for a caching-only server.

CODE EXAMPLE 13–3 Sample Master Boot File for Caching-only Server

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

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

A cache-only server requires:

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

Setting Up the Data Files

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

See *Solaris Naming Administration Guide* for a description of resource records, their formats, the fields they contain, special characters, and control entries.

Resource Record Types

The most commonly used types of resource records are listed in Table 13–3. They are usually entered in the order shown in Table 13–3, but that is not a requirement.

TABLE 13–3 Commonly Used Resource Record Types

Type	Description
SOA	Start of authority
NS	Name server
A	IPv4 Internet address (name to address)
AAAA	IPv6 Internet address (name to address)
PTR	Pointer (address to name)
CNAME	Canonical name (nickname)
TXT	Text information
MX	Mail exchanger

See *Solaris Naming Administration Guide* for detailed descriptions of these resource record types.

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

Setting Up the `named.ca` File

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

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

Internet `named.ca` File

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

- Anonymous FTP. The FTP site is: `ftp.rs.internic.net`. The file name is: `/domain/named.root`.

- Gopher. The Gopher site is: `rs.internic.net`. The file is: `named.root`, which can be found under the InterNIC Registration Services menu, InterNIC Registration Archives submenu.

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

CODE EXAMPLE 13–4 Example Internet `named.ca` file

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

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

(continued)

```
; temporarily housed at ISI (IANA)
.                        3600000    NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000    A    198.32.65.12
; End of File
```

Non-Internet named.ca File

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

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

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

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

For example, suppose that in addition to ourroot you also had DNS name servers called ourprimary and oursecondary. The named.ca files on all of your DNS servers would then look like this:

CODE EXAMPLE 13–5 Sample named.ca File (Non-Internet)

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

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

See "Setting Up a Non-Internet Root Master" on page 243 for a more complete discussion of setting up a domain that is not connected to the Internet.

Setting Up the hosts File

The hosts file contains all the data about every machine in your zone. If a zone covers more than one domain, all machines in all the domains covered by the zone are listed in the zone's host file (see "Setting Up Subdomains—Same Zone" on page 242).

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

There must be a separate, uniquely named, hosts file for each zone. If you have more than one zone, each zone's host file must include information about the master (primary and secondary) servers of the other zones, as described in "Setting Up Subdomains—Different Zones" on page 242.

CODE EXAMPLE 13–6 Sample hosts File

```
;
; SOA rec
doc.com  IN SOA sirius.doc.com sysop.centauri.doc.com (
                 1997071401       ;  serial number (YYYYMMDD##)
                      10800       ;  refresh every 3 hours
                      10800       ;  retry every 3 hours
                      604800      ;  expire after a week
                      86400 )     ;  TTL of 1 day
; Name Servers
doc.com                   IN  NS  sirius.doc.com
sales.doc.com             IN  NS  altair.sales.doc.com
; Addresses
localhost                 IN  A  127.0.0.1

sirius                    IN  A  123.45.6.1
rigel                     IN  A  123.45.6.112
antares                   IN  A  123.45.6.90
polaris                   IN  A  123.45.6.101
procyon                   IN  A  123.45.6.79
tauceti                   IN  A  123.45.6.69
altair.sales.doc.com      IN  A   111.22.3.4
; aliases
durvasa                   IN  CNAME sirius.doc.com
dnsmastr                  IN  CNAME sirius.doc.com
dnssales                  IN  CNAME altair.sales.doc.com
```

A hosts file usually contains these elements:

- A Start of Authority (SOA) record

- One or more Name Server (NS) records identifying primary and secondary DNS name servers
- Address (A) records for each host in the zone
- Canonical Name (CNAME) records for each host alias in the zone
- One or more Mail Exchange (MX) records

(See *Solaris Naming Administration Guide* for detailed descriptions of these resource record types.)

Setting Up the hosts.rev File

The hosts.rev file sets up inverse mapping.

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

CODE EXAMPLE 13–7 Sample hosts.rev File

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

A hosts.rev file contains these elements:

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

(See *Solaris Naming Administration Guide* for detailed descriptions of these resource record types.)

Setting Up the `named.local` File

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

CODE EXAMPLE 13–8 Sample `named.local`File

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

A `named.local` file contains these elements:

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

See *Solaris Naming Administration Guide* for detailed descriptions of these resource record types.

Initializing the Server

To initialize a server:

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

2. **Run** `in.named`**.**

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

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

Testing Your Installation

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

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

   See *Solaris Naming Administration Guide* for common DNS error messages and troubleshooting information.

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

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

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

3. **Look up a remote domain name with `nslookup`.**

   If your network is connected to the Internet, look up the name of a remote domain. (If your network is not connected to the Internet, look up the name of a subdomain in another zone, if you have one.) For example, to look up the name of the remote `internic.net` Internet domain, you would enter:

```
dnsmaster% nslookup internic.net
Server:  dnsmaster.doc.com
Address: 192.146.168.
Name:  internic.net
 Addresses: 198.41.0.9,  198.41.0.6,  198.41.0.5,  198.41.0.8
```

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

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

4. **Look up a host name in your domain from a remote domain.**

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

   For example, to look up the name of a host in your domain, from a remote Internet domain, you would enter two arguments after the nslookup command: First the name of the host you are searching for, and second, the name of the name server you are testing:

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

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

Setting Up Subdomains

- *Same zone.* The easiest method is to include the subdomain in the parent domain's zone. In this way, one set of DNS servers and data files applies to all the machines regardless of their domain. See "Setting Up Subdomains—Same Zone" on page 242.

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

- *Different zones*. You can place different domains in different zones. This is more complex because you have to specify how clients in one zone obtain DNS information about hosts in another zone. See "Setting Up Subdomains—Different Zones" on page 242.

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

Setting Up Subdomains—Same Zone

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

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

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

Setting Up Subdomains—Different Zones

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

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

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

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

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

The example files in Table 13–4 illustrate a network with two zones.

Setting Up a Non-Internet Root Master

If your network *is* connected to the Internet, your root domain server exists at the root domain Internet site; all you need to do is provide that site's Internet IP addresses in your cache file, as explained in "Internet `named.ca` File" on page 234.

If your network *is not* connected to the Internet, you must set up primary and secondary name servers in the root-level domain on your local network. This enables all domains in your network to have a consistent authoritative server to which to refer; otherwise, machines might not be able to resolve queries.

For example, suppose your non-Internet domain is named `private` and you designate the machine `ourroot` as your root server. The `ourroot` machine has an IP address of `192.1.1.10`. You would then perform the following steps:

1. **Create `named.ca` files on your primary master servers that point to your own root server.**

   For example:

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

A named.ca file must also contain NS records for each internal domain and subdomain and A records for each server as described in “Setting Up a Non-Internet Root Master” on page 243.

2. **Add a cache resource record for the root domain to the boot files of all DNS name servers on your network.**

   For example:

   ```
   cache      .     named.ca
   ```

   (See “Configuration File” on page 223 for more information.)

3. **Remove the root domain cache resource record from the boot file of your root server.**

   In other words, delete the `cache . named.ca` line from `ourroot`’s boot file.

4. **Insert a `primary` line for the root domain in the root server’s boot file.**

   For example, in `ourroot`’s boot file you would add the line:

   ```
   primary    .  ourroot.private
   ```

Caution - If you later decide to connect your network to the Internet, you *must* replace all the named.ca files on all of your servers with the current Internet files and replace the `primary .` *rootserver* line with a `cache . named.ca` line in the former root server’s boot file.

A Practical Example

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

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

This practical example assumes:

- An environment connected to the Internet

- Two networks, each with its own domain (doc.com and sales.doc.com) and its own DNS zone
- The doc.com domain and zone is the top zone over the sales.doc.com subdomain and zone.
- Each network has its own network number

TABLE 13–4 Example Network Domain and Zone Configuration

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

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

TABLE 13–5 Example Network DNS Servers

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

Example Boot Files

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

CODE EXAMPLE 13–9 Example Boot File for `dnsmastr` Server

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

CODE EXAMPLE 13–10 Example Boot File for `dnssales` Server

```
; named.boot file on the dnssales (altair)
;
; in.named is located here
directory /var/named
; here are the names of the primary files
cache      .                        named.ca
primary    sales.doc.com            db.sales
primary    0.0.127.in-addr.arpa     db.127.0.0
primary    3.22.111.in-addr.arpa    db.192.168.8
```

CODE EXAMPLE 13–11 Example Boot File for `dnssecond` Server

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

Example `resolv.conf` Files

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

CODE EXAMPLE 13–12 Example `resolve.conf` File for `dnsmastr` Server

```
;
; /etc/resolv.conf file for dnsmaster (sirius)
```

(Continuation)

```
;
domain           doc.com
nameserver       0.0.0.0
nameserver       111.22.3.5
```

CODE EXAMPLE 13–13 Example `resolve.conf` File for `dnssales` Server

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

CODE EXAMPLE 13–14 Example `resolve.conf` File for `dnssecond` Server

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

Example `named.local` File

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

CODE EXAMPLE 13–15 Example `named.local` File for Both Primary Servers

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

Example hosts Files

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

CODE EXAMPLE 13–16 Example db.doc File for dnsmastr server

```
; SOA rec
doc.com.  IN SOA sirius.doc.com. sysop.centauri.doc.com. (
                           19970332    ; serial number
                           10800       ; refresh every 3 hours
                           10800       ; retry every 3 hours
                           604800      ; expire after a week
                           86400 )     ; TTL of 1 day
; Name Servers
doc.com.                IN  NS  sirius.doc.com.
sales.doc.com.          IN  NS  altair.sales.doc.com.
; Addresses
localhost               IN  A  127.0.0.1
sirius                  IN  A  123.45.6.1
rigel                   IN  A  123.45.6.112
antares                 IN  A  123.45.6.90
polaris                 IN  A  123.45.6.101
procyon                 IN  A  123.45.6.79
tauceti                 IN  A  123.45.6.69
altair.sales.doc.com.   IN  A   111.22.3.4
; aliases
dnsmastr                IN  CNAME    sirius.doc.com.
dnssecond.doc.com       IN  CNAME   deneb.doc.com
```

CODE EXAMPLE 13–17 Example db.sales File for dnssales server

```
; SOA rec
sales.doc.com.  IN SOA altair.sales.doc.com. sysop.polaris.doc.com. (
                           19970332    ; serial number
                           10800          ; refresh every 3 hours
                           10800          ; retry every 3 hours
                           604800        ; expire after a week
                           86400 )        ; TTL of 1 day
; Name Servers
doc.com.                IN  NS  sirius.doc.com.
sales.doc.com.          IN  NS  altair.sales.doc.com.
; Addresses
altair                  IN  A  111.22.3.4
localhost               IN  A  127.0.0.1
sirius.doc.com.         IN  A  123.45.6.1
luna                    IN  A  192.168.8.22
phoebus                 IN  A  192.168.8.24
deimos                  IN  A  192.168.8.25
ganymede                IN  A  192.168.8.27
europa                  IN  A  192.168.8.28
```

(continued)

(Continuation)

```
callisto                IN  A  192.168.8.29
;
; aliases
dnssales.sales.doc.com  IN  CNAME    altair.sales.doc.com
```

Example hosts.rev Files

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

CODE EXAMPLE 13–18 Example `doc.rev` File for `dnsmastr` server

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

CODE EXAMPLE 13–19 Example `hosts.rev` File for `dnssales` Server

```
; SOA rec
3.22.111.in-addr.arpa.  IN SOA altair.sales.doc.com. sysop.polaris.doc.com. (
                           19970331    ; serial number
                           10800       ; refresh every 3 hours
                           10800       ; retry every 3 hours
                           604800      ; expire after a week
                           86400 )     ; TTL of 1 day
; Name Servers
3.22.111.in-addr.arpa.   IN  NS  altair.sales.doc.com.
;Pointer records for 111.22.3
22                       IN  PTR  luna
23                       IN  PTR  deneb
24                       IN  PTR  phoebus
25                       IN  PTR  deimos
```

(continued)

(Continuation)

```
26                        IN  PTR  altair
27                        IN  PTR  ganymede
28                        IN  PTR  europa
29                        IN  PTR  callisto
```

Example `name.ca` File

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

CODE EXAMPLE 13–20 Example `named.ca` File

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

E.ROOT-SERVERS.NET.       3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
.                         3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.       3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
.                         3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.       3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
.                         3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.       3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
.                         3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.       3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
.                         3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.       3600000      A     198.41.0.10
```

(continued)

(Continuation)

```
;
; temporarily housed at NSI (InterNIC)
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     198.41.0.11
;
; temporarily housed at ISI (IANA)
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; temporarily housed at ISI (IANA)
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     198.32.65.12
; End of File
```

Index