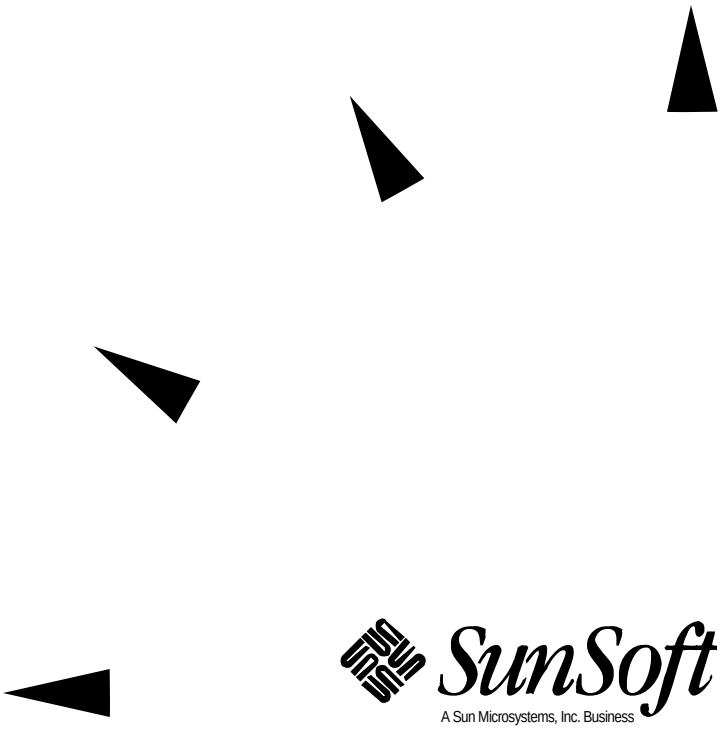
NIS+ and DNS Setup and Configuration Guide



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

© 1995 Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX® system, licensed from UNIX Systems Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's Suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS
Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, NFS, AdminTool, and AdminSuite are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. OPEN LOOK is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc.

All SPARC trademarks are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. SPARCcenter, SPARCcluster, SPARCompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC-II, and UltraSPARC are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

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

X Window System is a trademark of X Consortium, Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN, THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAMS(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Please
Recycle

Contents

Preface xix

1. Getting Started With NIS+ 1

NIS+ Overview 1

Setting Up NIS+ 2

Before You Start NIS+ 3

Planning Your NIS+ Layout 3

Determining Server Requirements 4

Disk Space and Memory Recommendations 4

Preparing the Existing Namespace 5

Configuration Worksheets 7

2. NIS+ Setup Scripts—Introduction 13

About the NIS+ Scripts 13

What the NIS+ Scripts Will Do 14

What the NIS+ Scripts Won't Do 14

3. Setting Up NIS+ With Scripts 17

NIS + Setup Overview 19
Creating a Sample NIS+ Namespace 20
Summary of NIS+ Scripts Command Lines 22
Setting Up NIS+ Root Servers 24
Prerequisites to Running `nisserver` 25
Information You Need 25
▼ How to Create a Root Master Server 25
▼ How to Change Incorrect Information 28
Populating NIS+ Tables 30
Prerequisites to Running `nispopulate` 30
Information You Need 32
▼ How to Populate the Root Master Server Tables 32
Setting Up Root Domain NIS+ Client Machines 39
Prerequisites to Running `nisclient` 39
Information You Need 40
▼ How to Initialize a New Client Machine 40
Creating Additional Client Machines 42
Initializing NIS+ Client Users 42
Prerequisites to Running `nisclient` 42
Information You Need 43
▼ How to Initialize an NIS+ User 43
Setting Up NIS+ Servers 44
Prerequisites to Running `rpc.nisd` 44
Information You Need 45

Configuring a Client as an NIS+ Server 45
▼ How to Configure a Server Without NIS Compatibility 45
▼ How to Configure a Server *With* NIS Compatibility 45
▼ How to Configure a Server With DNS and NIS Compatibility 45
Creating Additional Servers 46
Designating Root Replicas 46
Prerequisites to Running `nisserver` 46
Information You Need 47
▼ How to Create a Root Replica 47
Creating Additional Replicas 49
Creating a Subdomain 49
Prerequisites to Running `nisserver` 49
Information You Need 50
▼ How to Create a New NonRoot Domain 50
Creating Additional Domains 52
Populating the New Domain's Tables 52
Prerequisites to Running `nispopulate` 53
Information You Need 54
Populating the Master Server Tables 55
▼ How to Populate the Tables From Files 56
▼ How to Populate the Tables From NIS Maps 56
Designating Replicas 56
Prerequisites to Running `nisserver` 57

Information You Need 57
▼ How to Create a Replica 58
Initializing Subdomain NIS+ Client Machines. 58
Prerequisites to Running `nisclient` 59
Information You Need 59
▼ How to Initialize a New Subdomain Client Machine . . 60
Initializing Subdomain NIS+ Client Users 60
Prerequisites to Running `nisclient` 60
Information You Need 61
▼ How to Initialize an NIS+ Subdomain User. 61
Summary of Commands for the Sample NIS+ Namespace . . . 62
4. Setting Up the Root Domain 67
Standard versus NIS-Compatible Setup Procedures 68
Establishing the Root Domain 68
Summary of Steps 68
▼ How to Set Up a Root Domain. 70
Root Domain Setup Summary 87
5. Setting Up NIS+ Clients 89
Client Setup 90
▼ How to Set Up an NIS+ Client 92
Changing a Workstation's Domain 96
Specifying a Domain Name After Installation 97
▼ How to Change a Client's Domain Name 97
Initializing an NIS+ Client 98

Broadcast Initialization 99
Security Considerations. 99
Prerequisites 99
Information You Need 99
▼ How to Initialize a Client—Broadcast Method . . 99
Host-Name Initialization 100
Security Considerations. 100
Prerequisites 100
Information You Need 100
▼ How to Initialize a Client—Host-name Method . 100
Cold-Start File Initialization 101
Security Considerations. 101
Prerequisites 101
Information You Need 101
▼ How to Initialize a Client—Cold-Start Method . . 102
NIS+ Client Setup Summary 103
6. Setting Up NIS+ Servers 105
Setting Up an NIS+ Server 105
Standard versus NIS-Compatible Setup Procedures 106
▼ How to Set Up an NIS+ Server 107
Adding a Replica to an Existing Domain 109
▼ How to Add a Replica Server. 110
Server Setup Summary 112
7. Setting Up a Nonroot Domain 113

Setting Up a Nonroot Domain 113
Standard versus NIS-Compatible Setup Procedures 114
▼ How to Set Up a Nonroot Domain 116
Subdomain Setup Summary 122
8. Setting Up NIS+ Tables 123
Populating Tables—Options 124
Populating NIS+ Tables From Files 125
Security Considerations 125
Prerequisites 126
Information You Need 126
▼ How to Populate NIS+ Tables From Files 127
Populating NIS+ Tables From NIS Maps 131
Security Considerations 132
Prerequisites 132
Information You Need 132
▼ How to Populate Tables From Maps 133
Transferring Information From NIS+ to NIS 136
Security Considerations 137
Prerequisites 137
▼ How to Transfer Information From NIS+ to NIS 137
Limiting Access to the Passwd Column to Owners and Administrators 137
Security Considerations 138
Prerequisites 138

Information You Need 138
▼ How to Limit Read Access to the Passwd Column 138
Table Population Summaries 140
9. Setting Up the Name Service Switch 143
Selecting an Alternate Configuration File 143
▼ How to Select an Alternate Configuration File 144
Enabling an NIS+ Client to Use DNS 145
▼ How to Enable an NIS+ Client to Use DNS 145
Adding Compatibility With +/- Syntax 146
▼ How to Add DNS Compatibility With +/- Syntax 147
10. Introduction to DNS 151
DNS Basics 152
Name-to-Address Resolution 152
DNS Administrative Domains 154
`in.named` and DNS Name Servers 155
DNS Clients and the Resolver 156
Introducing the DNS Namespace 156
DNS Namespace Hierarchy 156
DNS Hierarchy in a Local Domain 157
DNS Hierarchy and the Internet 158
Joining the Internet 160
Domain Names 160
Administering DNS 162
Zone s 162

Reverse Mapping 163
The `in.addr.arpa` Domain 163
Master Servers 164
Primary Name Server 164
Secondary Name Server 164
Root Domain Name Server 165
Caching and Caching-Only Servers 165
How DNS Affects Mail Delivery 165
11. Setting Up DNS Clients 167
Creating the resolv.conf File 167
Format of `/etc/resolv.conf` 168
Modifying the /etc/nsswitch.conf File 168
12. Setting Up DNS Servers 169
Introduction to Boot and Data Files 170
The `named.boot` File 170
The `named.ca` File 171
The `hosts` File 171
The `hosts.rev` File 171
The `named.local` File 171
Setting Up the Boot File 172
The `directory` Line 172
The cache Line 173
The primary Lines 173
Setting Up the Data Files 175

Standard Resource Record Format 175
The *name* Field 176
The *ttl* Field 176
The *class* Field 176
The *record-type* Field 177
The *record-specific-data* Field 177
Special Resource Record Characters 177
Control Entries 178
Resource Record Types 179
Setting Up the `named.ca` File 180
Setting Up the `hosts` File 180
SOA— Start of Authority 182
NS—Name Server 184
A—Address 184
HINFO—Host Information 185
WKS—Well-Known Services 186
CNAME—Canonical Name 186
PTR—Pointer Record 187
MX—Mail Exchanger 188
Setting Up the `hosts.rev` File 189
Setting Up the `named.local` File 189
Modifying the Data Files 190
A Practical Example 191
Setting Up a Root Server for a Local Network 198

Index 201

Figures

Figure 3-1 Sample NIS+ domain 22

Figure 10-1 Name to Address Resolution 153

Figure 10-2 Name to Address Resolution for a Remote Host 154

Figure 10-3 Domains and Subdomains 157

Figure 10-4 Hierarchy of DNS Domains in a Single Organization 158

Figure 10-5 Hierarchy of Internet Domains 159

Figure 10-6 Wiz Domain's Position in the DNS Namespace 161

Figure 10-7 Domains and Zones. 163

Tables

Table P-1	Typographic Conventions	xxii
Table P-2	Shell Prompts	xxii
Table 1-1	Servers, Credentials, Directories, and Groups Worksheet	7
Table 1-2	NIS+ Tables Worksheet	8
Table 2-1	NIS+ Scripts	14
Table 3-1	Recommended NIS+ Setup Procedure Overview	19
Table 3-2	NIS+ Domains Setup Command Lines Summary	23
Table 3-3	Creating the Sample Namespace: Command Summary	62
Table 4-1	Setting Up a Root Domain: Command Summary	88
Table 5-1	Setting Up a Client: Command Summary	103
Table 6-1	Starting Up a Nonroot Master Server: Command Summary	112
Table 6-2	Adding a Replica: Command Summary	112
Table 7-1	Setting Up a Subdomain Command Summary	122
Table 8-1	Transferring Files Into NIS+ Tables: Command Summary	140
Table 8-2	Transferring Maps Into NIS+ Tables: Command Summary	141
Table 8-3	Transferring NIS+ Tables to NIS Maps: Command Summary	141

Table 8-4 Limiting Acces to Passwd Column: Command Summary . . . 141
Table 10-1 Internet Organizational Domains 159
Table 10-2 Domains and Mail Hosts 166
Table 12-1 Special Resource Record Characters. 177
Table 12-2 Commonly Used Resource Record Types 179
Table 12-3 Domain Configuration of Example Network—Class C. 191
Table 12-4 Domain Configuration of Example Network—`gull` Zone . . 191
Table 12-5 Domain Configuration of Example Network—`falcon` Zone 191
Table 12-6 Domain Configuration of Example Network—`owl` Zone. . . . 192

Code Samples

Code Example 4-1 `nsswitch.conf` file 72

Code Example 5-1 NIS+ Version of `nsswitch.conf` File 94

Code Example 11-1 Sample `resolv.conf` File 168

Code Example 12-1 Master Boot File for Primary Server. 172

Code Example 12-2 Sample Master Boot File for Secondary Server. 174

Code Example 12-3 Sample Master Boot File for Caching-only Server . . 175

Code Example 12-4 Sample named.ca File 180

Code Example 12-5 Sample `hosts` File 181

Code Example 12-6 SOA Record Format 182

Code Example 12-7 Sample SOA Resource Record. 184

Code Example 12-8 NS Record Format. 184

Code Example 12-9 Sample NS Resource Record 184

Code Example 12-10 Address Record Format 184

Code Example 12-11 Sample Address Record 185

Code Example 12-12 HINFO Record Format. 185

Code Example 12-13 Sample HINFO Resource Record 185

Code Example 12-14 WKS Record Format 186
Code Example 12-15 Sample WKS Resource Record 186
Code Example 12-16 CNAME Record Format. 186
Code Example 12-17 Sample CNAME Resource Record 187
Code Example 12-18 PTR Record Format. 187
Code Example 12-19 Sample PTR Resource Record 187
Code Example 12-20 MX Record Format 188
Code Example 12-21 Sample MX Resource Record 188
Code Example 12-22 Sample hosts.rev File 189
Code Example 12-23 Sample named.local File. 190
Code Example 12-24 Example Network Server Boot Files. 192
Code Example 12-25 Example `resolve.conf` Files 193
Code Example 12-26 Example named.local Files. 194
Code Example 12-27 Example `hosts` File for Server `gull`. 195
Code Example 12-28 Example `include` File for Server `gull` 196
Code Example 12-29 Example `hosts` and `include` File for Server `Falcon` 197
Code Example 12-30 Example Reverse Address for Server Gull 198

Preface

NIS+ and DNS Setup and Configuration Guide describes how to set up and configure NIS+ and DNS name services on a network. It includes network planning instructions and a tutorial on how to use the NIS+ start-up scripts to easily configure a basic NIS+ namespace. The DNS chapters show you how to configure DNS clients and servers. This manual is part of the Solaris™ 2.4 System and Network Administration manual set.

Who Should Use This Book

This book is for system and network administrators who want to set up a basic network using NIS+ or DNS. It assumes the reader is an experienced system administrator.

Although this book introduces networking concepts relevant to NIS+ and DNS, 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.

(*NIS+ and FNS Administration Guide* contains a thorough description of the NIS+ system, a glossary of NIS+ terms, and a listing of common NIS+ error messages.)

How This Book Is Organized

Chapter 1, "Getting Started With NIS+," describes the methods of NIS+ setup and the minimum requirements of an NIS+ namespace.

The remainder of this book is divided into three parts:

Part 1 — NIS+ Setup: Scripts

This part provides a tutorial on how to use the NIS+ setup scripts to establish and configure a NIS+ namespace. The scripts are the recommended method of setting up NIS+.

Chapter 2, "NIS+ Setup Scripts—Introduction," describes the NIS+ scripts and what they will and will not do.

Chapter 3, "Setting Up NIS+ With Scripts," takes you step-by-step through the configuring of an NIS+ namespace using the NIS+ scripts. At the end of this chapter are blank worksheets that you can use to determine your domain and server requirements.

Part 2—NIS+ Setup: Command Set

This part provides step-by-step instructions for setting up the components of an NIS+ namespace using the NIS+ command set. (If you are creating an entire NIS+ namespace from scratch, it is recommend that you use the set up scripts described in Part 1—NIS+ Setup: Scripts.)

Chapter 4, "Setting Up the Root Domain," provides step-by-step instructions for setting up the root domain, including using the NIS-compatibility mode.

Chapter 5, "Setting Up NIS+ Clients." provides step-by-step instructions for setting up an NIS+ client and includes three different initialization methods. These instructions apply to clients in both the root domain and subdomains, whether all-NIS+ or NIS-compatible.

Chapter 6, "Setting Up NIS+ Servers." provides step-by-step instructions for setting up any kind of NIS+ server except the root master.

Chapter 7, "Setting Up a Nonroot Domain," provides step-by-step instructions for creating and setting up a subdomain, including designating its master and replica servers.

Chapter 8, “Setting Up NIS+ Tables,” provides step-by-step instructions for populating NIS+ tables with information from input files or NIS maps.

Chapter 9, “Setting Up the Name Service Switch,” provides step-by-step instructions for setting up the name service switch to be used with NIS, NIS+, or DNS, as well as to provide backward compatibility with the +/- syntax.

Part 3 —DNS Setup

This part gives an overview of DNS (Domain Name System) and describes how to setup DNS clients and servers.

Chapter 10, “Introduction to DNS,”describes the structure of the Domain Name System.

Chapter 11, “Setting Up DNS Clients,” describes how to configure a DNS client.

Chapter 12, “Setting Up DNS Servers,” describes how to configure a DNS server.

Related Books

You can consult the following for more information on NIS+ and DNS. These books are also part of the Solaris 2.5 System and Network Administration manual set:

- *NIS+ and FNS 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 2.5 manual set:

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

What Typographic Changes and Symbols 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	#

Getting Started With NIS+ 1

This chapter discusses the information you need to assemble and the preparations you need to make before you start to set up and configure NIS+.

NIS+ Overview	*page 1*
Setting Up NIS+	*page 2*
Before You Start NIS+	*page 3*
Planning Your NIS+ Layout	*page 3*
Determining Server Requirements	*page 4*
Disk Space and Memory Recommendations	*page 4*
Preparing the Existing Namespace	*page 5*
Configuration Worksheets	*page 7*

NIS+ Overview

NIS+ (pronounced “en-eye-ess-plus” or “niss-plus”) 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 have access to it. This configuration of network information is referred to as the NIS+ *namespace*.

1

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

NIS+ uses a client-server model to store and have access to the information contained in an NIS+ namespace. Each domain is supported by a set of servers. The principal server is called the *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 propagated automatically 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 through description of NIS+, see *NIS+ and FNS Administration Guide*.

Setting Up NIS+

This manual describes two different methods of setting up an NIS+ namespace:

- *With the setup scripts*. Part 1 describes how to set up NIS+ using the three NIS+ scripts: `nisserver`, `nispopulate`, and `nisclient`. This is the easiest method, and recommended, method.

- *With the NIS+ command set*. Part 2 describes how to set up 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 set up a namespace with characteristics significantly different than those provided by the setup scripts.

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

Before You Start NIS+

Before you start to set up NIS+ at your site, you need to

1. Plan your NIS+ layout. See “Planning Your NIS+ Layout” on page 3 and use the planning worksheets on page 7 and page 8. See *NIS+ Transition Guide* for a complete description of the planning process.
2. Prepare your existing namespace (if any). See “Preparing the Existing Namespace” on page 5.
3. Choose a root domain name.
4. Choose a root server machine.
5. 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.)

To create the sample namespace described in the Part 1 tutorial, you need only do steps 2, 4, and 5 above. The tutorial does the NIS+ layout planning for you and chooses a domain name.

Planning Your NIS+ Layout

To plan the structure of your NIS+ namespace:

- Determine your server requirements (see page 4).
- Determine your disk space and memory requirements (see page 4).

- Sketch the domain hierarchy.
- Select servers to be used for the namespace.
- Determine the administrative groups and their members.
- Determine access rights to the namespace.

See *NIS+ Transition Guide* for a full description of these steps and use the "Configuration Worksheets" on page 7 to help plan your namespace.

You don't have to do any planning to work through the tutorial in Chapter 3, "Setting Up NIS+ With Scripts." You just need a few networked machines to practice on. But be sure to plan your site's hierarchy before you move from the tutorial to setting up your real NIS+ namespace.

Determining Server Requirements

Once you have determined the domain structure of your namespace, you can choose the servers that will support them. You need to differentiate between the requirements imposed by NIS+ and those imposed by the traffic load of your namespace.

NIS+ requires you to assign at least one server, the master, to each NIS+ domain. Although you can assign any number of replicas to a domain, more than 10 per domain is not recommended. An NIS+ server is capable of supporting more than one domain, but this is not recommended except in small namespaces or testing situations. The number of servers a domain requires is determined by the traffic load and the configuration of its servers.

Here are some guidelines for determining how many servers you will need:

- Assign one master server per domain in the hierarchy.
- Add at least one replica server for each domain. (A replica can answer requests when the master is unavailable.)
- Calculate the disk space requirements of each server. The next section, "Disk Space and Memory Recommendations," describes how to calculate disk space usage.

Disk Space and Memory Recommendations

Disk space requirements depend on four factors:

- Disk space consumed by the Solaris 2.5 software
- Disk space for `/var/nis` (and `/var/yp`)
- Amount of memory
- Swap space required for NIS+ processes

Depending on how much you install and whether or not you include the OpenWindows™ software, the Solaris 2.5 software can consume over 220 Mbytes of disk space. You should also count the disk space consumed by other software the server may use. NIS+ is part of the Solaris 2.4 distribution, so it does not consume additional disk space.

NIS+ data is stored in `/var/nis`. The directory `/var/nis` uses approximately 5 Kbytes of disk space per client of the domain. For example, if a domain has 1000 clients, `/var/nis` requires about 5 Mbytes of disk space. Because transaction logs, also kept in `/var/nis`, can grow large, you may want to add more space in addition to whatever is required for the domain's clients—an additional 10–15 Mbytes is recommended. In other words, for 1000 clients, allocate 15–20 Mbytes for `/var/nis`. You can reduce this amount if you checkpoint transaction logs regularly. Try to keep `/var/nis` on a separate partition; this separation will help during an operating system upgrade.

If you are going to load information into NIS+ from NIS maps, allocate an appropriate amount of space for `/var/yp` to hold those NIS maps.

Although 32 Mbytes is the minimum memory requirement for servers (root master, subdomain master servers, and replica servers), you should equip servers of medium-to-large domains with at least 64 Mbytes.

In addition to the server's normal swap space requirements, NIS+ requires swap space equal to two or three times the server's `rpc.nisd` process size because the server process forks during certain operations. See "Configuring a Client as an NIS+ Server" on page 45 and the `rpc.nisd` man page for more information.

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 if you like. (You can change it later to a hierarchical structure.) Read *NIS+ Transition Guide* before you start your transition from NIS to NIS+ for important planning and preparation information. The NIS+ scripts easily enable you to start NIS+ with data from

NIS maps. Chapter 3, “Setting Up NIS+ With Scripts,” shows you how to use the NIS+ scripts to create a NIS+ namespace from either system files or NIS maps.

However, in order for the scripts to run smoothly, 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`. (See the `hosts(4)`man page for detailed information on allowable host names.)
- *Root server must be running.* The machine that will be designated 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 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 setup 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 not rename either the directories or the files.

Configuration Worksheets

Use the worksheets on the following pages to record planning information prior to NIS+ setup. There are two worksheets for each domain:

- "Servers, Credentials, Directories, and Groups Worksheet" on page 7
- "NIS+ Tables Worksheet" on page 8

If you have more than one domain, make copies of the blank worksheets.

Table 1-1 Servers, Credentials, Directories, and Groups Worksheet

Domain:						
Servers	**Type**	**Name**				**Specifications**
	Master					
	First Replica					
	Second Replica					
Credentials	**Type of Principal**	**Type of Credential**				
	Servers					
	Clients					
	Administrators					
	Users					
Rights	**Types of Objects**	**Category & Rights**				
	Directories	**N**	**O**	**G**	**W**	**Use Defaults?**

Table 1-1 Servers, Credentials, Directories, and Groups Worksheet

Domain:

Servers	**Type**	**Name**				**Specifications**
	Groups	**N**	**O**	**G**	**W**	**Description**

Table 1-2 NIS+ Tables Worksheet

Domain:

Rights	**Types of Objects**	**Category & Rights**				
	Tables	**N**	**O**	**G**	**W**	**Notes**
	bootparams					
	hosts					
	passwd					
	cred					
	group					
	netgroup					
	mail_aliases					
	timezone					
	networks					
	netmasks					
	ethers					
	services					

Table 1-2 NIS+ Tables Worksheet

Domain:

protocols					
rpc					
auto_home					
auto_master					

1

Part 1 — NIS+ Setup: Scripts

Part 1 describes how to use the NIS+ setup and configuration scripts to establish and configure a namespace. It has two chapters.

NIS+ Setup Scripts—Introduction	*page 13*
Setting Up NIS+ With Scripts	*page 17*

NIS+ Setup Scripts—Introduction 2

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

About the NIS+ Scripts	*page 13*
What the NIS+ Scripts Won't Do	*page 14*

Caution – Before running the NIS+ setup scripts, make sure you have performed the steps described in "Before You Start NIS+" on page 3.

About the NIS+ Scripts

The three NIS+ scripts—`nisserver`, `nispopulate`, and `nisclient`—enable you to set up a NIS+ namespace easily. The NIS+ scripts are Bourne shell scripts that execute groups of NIS+ commands so you don't have to type the NIS+ commands individually. Table 2-1 on page 14 describes what each script does.

2

Table 2-1 NIS+ Scripts

NIS+ Script	What It Does
`nisserver`	Sets up the root master, nonroot 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 3, "Setting Up NIS+ With Scripts," 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. First running the scripts with `-x` may minimize unexpected surprises.

What the NIS+ Scripts Won't 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 wish to refer back to this section after you have created the sample NIS+ namespace.

The `nisserver` script will only set 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 3, “Setting Up NIS+ With Scripts,” 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 3, “Setting Up NIS+ With Scripts,” for how to use the `rpc.nisd` command instead of one of the NIS+ scripts to change NIS+ client machines into nonroot 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.

See Part 2 for information on how to perform any of the above tasks with the NIS+ command set.

2

Setting Up NIS+ With Scripts 3

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

Note – Using these scripts is the recommended method of setting up and configuring an NIS+ namespace. It is much simpler to use these scripts than to try setting up an NIS+ namespace with the NIS+ command set as described in Part 2.

This chapter provides the following information:

NIS + Setup Overview	*page 19*
Creating a Sample NIS+ Namespace	*page 20*
Setting Up NIS+ Root Servers	*page 24*
Populating NIS+ Tables	*page 30*
Setting Up Root Domain NIS+ Client Machines	*page 39*
Initializing NIS+ Client Users	*page 42*
Setting Up NIS+ Servers	*page 44*
Designating Root Replicas	*page 46*
Creating a Subdomain	*page 49*
Populating the New Domain's Tables	*page 52*
Designating Replicas	*page 56*

3

Initializing Subdomain NIS+ Client Machines	*page 58*
Initializing Subdomain NIS+ Client Users	*page 60*
Summary of Commands for the Sample NIS+ Namespace	*page 62*

This chapter also describes the following procedures:

▼ *How to Create a Root Master Server*	*page 25*
▼ *How to Change Incorrect Information*	*page 28*
▼ *How to Populate the Root Master Server Tables*	*page 32*
▼ *How to Initialize a New Client Machine*	*page 40*
▼ *How to Initialize an NIS+ User*	*page 43*
▼ *Configuring a Client as an NIS+ Server*	*page 45*
▼ *How to Create a Root Replica*	*page 47*
▼ *How to Create a New NonRoot Domain*	*page 50*
▼ *Populating the Master Server Tables*	*page 55*
▼ *How to Create a Replica*	*page 58*
▼ *How to Initialize a New Subdomain Client Machine*	*page 60*
▼ *How to Initialize an NIS+ Subdomain User*	*page 61*

See the `nisserver`, `nispopulate`, and `nisclient` man pages for complete descriptions of the scripts.

See the glossary in *NIS+ and FNS Administration Guide* for definitions of terms and acronyms you don't recognize.

You should *not* use the small sample NIS+ namespace described in this tutorial as a basis for your actual NIS+ namespace. You should destroy the sample namespace once you are done 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.

NIS + Setup Overview

Table 3-1 summarizes the recommended generic setup procedure. The left column lists the major setup activities, such as setting up 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 3-1 Recommended NIS+ Setup 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 5 and *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.)	
Set up root Domain	Create the root domain. Set up 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`

Set up root domain clients	Set up 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`
Set up root replicas	Designate one or more of the servers you just set up as replicas of the root domain.	`rpc.nisd` `nisserver`

Table 3-1 Recommended NIS+ Setup Procedure Overview *(Continued)*

Activity	Description	Script/NIS+ Commands
Set up nonroot domains	Create a new domain. Designate previously enabled server as its master. Create its admin group and admin credentials.	`rpc.nisd` `nisserver`
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`
Set up nonroot domain clients	Set up 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 *NIS+ Transition Guide* and *NIS+ and FNS Administration Guide* for more information on NIS-compatibility mode.

Note – Your site's actual NIS+ namespace and its domain hierarchy will probably differ from the sample namespace's, and yours will probably contain 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 merely an illustration of how to use the NIS+ scripts. Once 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 will contain the following components:

- A root master server named `master` for the wiz.com. domain
- Four clients of the root domain, wiz.com:
  - The first client, `wizclient1`, will become a root replica (for the wiz.com. domain).

  - The second client, `wizclient2`, will become a master server for a new subdomain (for the subwiz.wiz.com. domain).
  - The third client, `wizdlient3`, will become a nonroot replica server of the new subdomain (for the subwiz.wiz.com. domain).
  - The fourth client, `wizclient4`, will remain solely a client of the root domain (wiz.com.).
- Two clients, `subclient1` and `subclient2`, of the subdomain (subwiz.wiz.com.)

This scenario shows the scripts being used to set up 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.

Figure 3-1 shows the layout of the sample namespace. When you finish creating the sample domain, it should resemble the NIS+ domain in this figure. Notice that some machines are simultaneously servers and clients.

3

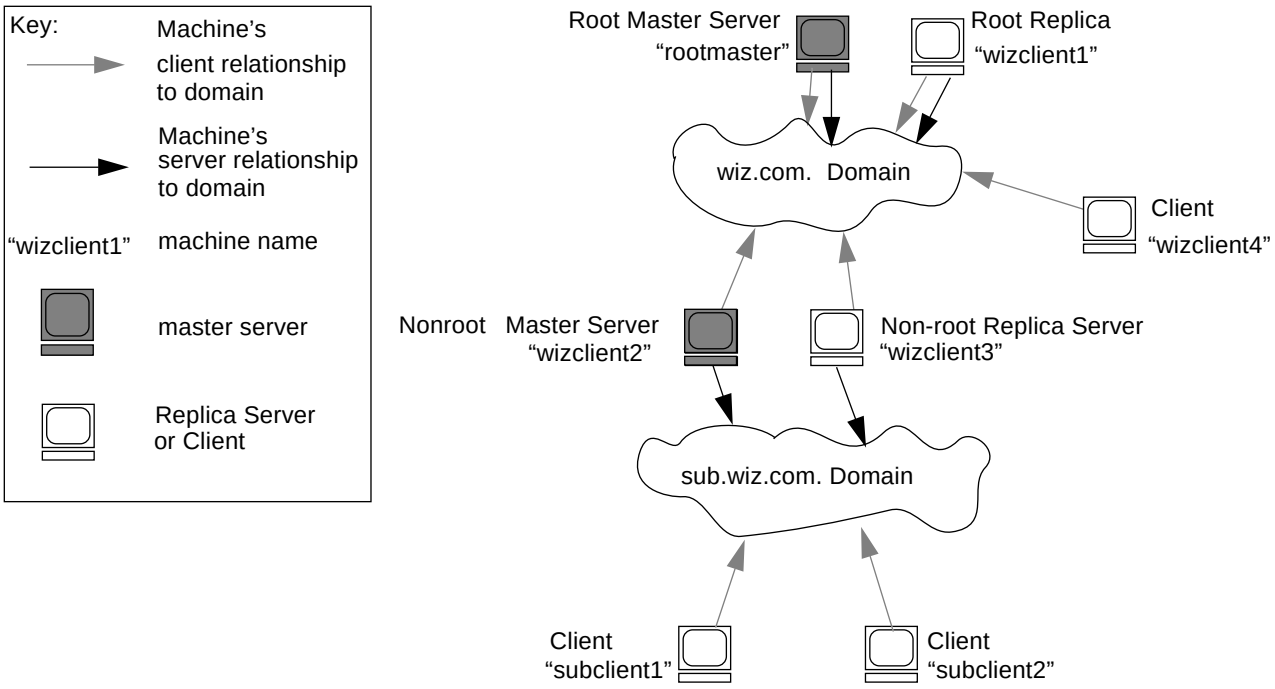


Figure 3-1 Sample NIS+ domain

Summary of NIS+ Scripts Command Lines

Table 3-2 on page 23 contains the generic sequence of NIS+ scripts and commands you will use to create the NIS+ domains shown in Figure 3-1. 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 3-2 as a quick-reference guide to the appropriate command lines. Table 3-2 is a summary of the actual commands with the appropriate variables that you will type to create the sample NIS+ namespace.

Table 3-2 NIS+ Domains Setup 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`

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*`.`

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` *NIS_servername*`\`
		`-a` *NIS_server_ipaddress* **`-y`** *NIS_domain*

Add additional users to the NIS+ admin group.	Root master server as superuser	`nisgrpadm -a admin.`*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`** *rootmaster*
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 nonroot 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 \`
		NIS_servername `-a` *NIS_server_ipaddress* `-y` *NIS_domain*

Table 3-2 NIS+ Domains Setup Command Lines Summary *(Continued)*

Action	Machine	Command
Convert a client to a master server replica.	Subdomain master server as superuser	`nisserver -R -d` *subdomain.domain.* `-h` *clientname*
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 will call without actually having the commands execute, use the `-x` option. The `-x` option will cause 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` may 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 set up a root master server using the `nisserver` script with default settings. The root master server will use 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 may be changed later. See *NIS+ and FNS Administration Guide* and Chapter 9, “Setting Up the Name Service Switch,” 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

In the following example, the machine that will be designated the root master server is called `rootmaster`, and wiz.com. will be the new root domain.

Note – Domains and hosts should not have the same name. For example, if you have `wiz.com` as a root domain you should not have a machine named `wiz` 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 applies to subdomains; for example, if you have a machine named `west`, you don't 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. **Type the following command as superuser (root) to set up a root master server.**
   The `-r` option indicates that a root master server should be set up. The `-d` option specifies the NIS+ domain name.

```
rootmaster# nisserver -r -d wiz.com.
This script sets up this machine “rootmaster” as a NIS+
root master server for domain wiz.com.

Domain name                    : wiz.com.
NIS+ group                     : admin.wiz.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 wiz.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 28 for instructions on how to change this name.

“NIS compatibility” refers to whether an NIS+ server will accept information requests from NIS clients. When set to OFF, the default setting, the NIS+ server will not fulfill requests from NIS clients. When set to ON, an NIS+ server will fulfill such requests. You can change the NIS-compatibility setting with this script. See “How to Change Incorrect Information” on page 28.

Note – This script only sets machines up 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 *NIS+ and FNS Administration Guide* and the `rpc.nisd(1M)` man page for more information on changing security levels.

---

3. **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 28” 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 wiz.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)
```

4. **Type** `y` **to continue the NIS+ setup.**
   (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 set up 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

setting up domain information "wiz.com." ...

setting up switch information ...

running nisinit ...
This machine is in the wiz.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.wiz.com. created
...
...
Enter login password:
```

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

5. **Type your machine's root password at the prompt and press Return.**
   In this case, the user typed the `rootmaster` machine's root password.

```
Wrote secret key into /etc/.rootkey

setting NIS+ group to admin.wiz.com. ...

restarting root server at security level 2 ...

This system is now configured as a root server for domain wiz.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 set up and ready for you to populate the NIS+ standard tables. To continue with populating tables, skip to "Populating NIS+ Tables" on page 30.

▼ How to Change Incorrect Information

If you typed `n` because some or all of the information returned to you was wrong in Step 3 in the above procedure, you will see the following:

```
Is this information correct? (type 'y' to accept, 'n' to change) n
Domain name: [wiz.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 wiz.com. The script then prompts for the NIS+ group name.

```
Is this information correct? (type 'y' to accept, 'n' to change) n
Domain name: [wiz.com.]
NIS+ group: [admin.wiz.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.wiz.com.] netadmin.wiz.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 45 for more information.

```
NIS (YP) compatibility (0=off, 1=on): [0]

Domain name             : wiz.com.
NIS+ group              : netadmin.wiz.com.
NIS (YP) compatibility  : OFF
Security level          : 2=DES

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

Once the information is correct, continue with Step 3 in “How to Create a Root Master Server” on page 25. You can keep choosing `n` until the information is correct.

Note – This script only sets machines up at security level 2. 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 *NIS+ and FNS Administration Guide* and the `rpc.nisd(1M)`man page for more information on changing security levels.

3

Populating NIS+ Tables

Once the root master server has been set up, you should 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 (wiz.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 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 setup 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. *NIS+ and FNS Administration Guide* and Chapter 8, "Setting Up NIS+ Tables," 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` you don't want to create a sales.west.myco.com subdomain.

- Remove any 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 running the `nispopulate` script, you must eliminate any dots in your host names. You can convert host name dots to hyphens or underscores. 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 will first need to collect the information and 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 `/nis+files`, for instance.)
- Edit four of the copied NIS files, `passwd`, `shadow`, `aliases`, and `hosts`, for security reasons. For example, you may want to remove the following lines from the copy of your local `passwd` file so they will not be 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 set up 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 would have to be the user root on the machine `rootmaster`.

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 Step a or Step b to populate the root master server tables and then continue with Step 2.**
   Step a shows you how to populate tables from files. Step b shows you how to populate tables from NIS maps. Type these commands in a scrolling window; otherwise, the script's output may scroll off the screen.

Note – The `nispopulate` script may 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 will use the `/tmp` directory.

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

```
rootmaster# nispopulate -F -p /nis+files -d wiz.com.

NIS+ domain name        : wiz.com.
Directory Path          : /nis+files


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

The `-F` option indicates that the tables will 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 `wiz.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.

```
rootmaster# nispopulate -Y -d wiz.com. -h corporatemachine
-a 130.48.58.111 -y corporate.wiz.com.

NIS+ domain name              : wiz.com.
NIS (YP) domain               : corporate.wiz.com
NIS (YP) server hostname      : corporatemachine

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

The `-Y` option indicates that the tables will 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 `corporatemachine`. You would 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 would 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

corporate.wiz.com; you would have to insert the NIS domain name of the real NIS domain at your site to create the sample domain. Remember that NIS domain names are case sensitive.)

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 28 for what you need to do if the information is incorrect.)

   a. If you performed Step 1a, 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
wiz.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)
```

   b. If you performed Step 1b, 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
wiz.com. from the NIS (YP) maps in domain corporate:
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)
```

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 may leave the tables only partially populated. The script does not do any automatic recovery or cleaning up. You can safely rerun the script, however, the tables will be overwritten with the latest information.

   a. If you are populating tables from files, 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 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 this Secure RPC password. 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.

b. 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 corporate.wiz.com NIS(YP)
domain...
auto_master table done.

populating auto_home table from file corporate.wiz.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 this Secure RPC password. Use this password when prompted for your network or Secure RPC password.

All the tables are now populated. You can ignore the `parse error` warnings shown above. The 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 your self 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 would enter:

```
rootmaster# nisgrpadm -a admin.wiz.com. topadm.wiz.com. secondadm.wiz.com.
Added "topadm.wiz.com." to group "admin.wiz.com.".
Added "secondadm.wiz.com." to group "admin.wiz.com.".
```

The `admin.wiz.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 set up NIS+.

You don't 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 43 for more information on initializing users. The group cache also has to expire before the new members will become active.

5. Type the following command to checkpoint the domain.

```
rootmaster# nisping -C wiz.com.
Checkpointing replicas serving directory wiz.com.
Master server is rootmaster.wiz.com.
        Last update occurred at date
Master server is rootmaster.wiz.com.
checkpoint scheduled on rootmaster.wiz.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 set up the root domain, this step affects only the root master server, as the root domain does not yet have replicas.

Caution – If you don't have enough swap or disk space, the server will be unable to checkpoint properly, but it won't notify you. One way to make sure all goes well is to list the contents of a table with the `niscat` command. For example, to check the contents of the `rpc` table, type

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

If you don't have enough swap space, you'll 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 doesn't *seem* to, this message indicates that you don't have enough swap space. Increase the swap space and checkpoint the domain again.

Setting Up Root Domain NIS+ Client Machines

Once the root master server's tables have been populated from files or NIS maps, you can initialize an NIS+ client machine. Since 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 NIS+ client machine is a different workstation than the NIS+ root server. The script will use:

- The domain used in previous examples, wiz.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 40 does not set up an NIS+ client to resolve host names requiring DNS. You need to explicitly include DNS for clients in their name service switch files. See *NIS+ and FNS Administration Guide* and Chapter 9, "Setting Up the Name Service Switch," 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 set up 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 be logged in as superuser on the machine that is to become an NIS+ client. In this example, the new client machine is named `wizclient1`.

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)

▼ How to Initialize a New Client Machine

1. **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 would be the current domain name.) The `-h` option specifies the NIS+ server's host name.

```
wizclient1#nisclient -i -d wiz.com. -h rootmaster

Initializing client wizclient1 for domain "wiz.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)
```

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

```
Do you want to continue? (type 'y' to continue, 'n' to exit this
script) y

Type server rootmaster's IP address:
```

3. **Type the correct IP address, and press Return.**
   This example uses the address 123.123.123.123.

```
Type server rootmaster's IP address: 123.123.123.123

setting up the domain information...

setting up the name service switch information...
```

4. **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.

```
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 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.
```

6. **Reboot your new client machine.**
   Your changes will 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 to the appropriate ones.

The sample NIS+ domain described in this chapter assumes that you will initialize four clients in the domain wiz.com. 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 wiz.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

Once 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 set up 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`.

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 that will become the NIS+ client

▼ How to Initialize an NIS+ User

1. **To become a NIS+ client, type the following 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** `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.

3

Setting Up NIS+ Servers

Now that the client machines have been initialized, you can change any of them to NIS+ servers but not into root NIS+ servers. Root NIS+ servers are a special type of NIS+ server. See “Setting Up NIS+ Root Servers” on page 24 for more information. You need NIS+ servers for three purposes:

- 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

You can configure servers any of three 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 `wizclient1` 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 set up 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 `wizclient1`.

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 also should be NIS compatible.

▼ How to Configure a Server Without NIS Compatibility

```
wizclient1# 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.**

```
wizclient1# 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.**

```
wizclient1# 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.

Designating Root Replicas

To have regularly available NIS+ service, you should always create root replicas. Having replicas may also speed network-request resolution because multiple servers are available to handle requests. The root replica server contains exact copies of the NIS+ tables on the root server. Replication of the master's database starts a few minutes after you perform this procedure and can take anywhere from a few minutes to a couple of hours to complete, depending on the size of your tables.

This example shows the machine `wizclient1` being configured as a root replica. This procedure uses the NIS+ `nisserver` script.

Prerequisites to Running `nisserver`

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

- The domain must have already been set up 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 have started `rpc.nisd` on the client.
- You must be logged in as root on the root master server. In this example, the root master machine is named `rootmaster`.

Information You Need

You need:

- The domain name
- The client machine name; (`wizclient1`, 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.**
   The `-R` option indicates that a replica should be set up. The `-d` option specifies the NIS+ domain name (wiz.com., in this example). The `-h` option specifies the client machine (`wizclient1`, in this example) that will become the root replica.

```
rootmaster# nisserver -R -d wiz.com. -h wizclient1
This script sets up a NIS+ replica server for domain wiz.com.
Domain name: :wiz.com.
NIS+ server: :wizclient1
Is this information correct? (type 'y' to accept, 'n' to change)
```

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 28 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 "wizclient1" as an NIS+
replica server for domain wiz.com. without NIS compatibility.
The NIS+ server daemon, rpc.nisd, must be running on wizclient1
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 wizclient1 is now configured as a replica server for
domain wiz.com..
The NIS+ server daemon, rpc.nisd, must be running on wizclient1
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 only need to modify 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 45 for more information on creating NIS-compatible servers.

The machine `wizclient1` is now an NIS+ root replica. The new root replica can handle requests from the clients of the root domain. Since there are now two servers available to the domain, information requests may be fulfilled faster.

Creating Additional Replicas

Repeat the preceding server-to-replica conversion procedure on as many server machines as you like. For performance reasons, you should have no more than a few replicas per domain. Do create as many replicas, though, as is necessary to serve physically distant sites. For example, it may make sense from an organizational point of view to have two physically distant sites in the same NIS+ domain. If a root replica and the master of the domain are at the first site, there will be much network traffic between the first site and the second site of the domain. Creating an additional root replica at the second site should reduce network traffic. See *NIS+ Transition Guide* for more information on replica distribution.

The sample NIS+ domain described in this chapter includes only one root replica. One of the other clients of the wiz.com. domain will be converted to a replica of the subdomain created in the next section.

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 wiz.com. domain. The hierarchical structure of NIS+ allows you to create a domain structure that parallels your organizational structure.

This example shows the machine `wizclient2` being converted to the master server of a new domain called subwiz.wiz.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 nonroot domain:

- The parent domain must have already been set up 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 `rootmaster`.

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 (`wizclient2,` in this example)
- The superuser password for the parent master server

In the following example, the new nonroot domain is called subwiz.wiz.com.

Note – Any NIS+ client can be converted to an NIS+ master server as long as it is itself in a domain above the domain it will be serving. For example, an NIS+ client in domain subwiz.wiz.com can serve domains below it in the hierarchy, such as corp.subwiz.wiz.com or even east.corp.subwiz.wiz.com. This client cannot, however, serve the domain wiz.com, because wiz.com is above the domain subwiz.wiz.com in the hierarchy. Root replicas are the only exception to this rule. They are clients of the domain that they serve.

▼ How to Create a New NonRoot Domain

1. **Type the following command as superuser (root) on the NIS+ domain's root master server to create a new nonroot domain master server.**
   The `-M` option indicates that a master server for a new nonroot domain should be created. The `-d` option specifies the *new* domain name, subwiz.wiz.com. in this instance. The `-h` option specifies the client machine, (`wizclient2,` in this example), that will become the master server of the new domain.

```
rootmaster# nisserver -M -d subwiz.wiz.com. -h wizclient2

This script sets up a non-root NIS+ master server for domain
subwiz.wiz.com.
Domain name             : subwiz.wiz.com.
NIS+ server             : wizclient2
NIS+ group              : admin.subwiz.wiz.com.
NIS (YP) compatibility  : OFF
Security level          : 2=DES
Is this information correct? (type 'y' to accept, 'n' to change)
```

Master servers of new nonroot domains are created with the same set of default values as root servers. See "How to Create a Root Master Server" on page 25 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 28" 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 "wizclient2" as an NIS+
non-root master server for domain subwiz.wiz.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 will exit 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.subwiz.wiz.com. created
groups_dir.subwiz.wiz.com. created
...
...
setting NIS+ group admin.subwiz.wiz.com. ...

The system wizclient2 is now configured as a non-root server for domain
subwiz.wiz.com. You can now populate the standard NIS+ tables by using the
nispopulate or /usr/lib/nis/nisaddent commands.
```

The machine `wizclient2` is now the master server of the subwiz.wiz.com. domain. The subwiz.wiz.com. domain is a subdomain of the wiz.com. domain. The machine `wizclient2` is simultaneously still a client of the root domain wiz.com., and the master server of the subwiz.wiz.com. domain. See Figure 3-1 on page 22.

You can now populate the standard NIS+ tables on the new master server of the subwiz.wiz.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 1, "Getting Started With NIS+," and "Configuration Worksheets" on page 7, for more information on planning an NIS+ hierarchy.

Populating the New Domain'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, subwiz.wiz.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 set up 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 files, `passwd`, `shadow`, `aliases`, and `hosts`, for security reasons. For example, you may want to remove the following lines from the copy of your local `passwd` file so they will not be 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 set up 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 `wizclient2`.

Note – The `nispopulate` script may 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 will use 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 32, this example only shows you what you would type to populate the tables of the new domain, subwiz.wiz.com. For more information about this procedure, see "How to Populate the Root Master Server Tables."

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

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

- 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 may otherwise scroll off the screen.

▼ How to Populate the Tables From Files

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

```
wizclient2# nispopulate -F -p /nis+files -d subwiz.wiz.com.

NIS+ domain name        : subwiz.wiz.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.

```
wizclient2# nispopulate -Y -d subwiz.wiz.com. -h businessmachine
-a IP_addr_of_NIS_server -y business.wiz.com

NIS+ Domain name              : subwiz.wiz.com.
NIS (YP) domain               : business.wiz.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 32 for additional information.

Designating Replicas

Just as you did in the wiz.com. domain, to have regularly available NIS+ service, you should always create replicas. Having replicas may also speed network-request resolution since multiple servers are available to handle requests. The replica server contains exact copies of the NIS+ tables on the master server of your new domain. Replication of the master's database starts a few minutes after you perform this procedure and can take anywhere from a few minutes to a couple of hours to complete, depending on the size of your tables.

You use the same procedure to create a replica as you do to create a root replica. The major difference between creating the root replica and this replica is that the machine you are going to convert to a replica will remain a client of the domain above the one it will be serving as a replica. This example shows you only what you would 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 47."

Prerequisites to Running `nisserver`

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

- The domain must have already been set up 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 `wizclient2`.

Information You Need

- The domain name
- The client machine name (`wizclient3,` 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.**

```
wizclient2# nisserver -R -d subwiz.wiz.com. -h wizclient3
This script sets up a NIS+ replica server for domain
subwiz.wiz.com.

Domain name ::subwiz.wiz.com.
NIS+ server  :wizclient3
Is this information correct? (type 'y' to accept, 'n' to change)
```

In this example, `wizclient2` is the master server. The `-R` option indicates that a replica should be set up. The `-d` option specifies the NIS+ domain name (subwiz.wiz.com. in this example). The `-h` option specifies the client machine (`wizclient3`, in this example) that will become the replica. Notice that this machine is still a client of the wiz.com. domain and not a client of the subwiz.wiz.com. domain.

See "How to Create a Root Replica" on page 47 for the rest of this script's output.

Initializing Subdomain NIS+ Client Machines

Once 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 than the NIS+ master server.

Note – The `-i` option used in "How to Initialize a New Subdomain Client Machine" on page 60 does not set up 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 an NIS+ Client to Use DNS" on page 145 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 40.

Prerequisites to Running `nisclient`

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

- The domain must have already been set up 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`.

Information You Need

You need:

- The domain name (subwiz.wiz.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 `wizclient2`)

▼ 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 subwiz.wiz.com. -h wizclient2

Initializing client subclient1 for domain "subwiz.wiz.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 would be the current domain name.) The `-h` option specifies the NIS+ server's host name.

See "How to Initialize a New Client Machine" on page 40 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 43.

Prerequisites to Running `nisclient`

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

- The domain must have already been set up 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 43 for the rest of this script's output.

3

Summary of Commands for the Sample NIS+ Namespace

Table 3-3 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. See Figure 3-1 on page 22 for a diagram of the sample namespace.

Table 3-3 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`**
Create root master server for wiz.com. domain.	`rootmaster#` **`nisserver -r -d wiz.com.`**
Populate the root master server's NIS+ tables—from files or from NIS maps.	`rootmaster#` **`nispopulate -F -p /nis+files -d wiz.com.`** or `rootmaster#` **`nispopulate -Y -d wiz.com. -h corporatemachine -a \ 130.48.58.111 -y corporate.wiz.com`**
Add additional members to the admin group (2).	`rootmaster#` **`nisgrpadm -a admin.wiz.com. topadmin.wiz.com. \ secondadmin.wiz.com.`**
Make a checkpoint of the NIS+ database.	`rootmaster#` **`nisping -C org_dir.wiz.com.`**
Initialize a NIS+ client machine in the wiz.com. domain.	`wizclient1#` **`nisclient -i -d wiz.com. -h rootmaster`**
Initialize user as a NIS+ client.	`wizclient1user1prompt%` **`nisclient -u`**
Convert NIS+ client to NIS+ server, without or with NIS compatibility or with NIS and DNS.	`wizclient1#`**`rpc.nisd`** or `wizclient1#`**`rpc.nisd -Y`** or `wizclient1#`**`rpc.nisd -Y -B`**
Create a root replica.	`rootmaster#` **`nisserver -R -d wiz.com. -h wizclient1`**
Convert a server to a nonroot master server of the subwiz.wiz.com. domain.	`rootmaster#` **`nisserver -M -d subwiz.wiz.com. -h wizclient2`**

Table 3-3 Creating the Sample Namespace: Command Summary *(Continued)*

Tasks	Commands

Populate the new master server's NIS+ tables—from files or from NIS maps.	`wizclient2# nispopulate -F -p /nis+files -d subwiz.wiz.com.`
	or
	`wizclient2# nispopulate -Y -d subwiz.wiz.com. -h \`
	`businessmachine -a 130.48.58.242 -y business.wiz.com`

Create a master server replica.	`wizclient2# nisserver -R -d subwiz.wiz.com. -h wizclient3`
Initialize a NIS+ client in the subwiz.wiz.com. domain.	`subclient1# nisclient -i -d subwiz.wiz.com. -h wizclient2`
Initialize user as a NIS+ client.	`subclient1user2prompt% nisclient -u`

3

Part 2 — NIS+ Setup: Command Set

Part 2 describes how to set up an NIS+ namespace using the NIS+ command set.

Note – The recommended method of setting up an NIS+ namespace is to use the set up scripts as described in Part 1. Setting up an NIS+ namespace with the NIS+ command set as described in this Part is much more difficult than using the scripts.

This part has six chapters.

Setting Up the Root Domain	*page 67*
Setting Up NIS+ Clients	*page 89*
Setting Up NIS+ Servers	*page 105*
Setting Up a Nonroot Domain	*page 113*
Setting Up NIS+ Tables	*page 123*
Setting Up the Name Service Switch	*page 143*

Setting Up the Root Domain 4

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

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.

See “Configuration Worksheets” on page 7, for worksheets that you can use to plan your NIS+ namespace.

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

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 each other. For instance, you must specify some security parameters before you create the root directory; the rest, after. To make the root domain easier to set up, this chapter separates these tasks into individual steps and arranges them into their most efficient order.

4

Standard versus NIS-Compatible Setup 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 10. The equivalent step for standard NIS+ domains is Step 11.

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 13. 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 page 87.

Summary of Steps

Here is a summary of the entire setup 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. Clean out leftover NIS+ material and processes.
5. Name the root domain's admin group.
6. Create the root directory and initialize the root master server.
7. [NIS-compatibility Only] Start the NIS+ daemon with `-Y`.
   [Standard NIS+ Only] Start the NIS+ daemon.
8. Verify that the daemon is running.
9. Create the root domain's subdirectories and tables.

10. Create DES credentials for the root master server.
11. Create the root domain's admin group.
12. Add the root master to the root domain's admin group.
13. Update the root domain's public keys.
14. Start the NIS+ cache manager.
15. Restart the NIS+ daemon with security level 2.
16. Add your LOCAL credentials to the root domain.
17. Add your DES credentials to the root domain.
18. Add credentials for other administrators.
19. Add yourself and other administrators to the root domain's admin group.

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 setup 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 may wish to review the security-related chapters of *NIS+ and FNS Administration Guide* before starting to set up your NIS+ environment.

Prerequisites

Before proceeding, make sure that

- The `/etc/passwd` file on the root master server must contain an entry for you and every other administrator whose credentials will be added to the root domain in this setup process.
- If the server will operate in NIS-compatibility mode and support DNS forwarding for Solaris 1.x clients, it must have a properly configured `/etc/resolv.conf` file as described in "Creating the resolv.conf File" on page 167.

- The server must have a unique machine name that does duplicate any user ID.
- 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 (for Step 1)
- The name of the root domain (for Step 2)
- The name of the root domain's admin group (for Step 8)
- Your UID and password
- The UID of any administrator whose credentials you will add to the root domain

▼ How to Set Up a Root Domain

1. **Log in as superuser on the machine to be the root master server.**
   The examples in these steps use `rootmaster` as the root master server and wiz.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.

Do not include a trailing dot with the domain name in this instance. The domainname command is not an NIS+ command, so it does not follow the NIS+ conventions of a trailing dot.

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

The above example changes the domain name of the root master server from strange.domain to wiz.com. When changing or establishing a domain name, make sure that the it has at least two labels; for example, wiz.com instead of wiz.

(More complete instructions are provided in "Specifying a Domain Name After Installation" on page 97.)

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 will be NIS+ tables.

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

This displays the current `nsswitch.conf` file as shown in Code Example 4-1 on page 72.

Code Example 4-1 `nsswitch.conf` file

```
#
# /etc/nsswitch.nisplus:
# “hosts:” and “services:” in this file are used only if the /etc/netconfig
# file contains “switch.so” as a nametoaddr library for “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+
#hosts:   nisplus dns [NOTFOUND=return] files

services:   nisplus [NOTFOUND=return] files
networks:   nisplus [NOTFOUND=return] files
protocols:  nisplus [NOTFOUND=return] files
rpc:        nisplus [NOTFOUND=return] files
ethers:     nisplus [NOTFOUND=return] files
netmasks:   nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files

publickey: nisplus

netgroup:  nisplus

automount: files nisplus
aliases:  files nisplus
```

If the root master server’s configuration file is different from the one in Code Example 4-1, change it to the NIS+ version.

4. **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 9, “Setting Up the Name Service Switch."

5. **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
```

6. **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 setup process. If you have stored any admin scripts in `/var/nis`, you may want to consider temporarily storing them elsewhere, until you finish setting up the root domain.

7. **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.

8. **Name the root domain's admin group.**
   Although you won't actually create the admin group until Step 15, 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 13.

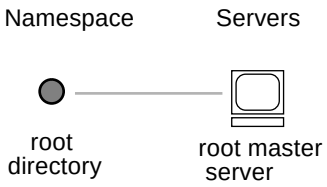
   To name the admin group, set the value of the environment variable `NIS_GROUP` to the name of the root domain's admin group. Here are two examples, one for `csh` users, and one for `sh/ksh` users. They both set `NIS_GROUP` to admin.wiz.com.

   For C Shell

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

   For Bourne or Korn Shell

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



9. **7Create 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 wiz.com. NIS+ domain
Setting up root server ...
All done.
```

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

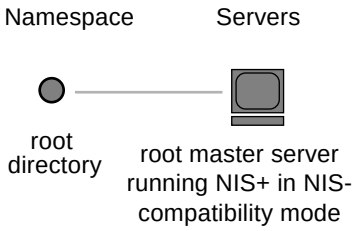
Within the `/var/nis` directory is a file named `root.object`.

```
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 will be created in Step 10 or Step 11.

In subsequent steps, other files will be 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+ setup procedures. 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*. In 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 not rename either the directories or the files.



10. [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 11 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, go to Step 13.

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.

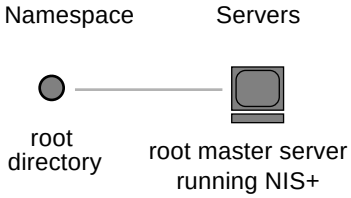
`rpc` *file with* DNS *forwarding*

```
EMULYP="-Y -B"
```

`rpc` *file without* DNS *forwarding*

```
EMULYP="-Y"
```

If you don't need to retain DNS forwarding capabilities, uncomment the line but don't add the `-B` flag.



11. [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 a higher security level, you would be locked out of the server.

12. Verify that the root objects have been properly created.

As a result of Step 10 or Step 11, 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 9. 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 wiz.com.
wiz.com.:
```

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

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

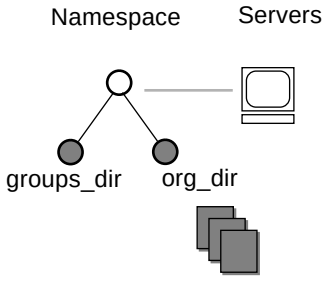
Note that the root directory object provides full (read, modify, create, and destroy) rights to both the owner and the group, while providing only read access to the world and nobody classes. (If your directory object does not provide these rights, you can change them using the `nischmod` command.)

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.



13. 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:

NIS-compatible only

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

Standard NIS+ only

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

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

```
rootmaster# /usr/lib/nis/nissetup
org_dir.wiz.com. created
groups_dir.wiz.com. created
auto_master.org_dir.wiz.com. created
auto_home.org_dir.wiz.com. created
bootparams.org_dir.wiz.com. created
cred.org_dir.wiz.com. created
ethers.org_dir.wiz.com. created
group.org_dir.wiz.com. created
hosts.org_dir.wiz.com. created
mail_aliases.org_dir.wiz.com. created
sendmailvars.org_dir.wiz.com. created
netmasks.org_dir.wiz.com. created
netgroup.org_dir.wiz.com. created
networks.org_dir.wiz.com. created
passwd.org_dir.wiz.com. created
protocols.org_dir.wiz.com. created
rpc.org_dir.wiz.com. created
services.org_dir.wiz.com. created
timezone.org_dir.wiz.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 11), it was empty. Now, however, it has two subdirectories.

```
rootmaster# nisls wiz.com.
wiz.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.

14. 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@wiz.com
Adding key pair for unix.rootmaster@wiz.com
          (rootmaster.wiz.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 will get a warning message and a prompt to repeat the password:

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

You can persist and retype the same password, and NIS+ will still create the credential. The new password will be stored in `/etc/.rootkey` and 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 *NIS+ and FNS Administration Guide.*

If you decide to use your login password after all, press Control-c and start the sequence over. If you were to simply 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.wiz.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 wiz.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.

15. **Create the root domain's admin group.**
    This step creates the admin group named in Step 8. Use the `nisgrpadm` command with the `-c` option. The example below creates the admin.wiz.com. group.

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

    This step only creates the group—it does not identify its members. That is done in Step 16. To observe the object properties of the group, use `niscat -o`, but be sure to append `groups_dir` in the group's name.

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

16. **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.wiz.com.` to the admin.wiz.com. group.

```
rootmaster# nisgrpadm -a admin.wiz.com. rootmaster.wiz.com.
Added rootmaster.wiz.com. to group admin.wiz.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 *NIS+ and FNS 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.wiz.com.
Group entry for admin.wiz.com. group:
  Explicit members:
    rootmaster.wiz.com.
  No implicit members
  No recursive members
  No explicit nonmembers
  No implicit nonmembers
  No recursive nonmembers
```

17. **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 *NIS+ and FNS 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 wiz.com.
rootmaster# /usr/lib/nis/nisupdkeys org_dir.wiz.com.
rootmaster# /usr/lib/nis/nisupdkeys groups_dir.wiz.com.
```

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

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

Now, if you look in any of those directories (use `niscat -o`), you will see this entry in the public key field:

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

18. 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 10 or Step 11), and downloads it into a file named `NIS_SHARED_DIRCACHE` in `/var/nis`.

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

```
rootmaster# nis_cachemgr
```

Once 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 *NIS+ and FNS Administration Guide.*

19. **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 one with security level 2.

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:

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.

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

20. **Add your LOCAL credentials to the root domain.**
Since 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.wiz.com.`

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

For more information about the `nisaddcred` command, see the credentials chapter of *NIS+ and FNS Administration Guide.*

21. **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@wiz.com -P topadmin.wiz.com. des
Adding key pair for unix.11177@wiz.com (topadmin.wiz.com.).
Enter login password:
```

If after entering your login password you get a `password differs from login password` warning and 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.

22. **Add credentials for 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 three different 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.wiz.com.

```
rootmaster# nisaddcred -p 33355 -P miyoko.wiz.com. local
rootmaster# nisaddcred -p unix.33355@wiz.com -P miyoko.wiz.com. des
Adding key pair for unix.33355@wiz.com (miyoko.wiz.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.wiz.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.wiz.com. local
rootmaster# nisaddcred -p unix.33355@wiz.com -P miyoko.wiz.com. des
Adding key pair for unix.33355@wiz.com (miyoko.wiz.com.).
Enter miyoko's login password:
nisaddcred: WARNING: password differs from login passwd.
Retype password:
rootmaster# nischown miyoko.wiz.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 *NIS+ and FNS Administration Guide* describes how to do this.

23. **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.wiz.com. group:

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

Root Domain Setup Summary

Table 4-1 on page 88 summarizes the steps requires to set up 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.

4

Table 4-1 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.wiz.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`.
or
[NIS+ Only] Start daemon with `-S 0`.
	
	`# rpc.nisd -Y -B -S 0`
	`# vi /etc/inet.d/rpc`
	or
	`# rpc.nisd -S 0`

Create `org_dir`, `groups_dir`, tables.
Create DES credentials for root master.
	`# /usr/lib/nis/nissetup [-Y]`
	`# nisaddcred des`
	`Enter login password:`

Create admin group. Assign full group rights to root directory	`# nisgrpadm -c admin.wiz.com.` `# nischmod g+rmcd wiz.com.`

Add root master to admin group.
Update root directory's keys.
Update `org_dir`'s keys.
Update `groups_dir`'s keys.
	`# nisgrpadm -a admin.wiz.com. rootmaster.wiz.com.`
	`# /usr/lib/nis/nisupdkeys wiz.com.`
	`# /usr/lib/nis/nisupdkeys org_dir.wiz.com.`
	`# /usr/lib/nis/nisupdkeys groups_dir.wiz.com.`

Start NIS+ cache manager
Kill existing daemon.
	`# nis_cachemgr`
	`# 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.
Add your DES credentials.
	`# nisaddcred -p 11177 -P topadmin.wiz.com. local`
	`# nisaddcred -p unix.11177@wiz.com \`
	`-P topadmin.wiz.com. des`
	`Enter login password:`

Add credentials for other admins.Add other admins to admin group.	`# nisaddcred ...` `# nisgrpadm -a admin.wiz.com.` *member* **…**

Setting Up NIS+ Clients 5

This chapter provides step-by-step instructions for using the NIS+ command set to perform the following tasks:

Client Setup	*page 90*
Initializing an NIS+ Client	*page 98*
Host-Name Initialization	*page 100*
Cold-Start File Initialization	*page 101*
Changing a Workstation's Domain	*page 96*

Note – It is much easier to perform this task with the NIS+ installation scripts as described 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.

See "Configuration Worksheets" on page 7 for worksheets that you can use to plan your NIS+ namespace.

If you have them available, the Solstice AdminSuite tools provide easier methods of adding and setting up NIS+ client machines.

This chapter describes how to set up 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 on Table 5-1 on page 103

Note that at Step 8 in the client setup instructions you must choose which of three methods to use: broadcast, host name, or cold-start file. Since 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 9.

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

Client Setup

This section describes how to set up a typical NIS+ client in either the root domain or in 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 don't want to create a sales.west.myco.com subdirectory.

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 setup 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. Logging in as superuser to the client
4. Assigning the client its new domain name
5. Checking the client's `nsswitch.conf` file

6. Cleaning out leftover NIS+ material and processes.
7. Initializing the client.
8. Killing and restarting the `keyserv` daemon.
9. Running `keylogin`.
10. 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 them to be created by an administrator who has 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.

Once an administrator creates the client's credentials, the client can complete the setup process. However, the client still needs read access to the directory object of its home domain. If you set up the client's home domain according to the instructions in either Chapter 4, "Setting Up the Root Domain," or Chapter 7, "Setting Up a Nonroot Domain," 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 Wiz.Com.
ObjectName    : Wiz
Owner         : rootmaster.Wiz.Com.
Group         : admin.Wiz.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 *NIS+ and FNS 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 set up and running NIS+
- An entry in either the master server's `/etc/hosts` file or in its domain's hosts 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

▼ How to Set Up 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 -P 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 Wiz.Com. domain:

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

   For more information about the `nisaddcred` command, see the credentials chapter of *NIS+ and FNS Administration Guide.*

3. **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.

4. **Assign the client its new domain name.**
   There are three ways to assign a new domain name to a client. Those methods are described in "Changing a Workstation's Domain" on page 96. Use one of those methods to change the client's domain name and then return to Step 5 below.

5. **Check the client's** `nsswitch.conf` **file.**
   Make sure the client is using the NIS+ version of the `nsswitch.conf` file. This ensures that the primary source of information for the client will be NIS+ tables. Code Example 5-1 shows the correct version of the file.

Code Example 5-1 NIS+ Version of `nsswitch.conf` File

```
# /etc/nsswitch.nisplus:
#
# An example file that could be copied over to /etc/nsswitch.conf; it
# uses NIS+ (NIS Version 3) in conduction with files.
#
# hosts: and services: in this file are used only if the /etc/netconfig
# file contains switch.so as a nametoaddr library for 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+
#hosts:      nisplus dns [NOTFOUND=return] files

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

If the file does not look like the one above, change it to the version recommended for NIS+. (Complete instructions are provided in Chapter 9, "Setting Up the Name Service Switch," and an example is shown in Step 6 below).

6. **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 9, "Setting Up the Name Service Switch," tell you to kill and restart the keyserver at this point, you don't need to do that in this case, since you will do so in Step 9.)

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`.

```
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 setup process. If you have stored any admin scripts in `/var/nis`, you may want to consider temporarily storing them elsewhere, until you finish setting up the root domain.

8. **Initialize the client.**
You can initialize a client in three different ways: by host name, by cold-start file, or by broadcast (see "Initializing an NIS+ Client" on page 98). Choose and perform one of those methods. After initializing the client, proceed with Step 9.

9. **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 9.

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

10. 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
```

11. Reboot the client.

Changing a Workstation's Domain

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.

Specifying a Domain Name After Installation

A workstation is usually assigned to its domain during installation. On an operating network, the installation script usually obtains the domain name automatically and simply asks the installer to confirm it. During the installation proper, the workstation's domain name is assigned to a variable called `domainname`, which is stored in the kernel. There, it is made available to any program that needs it.

However, when a workstation is rebooted, the setting of the `domainname` variable is lost. As a result, unless the domain name is saved somewhere else, the operating system no longer knows which domain the workstation belongs to. To solve this problem, the domain name is stored in a file called `/etc/defaultdomain`.

When the workstation is rebooted, the kernel automatically obtains the domain name from this file and resets the `domainname` variable. Thus, if you change a workstation's domain name, you must also edit the `/etc/defaultdomain` file; if you don't, after the next reboot, the workstation will revert to its previous domain name.

Security Considerations

You must perform this task as superuser on the workstation whose domain name you will change.

Information You Need

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

▼ 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 Wiz.Com. as the new domain name.

```
client1% su
Password:
```

2. **Change the workstation's domain name.**
   Type the new name with the `domainname` command. Do not use a trailing dot.

```
client1# domainname Wiz.Com
```

   If the workstation was 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
Wiz.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.

   Since 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.

Initializing an NIS+ Client

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

- Broadcast method (see "Broadcast Initialization" on page 99)
- Host-name method (see "Host-Name Initialization" on page 100)

- Cold-start file method (see “Cold-Start File Initialization” on page 101)

Broadcast Initialization

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

This is the simplest way to set up 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.

Information You Need

You need the superuser password to the client.

▼ 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 Wiz.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.

Host-Name Initialization

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` file for the trusted server.

Information You Need

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

▼ How to Initialize a Client—Host-name Method

1. **Check the client's `/etc/hosts` 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 Wiz.Com. NIS+ domain.
Setting up NIS+ client ...
All done.
```

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

Cold-Start File Initialization

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 set up and running NIS+.

Information You Need

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

▼ 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
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 Wiz.Com. NIS+ domain.
Setting up NIS+ client ...
All done.
```

NIS+ Client Setup Summary

Table 5-1 shows a summary of the steps required to set up a client. 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 5-1 Setting Up a Client: Command Summary

Tasks	Commands
Log in to domain's master. Create DES credentials for client.	`rootmaster%` `rootmaster% nisaddcred -p unix.client1.Wiz.Com -P client1.Wiz.Com. des`
Log in, as superuser, to the client.	`client1% su` `Password:`

Assign the client a domain name.
Check the switch configuration file.
	`client1# domainname Wiz.Com`
	`client1# domainname > /etc/defaultdomain`
	`client1# more /etc/nsswitch.conf`

Clean out `/var/nis`. Initialize the client.	`client1# rm -rf /var/nis/*` `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.
Reboot the client.
	`client1# keylogin -r`
	`Password:`
	`client1# init 6`

5

Setting Up NIS+ Servers 6

This chapter provides step-by-step procedures for using the NIS+ command set to perform three server-related tasks:

Setting Up an NIS+ Server	*page 105*
Adding a Replica to an Existing Domain	*page 109*

Note – It is much easier to perform this task with the NIS+ installation scripts as described 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.

See “Configuration Worksheets” on page 7, for worksheets that you can use to plan your NIS+ namespace.

A summary of each task is provided at the end of the chapter.

Setting Up an NIS+ Server

This section applies to any NIS+ server except the root master; that is, root replicas, nonroot masters, and nonroot replicas, whether running in NIS-compatibility mode or not.

Standard versus NIS-Compatible Setup 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 Setup Procedures” on page 68). 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 (the equivalent step for standard NIS+ servers is Step 3).

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 setup process:

1. Logging in as superuser to the new replica server.
2. [NIS-Compatibility Only] Starting the NIS+ daemon with `-Y`.
3. [Standard NIS+ Only] Starting the NIS+ daemon.

Security Considerations

The NIS+ security system is complex. If you are not familiar with NIS+ security, you may wish to review the security-related chapters of *NIS+ and FNS Administration Guide* before starting to set up your NIS+ environment.

You must perform this operation as superuser on the server. The security level at which you start the server (Step 4) determines the credentials that its clients must have. For instance, if the server is set up with security level 2 (the default), the clients in the domain it supports must have DES credentials. If you have set up 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 setup 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 set up (see Chapter 4, “Setting Up the Root Domain”).
- The server must have already been initialized as an NIS+ client (see Chapter 5, “Setting Up NIS+ Clients”).
- If the server will run in NIS-compatibility mode and support DNS forwarding, it must have a properly configured `/etc/resolv.conf` file (described in Chapter 9, “Setting Up the Name Service Switch.”)

Information You Need

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

▼ How to Set Up an NIS+ Server

1. **Log in as superuser to the new replica server.**
   The following steps assume you rebooted the workstation after you set it up as an NIS+ client, as instructed in “Client Setup” on page 90. 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 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 *NIS+ and FNS Administration Guide.*

Caution – Do not rename the `/var/nis` directory or the `/var/nis/trans.log` or `/var/nis/data.dict` files.

Now this server is ready to be designated a master or replica of a domain, as described in Chapter 7, "Setting Up a Nonroot Domain." This step completes this task. A task summary is provided on page 112.

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 *NIS+ and FNS Administration Guide.*

Caution – Do not rename the `/var/nis` directory or the `/var/nis/trans.log` or `/var/nis/data.dict` files.

Now this server is ready to be designated a master or replica of a domain, as described in Chapter 7, “Setting Up a Nonroot Domain.” This step completes this task. A task summary is provided on page 112.

Adding a Replica to an Existing Domain

An easier way to add a replica server to an existing domain is to use the `nisserver` script as described in Chapter 3, “Setting Up NIS+ With Scripts.

This section describes how to add a replica server to an existing domain using the raw NIS+ command, whether root or nonroot. Here is a list of the steps:

1. First set up the server as described in “Setting Up an NIS+ Server” on page 105.
2. Log in to the domain’s master server.
3. Add the replica to the domain.
4. Run `nisping` on the replica.

Note – 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.

Security Considerations

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

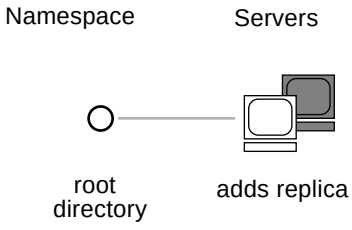
Prerequisites

- The server that will be designated a replica must have already been set up.
- The domain must have already been set up and assigned a master server.

Information You Need

- The name of the server
- The name of the domain

▼ How to Add a Replica Server



1. **Log in to the domain's master server.**

2. **Add the replica to the domain.**
   Run the `nismkdir` command with the `-s` option. The example adds the replica machine named `rootreplica` to the Wiz.Com. domain.

```
rootmaster# nismkdir -s rootreplica Wiz.Com.
rootmaster# nismkdir -s rootreplica org_dir.Wiz.Com.
rootmaster# nismkdir -s rootreplica group_dir.Wiz.Com.
```

When you run the `nismkdir` command on a directory object that already exists, it does not recreate the directory but simply 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 – 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 replicas.

3. 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 Wiz.Com. domain name in the example below is not necessary.)

```
rootmaster# nisping Wiz.Com.
rootmaster# nisping org_dir.Wiz.Com.
rootmaster# nisping group_dir.Wiz.Com.
```

You should see results similar to these:

```
rootmaster# nisping Wiz.Com.
Pinging replicas serving directory Wiz.Com. :
Master server is rootmaster.Wiz.Com.
        No last update time

Replica server is rootreplica.Wiz.Com.
        Last update seen was Wed Nov 18 11:24:32 1992

        Pinging ... rootreplica.Wiz.Com.
```

If you have set up the domain's tables immediately after completing the domain setup, this step propagates the tables down to the replica. For more information about `nisping`, see the directories chapter of *NIS+ and FNS Administration Guide*.

6

Server Setup Summary

Table 6-1 and Table 6-2 provide a summary of the tasks described in this chapter. 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. This summary does not show the server's responses to each command.

Table 6-1 Starting Up a Nonroot Master Server: Command Summary

Tasks	Commands
Log in to the server as root.	`server% su`
NIS-compat only: Start daemon with `-Y -B`. Change to `EMULYP= -Y -B`.	`server# rpc.nisd -Y - B` `server# vi /etc/inet.d/rpc`
NIS+-Only: Start daemon.	`server# rpc.nisd`

Table 6-2 Adding a Replica: Command Summary

Tasks	Commands
Log in as superuser to domain master.	`rootmaster% su`

Designate the new replica.	`# nismkdir -s rootreplica Wiz.Com.`
	`# nismkdir -s rootreplica org_dir.Wiz.Com.`
	`# nismkdir -s rootreplica groups_dir.Wiz.Com.`

Ping the replica.	`#/usr/lib/nis/nisping Wiz.Com`
	`#/usr/lib/nis/nisping org_dir.Wiz.Com`
	`#/usr/lib/nis/nisping groups_dir.Wiz.Com`

Setting Up a Nonroot Domain 7

This chapter provides step-by-step instructions for using NIS+ commands to set up a nonroot domain (also known as a subdomain). You should not set up a nonroot domain until *after* you have set up servers.

A summary of this task is provided by Table 7-1 on page 122.

Note – It is much easier to perform this task with the NIS+ installation scripts as described 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.

See “Configuration Worksheets” on page 7 for worksheets that you can use to plan your NIS+ namespace.

Setting Up a Nonroot Domain

Setting up a nonroot 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

However, as with setting up the root domain, these tasks cannot be performed sequentially. To make the setup 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 Setup 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 Setup Procedures” on page 68).

The NIS+ daemon for each server in an NIS-compatible domain should have been started with the `-Y` option, as instructed in Chapter 6. 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, in Step 4. (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 setup process:

1. Logging in to the domain’s master server.
2. Naming the domain’s administrative group.
3. Creating the domain’s directory and designate its servers.
4. Creating the domain’s subdirectories and tables.
5. Creating the domain’s admin group.
6. Assigning full group access rights to the directory object.
7. Adding the servers to the domain’s admin group.
8. Adding credentials for other administrators.
9. Adding the administrators to the domain’s admin group.

Security Considerations

The NIS+ security system is complex. If you are not familiar with NIS+ security, you may wish to review the security-related chapters of *NIS+ and FNS Administration Guide* before starting to set up 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 Wiz.Com.
Object Name   : Wiz
Owner         : rootmaster
Group         : admin.Wiz.Com.
Domain        : Com.
Access Rights : r---rmcdrmcdr---
:
```

If you are more concerned about convenience than security, you could simply make the new domain's master server a member of its parent domain's admin group and then perform the entire procedure from the server. Use the `nisgrpadm` command, described in the groups chapter of *NIS+ and FNS Administration Guide.*

Prerequisites

- The parent domain must be set up 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` file or from its NIS+ hosts table.

Information You Need

- The name of the new domain (for Step 3)
- The name of the new domain's master and replica servers
- The name of the new domain's admin group (for Step 2)
- User IDs (UID) of the administrators who will belong to the new domain's admin group (for Step 8)

▼ How to Set Up a Nonroot 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 Wiz.Com. domain, and will become the master server of the Sales.Wiz.Com. subdomain. The administrator performing this task is `nisboss.Wiz.Com.`, a member of the `admin.Wiz.Com.` group. That group has full access rights to the `Wiz.Com.` directory object.

2. **Name the domain's administrative group.**
   Although you won't actually create the admin group until Step 5, 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.

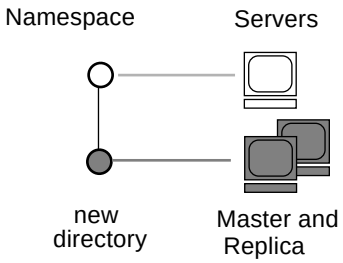
   Set the value of the environment variable `NIS_GROUP` to the name of the domain's admin group. Here are two examples, one for C shell users and one for Bourne or Korn shell users. They both set `NIS_GROUP` to `admin.Sales.Wiz.Com.`

   For C Shell

```
smaster# setenv NIS_GROUP admin.Sales.Wiz.Com
```

   For Bourne or Korn Shell

```
smaster# NIS_GROUP=admin.Sales.Wiz.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.

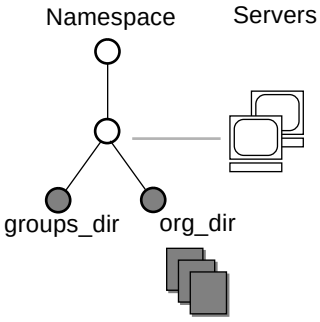
```
smaster# nismkdir -m smaster -s salesreplica Sales.Wiz.Com.
```

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.Wiz.Com.
Object Name   : Sales
Owner         : nisboss.Wiz.Com.
Group         : admin.Sales.Wiz.Com.
Domain        : Wiz.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.Wiz.Com.
```

   NIS+

```
smaster# /usr/lib/nis/nissetup  Sales.Wiz.Com.
```

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

```
smaster# /usr/lib/nis/nissetup
org_dir.Sales.Wiz.Com. created
groups_dir.Sales.Wiz.Com. created
auto_master.org_dir.Sales.Wiz.Com. created
auto_home.org_dir.Sales.Wiz.Com. created
bootparams.org_dir.Sales.Wiz.Com. created
cred.org_dir.Sales.Wiz.Com. created
ethers.org_dir.Sales.Wiz.Com. created
group.org_dir.Sales.Wiz.Com. created
hosts.org_dir.Sales.Wiz.Com. created
mail_aliases.org_dir.Sales.Wiz.Com. created
sendmailvars.org_dir.Sales.Wiz.Com. created
netmasks.org_dir.Sales.Wiz.Com. created
netgroup.org_dir.Sales.Wiz.Com. created
networks.org_dir.Sales.Wiz.Com. created
passwd.org_dir.Sales.Wiz.Com. created
protocols.org_dir.Sales.Wiz.Com. created
rpc.org_dir.Sales.Wiz.Com. created
services.org_dir.Sales.Wiz.Com. created
timezone.org_dir.Sales.Wiz.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 8 (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. Use the `nisgrpadm` command with the `-c` option. This example creates the `admin.Sales.Wiz.Com.` group

```
smaster# nisgrpadm -c admin.Sales.Wiz.Com.
Group admin.Sales.Wiz.Com. created.
```

   This step only creates the group—it does not identify its members. That is done in Step 9.

6. **Assign full group access rights to the directory object.**
   By default, the directory object only grants its group read access, which makes the group no more useful than the world class. To make the setup 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.Wiz.Com.
```

   Complete instructions for using the `nischmod` command are provided in the rights chapter of *NIS+ and FNS 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.Wiz.Com.` and `salesreplica.Wiz.Com.` to the `admin.Sales.Wiz.Com.` group:

```
smaster# nisgrpadm -a admin.Sales.Wiz.Com.   smaster.Wiz.Com. salesreplica.Wiz.Com.
Added smaster.Wiz.Com. to group admin.Sales.Wiz.Com.
Added salesreplica.Wiz.Com. to group admin.Sales.Wiz.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 *NIS+ and FNS Administration Guide).*

```
smaster# nisgrpadm -l admin.Sales.Wiz.Com.
Group entry for admin.Sales.Wiz.Com. group:
    Explicit members:
        smaster.Wiz.Com.
        salesreplica.Wiz.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, simply add LOCAL credentials. Use the `nisaddcred` command with both the `-p` and the `-P` flags.

```
smaster# nisaddcred -p 33355 -P nisboss.Wiz.Com. local
```

   For administrators that 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.Wiz.Com.` adds his own credentials to the Sales.Wiz.Com. domain.

```
smaster# nisaddcred -p 22244 -P juan.Sales.Wiz.Com. local
smaster# nisaddcred -p unix.22244@Sales.Wiz.Com -P juan.Sales.Wiz.Com. des
Adding key pair for unix.22244@Sales.Wiz.Com.
Enter login password:
```

- The other way is for you to create temporary credentials for the other administrators, using dummy passwords (note that each administrator must have an entry in the NIS+ passwd table).

```
smaster# nisaddcred -p 22244 -P juan.Sales.Wiz.Com. local
smaster# nisaddcred -p unix.22244@Sales.Wiz.Com -P juan.Sales.Wiz.Com. des
Adding key pair for unix.22244@Sales.Wiz.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 *NIS+ and FNS Administration Guide* describe how to do this.

Note – In the two Step 8 example shown above, 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.Wiz.Com.` group:

```
smaster# nisgrpadm -a admin.Sales.Wiz.Com. juan.Sales.Wiz.Com.
Added juan.Sales.Wiz.Com. to group admin.Sales.Wiz.Com.
```

7

Subdomain Setup Summary

Table 7-1 is a summary of the steps required to set up 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 7-1 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.Wiz.Com.` `# export NIS_GROUP`
Create the domain's directory and designate its servers.	`# nismkdir -m smaster -s salesreplica Sales.Wiz.Com.`
Create `org_dir`, `groups_dir`, and tables. (For NIS-compatibility, use `-Y`.)	`# /usr/lib/nis/nissetup Sales.Wiz.Com.`
Create the admin group.	`# nisgrpadm -c admin.Sales.Wiz.Com.`
Assign full group rights to the domain's directory.	`# nischmod g+rmcd Sales.Wiz.Com.`
Add servers to admin group.	`# nisgrpadm -a admin.Sales.Wiz.Com. smaster.Wiz.Com. \` `  sreplica.Wiz.Com.`

Add credentials for other admins.	`# nisaddcred -p 22244 -P juan.Sales.Wiz.Com. local`
	`# nisaddcred -p unix.22244@Sales.Wiz.com. \`
	`  juan.Sales.Wiz.Com. DES`

Add admins to domain's admin group.	`# nisgrpadm -a admin.Sales.Wiz.Com. juan.Sales.Wiz.Com.`

Setting Up NIS+ Tables 8

If you have them available, the Solstice AdminSuite tools provide easier methods of working with NIS+ tables.

This chapter provides step-by-step instructions for using the NIS+ command set to populate NIS+ tables on a root or nonroot master server from `/etc` files or NIS maps. This chapter also describes how to transfer information back from NIS+ tables to NIS maps, a procedure that may be required during a transition from NIS to NIS+. Finally, it includes two tasks that describe how to limit access to the passwd column of the passwd table:

Populating Tables—Options	*page 124*
Populating NIS+ Tables From Files	*page 125*
Populating NIS+ Tables From NIS Maps	*page 131*
Transferring Information From NIS+ to NIS	*page 136*
Limiting Access to the Passwd Column to Owners and Administrators	*page 137*

Note – It is much easier to perform this task with the NIS+ installation scripts as described 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.

See “Configuration Worksheets” on page 7 for worksheets that you can use to plan your NIS+ namespace.

You can populate NIS+ tables in four ways:

- From files, as described in “Populating NIS+ Tables From Files” on page 125

- From NIS maps, as described in “Populating NIS+ Tables From NIS Maps” on page 131
- With the `nispopulate` script, as described in “Populating NIS+ Tables” on page 30 and “Populating the New Domain’s Tables” on page 52
- 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 4, “Setting Up the Root Domain,” and Chapter 7, “Setting Up a Nonroot Domain.” 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 three 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 simply adds the source entries to the NIS+ table.
- *Merge* - The merge option produces the same result as the replace option but uses a different process, one that can greatly reduce the number of operations that must be sent to the replicas. 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. Checking the content of each file that you will be transferring data from.
2. Making a copy of each file. Using this copy to make the actual transfer from. (In this guide, copies of files to be transferred have names ending in `xfr` (for example, `hosts.xfr`).
3. Logging in to an NIS+ client. (You must have credentials and permissions allowing you to update the tables. See “Security Considerations,” below.)
4. Adding `/usr/lib/nis` to the search path for this shell (if not already done).
5. Using `nisaddent` to transfer any of these files one at a time: `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`, `shadow`.
6. Transferring the `publickey` file.
7. Transferring the automounter information.
8. Checkpointing the tables.

Security Considerations

You can perform this task from any NIS+ client, including the root master server, as long as you 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 wish to review the security-related chapters of *NIS+ and FNS 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 "Disk Space and Memory Recommendations" on page 4.
- 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 30 for information on the 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 would need to remove.
- No duplicate machine and user names. 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 text files that will be transferred.

▼ 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 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 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
```

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. Since 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.

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:/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`, `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; for the automounter files, go to Step 7; for the `passwd` and `shadow` files, go to Step 8.

   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 -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 simply 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.Wiz.Com.` master server belongs to the Wiz.Com. domain. To populate tables in the Sales.Wiz.Com. domain from that master server, you would have to append the Sales.Wiz.Com. domain name to the `nisaddent` statement.

```
salesmaster# nisaddent -f /etc/hosts.xfr hosts Sales.Wiz.Com.
```

If you performed this operation as a client of the Sales.Wiz.Com. domain, you would not need to append the domain name to the syntax. For more information about `nisaddent`, see the tables chapter of *NIS+ and FNS 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 *NIS+ and FNS Administration Guide.*

```
rootmaster# niscat group.org_dir
root::0:root
other::1::
bin::2:root,bin,daemon
.
.
.
```

6. Transfer the `publickey` file.

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` text file that you transfer into the cred table. You can avoid this by removing those credentials from the `publickey` text file. For `rootmaster`, that line would be:

```
unix.rootmaster@Wiz.Com public-key:private-key
```

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 only transfers DES credentials into the cred table. You will 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. **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 have just set 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.Wiz.Com. :
Master server is rootmaster.Wiz.Com.
     Last update occurred at July 14, 1994

Master server is rootmaster.Wiz.Com.
checkpoint succeeded.
```

   If you don't have enough swap space, the server will be unable to checkpoint properly, but it won't notify you. One way to make sure all went well is to list the contents of a table with the `niscat` command. If you don't 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. Checking the content of each NIS map that you will be transferring data from.
2. Logging in to an NIS+ client.
3. Adding `/usr/lib/nis` to the search path for this shell.

4. Using `nisaddent` to transfer any of these maps, one at a time: `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`.
5. Transferring the `publickey` map.
6. Transferring the automounter information.
7. Checkpointing the tables.

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 only need create rights.

After you complete this operation, the table entries will be 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`.
- No duplicate machine and user names. 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.

▼ 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.**
   Since 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:/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, and for the automounter files, go to Step 7.

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 correspond to the *table* argument. Here are some examples:

```
rootmaster# nisaddent -m -y oldwiz hosts
rootmaster# nisaddent -m -y oldwiz passwd
rootmaster# nisaddent -m -y oldwiz groups
```

The first example transfers the contents of the `hosts.byname` and `hosts.byaddr` maps in the oldwiz (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 *NIS+ and FNS 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 and then open that file with your text editor.**

```
rootmaster# makedbm -u /var/yp/oldwiz/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`, that line would be:

```
unix.rootmaster@Wiz.Com public-key:private-key
```

   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
```

   Note, however, that this operation transfers only DES credentials into the cred table. You will 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 oldwiz -Y auto.master -t auto_master.org_dir key-value
rootmaster# nisaddent -y oldwiz -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, oldwiz). 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. **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.Wiz.Com. :
Master server is rootmaster.Wiz.Com.
     Last update occurred at July 14, 1994

Master server is rootmaster.Wiz.Com.
checkpoint succeeded.
```

If you don't have enough swap space, the server will be unable to checkpoint properly, but it won't notify you. One way to make sure all went well is to use list the contents of a table with the `niscat` command. If you don't 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.

Transferring Information From NIS+ to NIS

This task transfers the contents of NIS+ tables into the NIS maps on a Solaris 1.x NIS master server. Here is a an outline of the procedure:

1. Logging in to the NIS+ server.
2. Transferring the NIS+ tables in to output files.

3. Transferring the contents of the output files to the NIS maps.

Security Considerations

To perform this task, you must have read access to each table whose contents you transfer.

Prerequisites

The maps must have already been built on the NIS server.

▼ 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 in 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, and 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 only 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   ----
```

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 2.3 or a later release.
- Users' network passwords (used to encrypt their DES credentials) must be the same 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.

▼ 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
             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 *NIS+ and FNS 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. 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 8-1 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 8-2 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 oldwiz hosts`** `% .` `% .` `% .`
Dump `publickey` map to a file.	**`% makedbm -u /var/yp/oldwiz/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 oldwiz -Y auto.master -t auto_master.org_dir key-value`**
Checkpoint the table directory.	**`% nisaddent -y oldwiz -Y auto.home -t auto_home.org_dir key-value`** **`% nisping -C org_dir`**

Table 8-3 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 8-4 Limiting Acces 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`**

8

Setting Up the Name Service Switch 9

This section provides step-by-step instructions for using the name service switch.

Selecting an Alternate Configuration File	*page 143*
Enabling an NIS+ Client to Use DNS	*page 145*
Adding Compatibility With +/- Syntax	*page 146*

Note – It is much easier to perform this task with the NIS+ installation scripts as described 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.

For information on customizing or modifying an `nsswitch.conf` file, see *NIS+ and FNS Administration Guide.*

Selecting an Alternate Configuration File

This section describes how to select an alternate switch-configuration file for an NIS+ client. 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.

Here is a list of the steps:

1. Logging in as superuser to the client.
2. Copying the alternate file over the `nsswitch.conf` file.
3. Rebooting the workstation now. (This is necessary because `nscd` caches the switch information which it reads only at start up.)

Security Considerations

You must perform this operation as superuser.

▼ How to Select an Alternate Configuration File

1. **Log in as superuser to the client.**
2. **Copy the alternate file over the `nsswitch.conf` file.**
   The `/etc/nsswitch.conf` file is the working configuration file used by the name service switch. Also in the `/etc` directory are three alternate versions of the file: one for NIS+, one for NIS, and one for local files. To select one, simply copy it over the working file. Of course, you can create additional alternates. Here are four examples:

   NIS+ version

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

   NIS version

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

   Local files version

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

Custom version

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

3. **Reboot the workstation now.**
   Because the `nscd` name service cache daemon caches switch information and 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 an NIS+ Client to Use DNS

This section describes how to set up the name service switch configuration file so that an NIS+ client can also use the Domain Name System (DNS). Here is a list of the steps:

1. Logging in as superuser.
2. Opening the `/etc/nsswitch.conf` file.
3. Specifying DNS as a source of hosts information.
4. Saving the file and reboot the workstation.

Prerequisites

The NIS+ client must have a properly configured `/etc/resolv.conf` file (as described in “Creating the resolv.conf File” on page 167).

Security Considerations

You must perform this operation as superuser.

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

   Do *not* use the above syntax for NIS clients, since it will make them look for unresolved names twice in DNS. If you have NIS servers doing DNS forwarding, use the `-B` flag.

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. Here is a list of the steps:

1. Logging in as superuser.
2. Opening the `/etc/nsswitch.conf` file.
3. Changing the passwd and group sources to `compat`.
4. Adding + or + netgroup to `/etc/passwd`, `/etc/shadow` and `/etc/group`.
5. Saving the file and reboot the workstation.

Security Considerations

You must perform this operation as superuser.

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

▼ 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`**.**

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

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.

If you would like to use the +/- semantics with NIS+ instead of NIS, add a `passwd_compat: nisplus` entry to the `nsswitch.conf` file after the passwd or group entry, as shown below:

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

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

Caution – If you fail to add the `+` or `+ netgroup` entries to `/etc/shadow` and `/etc/passwd`, you won't 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.

9

Part 3 — DNS Setup

This part of the manual describes how to set up and administer DNS. It has three chapters.

Introduction to DNS	*page 151*
Setting Up DNS Clients	*page 167*
Setting Up DNS Servers	*page 169*

Introduction to DNS 10

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

This chapter describes the purpose and structure of DNS. Refer to Chapter 11, “Setting Up DNS Clients,“and Chapter 12, “Setting Up DNS Servers,” for specific setup procedures. If you are already familiar with DNS, you may want to skip ahead to these chapters.

DNS Administrative Domains	*page 154*
in.named and DNS Name Servers	*page 155*
DNS Clients and the Resolver	*page 156*
Introducing the DNS Namespace	*page 156*
Administering DNS	*page 162*

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

10

DNS Basics

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

Name-to-Address Resolution

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

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

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

Figure 10-1 shows name-to-address mapping as it occurs between a DNS client and a name server, probably on the client's local network.

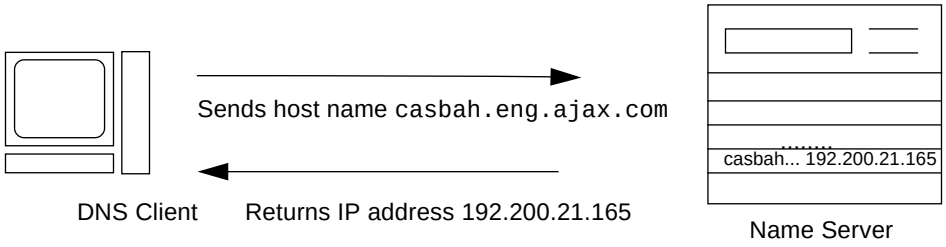


Figure 10-1 Name to Address Resolution

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

Fortunately, the local name server maintains a list of host names and IP addresses of *root domain name servers*, to which it will forward the request from your machine. These root name servers are authoritative for huge organizational domains, as explained fully in "DNS Hierarchy and the Internet" on page 158. These hierarchies resemble UNIX file systems, in that they are organized into an upside-down tree structure.

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

Figure 10-2 shows name-to-address resolution outside the local domain.

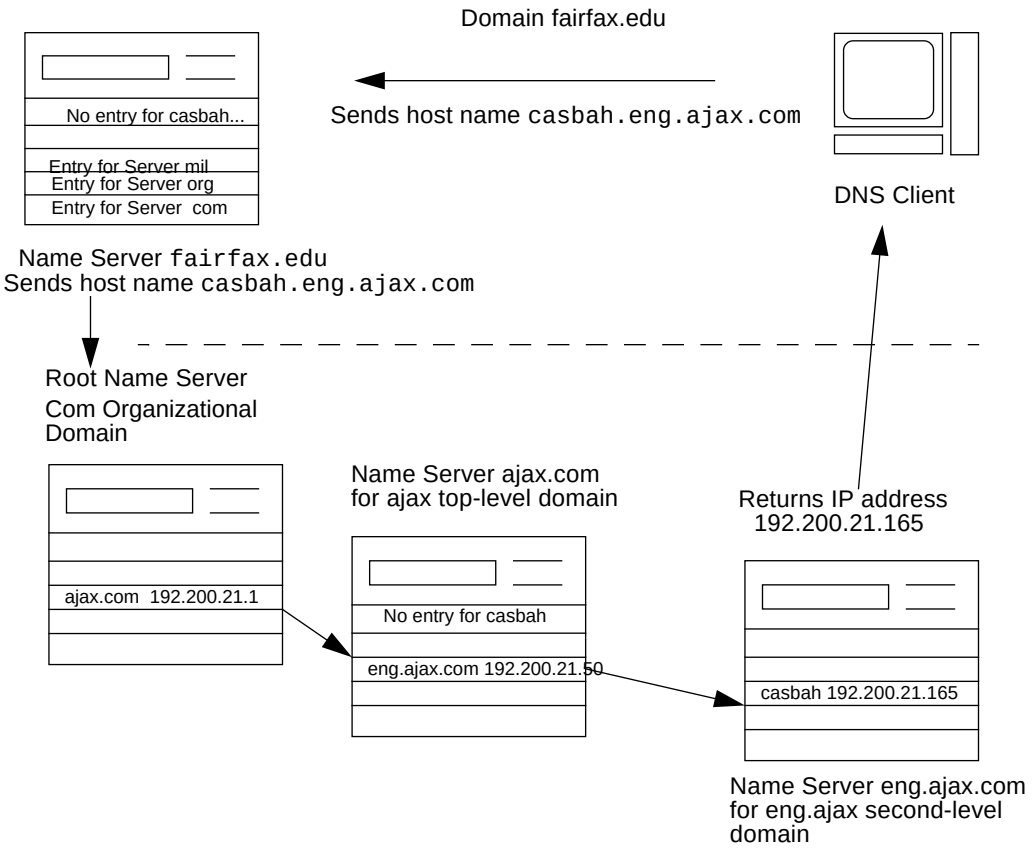


Figure 10-2 Name to Address Resolution for a Remote Host

DNS Administrative Domains

From a DNS perspective, an *administrative domain* is a group of machines that are administered as a unit. Information about this domain is maintained by at least two name servers; they are “authoritative” for the domain. The DNS domain is a purely logical grouping of machines. It could correspond to a physical grouping of machines, such as all machines attached to the Ethernet in

a small business. But a local DNS domain just as likely could include all machines on a vast university internetwork that belong to the computer science department or to university administration.

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

Each administrative domain must have its own unique subdomain name. Moreover, if you want your network to participate in the Internet, the network must be part of a registered administrative domain. The section “Joining the Internet” on page 160 has full details about domain names and domain registration.

`in.named` *and DNS Name Servers*

As mentioned previously, name servers in an administrative domain maintain the DNS database. They also run the `in.named` daemon, which implements DNS services, most significantly, name-to-address mapping. `in.named` is a standard TCP/IP program and included with the Solaris 2.5 operating environment.

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

There are three types of DNS name servers:

- Primary server
- Secondary server
- Cache-only server

Each domain must have one primary server and at least one secondary server to provide backup. “Administering DNS” on page 162 explains primary and secondary servers in detail.

10

DNS Clients and the Resolver

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

When the `/etc/nsswitch.conf` file specifies dns first, the resolver libraries are automatically used.

There are two kinds of DNS clients:

- Client-only
- Client-server

A client-only DNS client does not run `in.named`; instead, it consults the resolver. The resolver provides a list of name servers for the domain, to which queries are then directed. A client-server client uses the services provided by `in.named` to resolve a user's queries.

The Solaris 2.5 operating environment includes the library routines making up the resolver by default. Chapter 11, "Setting Up DNS Clients," contains instructions for setting up a host as a DNS client.

Introducing the DNS Namespace

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

DNS Namespace Hierarchy

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

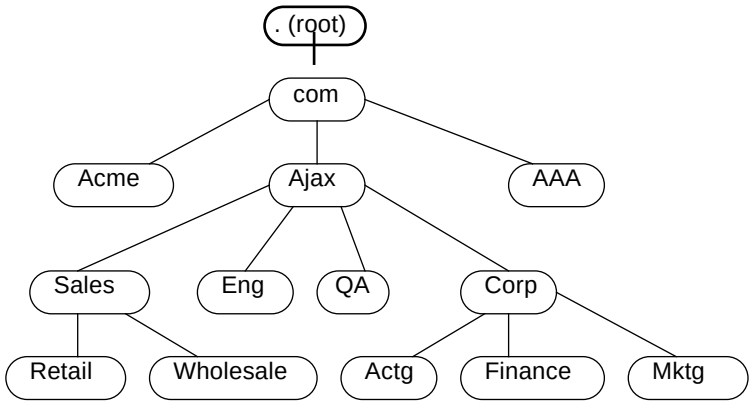


Figure 10-3 Domains and Subdomains

For example, in Figure 10-3, `com` is a parent domain to the `Acme`, `Ajax`, and `AAA` domains. Or you could just as easily say that those are subdomains relative to the `com` domain. In its turn, the `Ajax` domain is a parent to four subdomains (`Sales`, `Eng`, `QA`, and `Corp`).

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

DNS Hierarchy in a Local Domain

If your company is large enough, it may support a number of domains, organized into a local namespace. Figure 10-4 shows a domain hierarchy that might be in place in a single company. The top-level, or “root” domain for the organization is `ajax.com`, which has three sub-domains, `sales.ajax.com`, `test.ajax.com`, and `eng.ajax.com`.

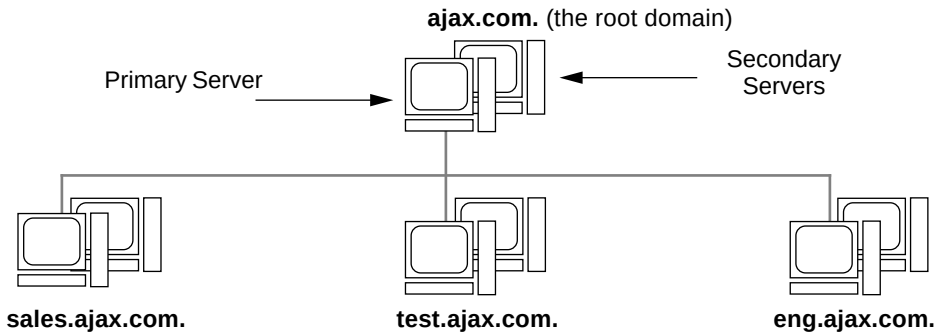


Figure 10-4 Hierarchy of DNS Domains in a Single Organization

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

DNS Hierarchy and the Internet

The domain hierarchy shown in Figure 10-4 on page 158 is, conceptually, a "leaf" of the huge DNS namespace supported on the Internet.

The DNS namespace for the Internet is organized hierarchically, as shown in Figure 10-5. It consists of the root directory, represented as a dot (.) and two main domain hierarchies, one organizational and one geographical. Note that the `com` domain introduced inFigure 10-3 on page 157 is one of a number of top-level organizational domains in existence on the Internet.

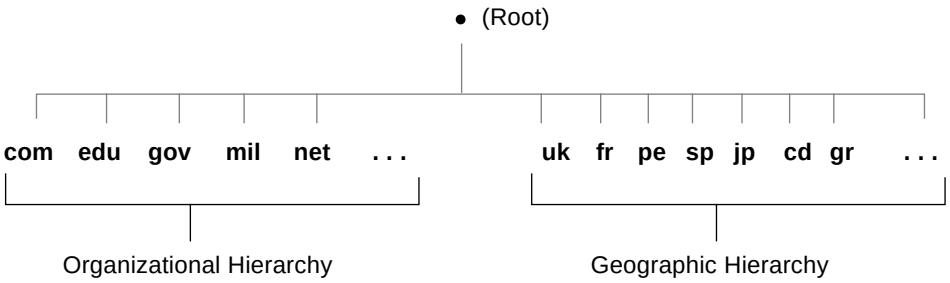


Figure 10-5 Hierarchy of Internet Domains

The organizational hierarchy divides its namespace into the top-level domains listed in Table 10-1.

Table 10-1 Internet Organizational Domains

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

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

Joining the Internet

The Internet root domain, top-level organizational domains, and top-level geographic domains are maintained by the Internet governing bodies. Organizations with networks of any size can join the Internet by applying for membership in either the organizational or the geographical hierarchy.

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

To join the Internet, you or another network administrator has to:

- Register your network and obtain a network number from the Internet governing bodies.
- Register your DNS domain with the Internet governing bodies.

There are two ways to accomplish this. You can directly contact the InterNIC, currently the organization that handles network address and domain registration. See *TCP/IP and Data Communications Administration Guide* for addresses and instructions.

But today, the more common approach is to employ an Internet Service Provider (ISP) to assist you. ISPs can help set up your physical Internet connection, register your network, and assist you with DNS issues. Some ISPs may provide secondary DNS name servers to back up the primary server at your site. If your network is small, some ISPs may include it in a local domain that they administer. Contact the various regional and national ISPs listed in your phone book and computer trade magazines to find the Internet Service Provider that best supports your site's needs.

Domain Names

Domain names indicate a domain's position in the overall DNS namespace, much as path names indicate a file's position in the UNIX file system. After your local domain is registered, its name is prepended to the name of the Internet hierarchy to which it belongs. For example, the `ajax` domain shown in Figure 10-4 on page 158 has been registered as part of the Internet `com` hierarchy. Therefore, its Internet domain name becomes `ajax.com`.

Figure 10-6 shows the position of the `ajax.com` domain in the DNS namespace on the Internet.

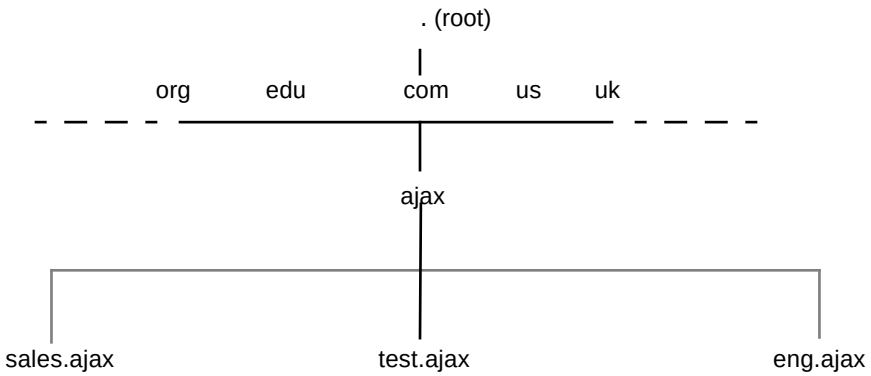


Figure 10-6 Wiz Domain's Position in the DNS Namespace

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

```
sales.ajax.com
test.ajax.com
eng.ajax.com
```

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

```
Boss.ajax.com
quota.Sales.ajax.com
```

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

Fully-Qualified Domain Names

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

<local_domain_name>.<Internet_Org_name> **.**

root domain

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

```
ajax.com.
test.ajax.com.
eng.ajax.com.
```

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

Administering DNS

DNS service for a domain is managed on the set of name servers first introduced on page 155. Name servers can manage a single domain, or multiple domains, or domains and some or all of their corresponding subdomains. The part of the namespace that a given name server controls is called a *zone*; thus, the name server is said to be authoritative for the zone. If you are responsible for a particular name server, you may be given the title zone administrator.

Zone s

The data in a name server's database are called *zone files*. One type of zone file stores IP addresses and host names. When someone attempts a remote procedure such as `ftp` or `telnet`, the file provides the name of the remote host. DNS performs name-to-address mapping, by look up the host name in the zone file and converting it into its IP address.

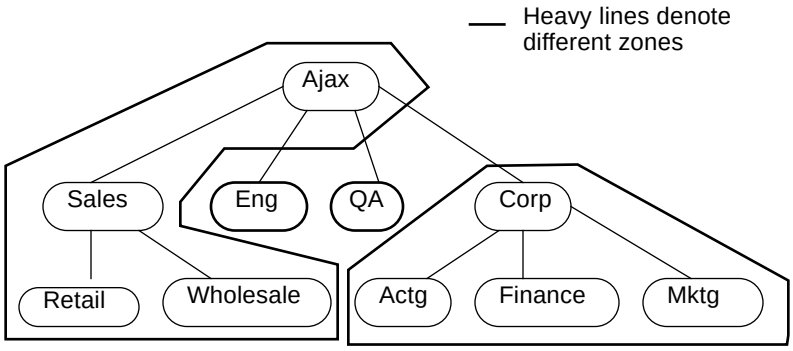


Figure 10-7 Domains and Zones

For example, the `Ajax` domain shown in Figure 10-7 contains a top domain (`Ajax`), four subdomains, and five sub-subdomains. It is divided into four zones shown by the thick lines. Thus, the `Ajax` name server administers a zone composed of the `Ajax`, `Sales`, `Retail`, and `Wholesale` domains. The `R&D` and `QA` domains are zones unto themselves served by their own name servers, and the `Corp` name server manages a zone composed of the `Corp`, `Actg`, `Finance`, and `Mktg` domains.

Reverse Mapping

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

The `in.addr.arpa` *Domain*

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

Just as DNS domain names are read with the lowest level subdomain occupying the furthest left position and the root at the far right, in.addr.arpa domain IP addresses are read from lowest level to the root. Thus, the IP addresses are read backward. For example, suppose a host has the IP address 192.200.21.165. In the `in.addr.arpa` zone files, its address is listed as `165.21.200.192.in.addr.arpa.` with the dot at the end indicating the root of the `in.addr.arpa` domain.

Master Servers

The *master* name servers maintain all the data corresponding to the zone, making them the authority for that zone. These are commonly called *authoritative* name servers. The data corresponding to any given zone should be available on at least *two* authoritative servers. You should designate one name server as the primary master server and at least one as a secondary master server, to act as a backup if the primary is unavailable or overloaded.

Primary Name Server

The *primary* master server is the name server where you make changes for the zone. This server loads the master copy of its data from disk when it starts `in.named`. The primary server may also delegate authority to other servers in its zone as well as to servers outside its zone.

Secondary Name Server

The *secondary* master server maintains a copy of the data for the zone. The primary server sends its data and delegates its authority to the secondary server. When the secondary server boots `in.named`, it requests all the data for the given zone from the primary. The secondary server then periodically checks with the primary to see if it needs to update its database. The process of sending the most recent zone database from the primary to the secondary is called a *zone transfer.*

A server may function as a master for multiple zones: as a primary for some zones, and as a secondary for others.

Root Domain Name Server

The DNS name space must have a *root domain name server*. If your site is not connected to the Internet, you must set up a root domain for your organization and administer primary and secondary name servers for the root level of the local network.

Caching and Caching-Only Servers

All name servers are *caching servers*. This means that the name server caches received information until the data expires. The expiration process is regulated by the time-to-live field attached to the data when it is received from another server.

Additionally, you can set up a *caching-only server* that is not authoritative for any zone. This server handles queries and asks other name servers that have the authority for the information needed. But the caching-only server does not maintain any authoritative data itself.

How DNS Affects Mail Delivery

DNS provides two principal services, it performs name to address mapping (and also maps addresses to host names), as discussed in on page 152. It also helps mail delivery agents, such as `sendmail` and POP, deliver mail along the Internet.

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

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

Table 10-2 Domains and Mail Hosts

DNS Domain	Mail Host
International.com.	129.44.1.1
sales.ajax.com.	SalesAjaxMailer
eng.ajax.com.	EngAjaxMailer
Fab.com.	FabMailer

When the mail agent receives a request to send mail to another domain, it parses the address of the recipient from right to left and looks for a match in the table.

If it receives a request to send mail to `neverhome.sales.ajax.com`, it first extracts the topmost label, `com`. It examines the mail exchange record to see if there is an entry for `com`. Since there is none, it continues parsing. It extracts the next label and looks for an entry for `ajax.com`. Since there is none, it continues looking. The next entry it looks for is `sales.ajax.com`. As you can see in Table 10-2, the mail host for that domain is `SalesAjaxMailer`. Because that is a host name, the mail agent asks DNS to resolve it. When DNS provides that mail host's IP address, the mail agent sends the message.

If, instead of the mail host name, the mail exchange record had specified an IP address, the mail agent would have sent the message directly to that address, since it would have needed no name resolution from DNS.

Setting Up DNS Clients 11

Setting up DNS on a client involves two tasks, which are described in this chapter.

Creating the resolv.conf File	*page 167*
Modifying the /etc/nsswitch.conf File	*page 168*

If you are setting up DNS on a name server, you need to complete these tasks in addition to setting up boot and data files. The server tasks are described in Chapter 12, “Setting Up DNS Servers.”

Creating the `resolv.conf` *File*

DNS clients use the 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 the information needed. The resolver reads the `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. Every DNS client system on your network must have a `resolv.conf` file in its `/etc` directory.

11

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.

Format of `/etc/resolv.conf`

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). Succeeding lines list the IP addresses that the resolver should consult to resolve queries. IP address entries have the form:

```
nameserver IP_address
```

Code Example 11-1 shows a sample `resolv.conf` file:

Code Example 11-1 Sample `resolv.conf` File

```
; Sample resolv.conf file
domain University.Edu.
; try local name server
nameserver 127.0.0.1
; if local name server down, try these servers
nameserver 128.32.0.4
nameserver 128.32.0.10
```

Modifying the `/etc/nsswitch.conf` *File*

To use DNS as the source of host-name information, follow the directions for enabling an NIS+ client to use DNS in “Enabling an NIS+ Client to Use DNS” on page 145. For additional information on the `nsswitch.conf` file, see *NIS+ and FNS Administration Guide.*

Setting Up DNS Servers 12

This chapter describes how to set up a DNS name server. If you need background information on DNS servers, refer to Chapter 10, “Introduction to DNS.”

Introduction to Boot and Data Files	*page 170*
Setting Up the Boot File	*page 172*
Setting Up the Data Files	*page 175*
Modifying the Data Files	*page 190*
A Practical Example	*page 191*
Setting Up a Root Server for a Local Network	*page 198*

Because every name server is a client of other name servers, you must complete the tasks involved in setting up DNS on a client before you set up a machine to be a name server, as described in Chapter 11, “Setting Up DNS Clients.”

Once you complete these tasks, you can then create the boot and data files that the name server daemon, `in.named`, uses. Instructions for creating these files appear in this chapter. The server’s initialization script (`/etc/init.d/inetsvc`) will automatically start the `in.named` daemon when the boot file (`/etc/named.boot`) is properly installed.

12

Note – If your local network is not connected to the Internet, you must set up primary and secondary servers in the root-level domain on the local network. Instructions for setting up a root domain name server appear in “Setting Up a Root Server for a Local Network” on page 198.

Introduction to Boot and Data Files

In addition to the `in.named` daemon, DNS on a name server consists of a boot file and local data files.

The location of the boot file is `/etc/named.boot`. (See “Setting Up the Boot File” on page 172 and Code Example 12-1 through Code Example 12-3 starting on page 172.) The boot file contains a list of domain names and the file names containing host information. Common names for the local data files are:

- `named.ca` – See “Setting Up the named.ca File” on page 180 and Code Example 12-4 on page 180.
- `hosts` – See “Setting Up the hosts File” on page 180 and Code Example 12-5 on page 181.
- `hosts.rev` – See “Setting Up the hosts.rev File” on page 189 and Code Example 12-22 on page 189.
- `named.local` – See “Setting Up the named.local File” on page 189 and Code Example 12-23 on page 190.

These names are used in the descriptions of these files that follow. However, you can name these files whatever you wish.

The `named.boot` *File*

The boot file `named.boot` establishes the server as a primary, secondary, or caching-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 boot file is read by `in.named` when the daemon is started by the server’s start up script, `/etc/init.d/inetsvc`. The boot file directs `in.named` either to other servers or to local data files for a specified domain. (See “Setting Up the Boot File” on page 172 and Code Example 12-1 through Code Example 12-3 starting on page 172.)

The `named.ca` *File*

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

The `hosts` *File*

The `hosts` file contains all the data about the machines in the local zone. The name of this file is specified in the boot file. To avoid confusion with `/etc/inet/hosts`, name the file something other than `hosts`. In the sample boot file shown in Code Example 12-1, the `hosts` file is called `mydomain.zone`.

(See “Setting Up the hosts File” on page 180 and Code Example 12-5 on page 181.)

The `hosts.rev` *File*

The `hosts.rev` file specifies a zone in the `in-addr.arpa.` domain, the special domain that allows reverse (address-to-name) mapping. The name of this file is specified in the boot file. In the sample boot file shown in Code Example 12-1, the file is called `mydomain.zone.rev`.

(See “Setting Up the hosts.rev File” on page 189 and Code Example 12-22 on page 189.)

The `named.local` *File*

The `named.local` file specifies the address for the local loopback interface, or `localhost`, with the network address 127.0.0.1. The name of this file is specified in the boot file. Like other files, you can give it a name other than the name used in this book. (See “Setting Up the named.local File” on page 189 and Code Example 12-23 on page 190.)

12

Setting Up the Boot File

The contents of the boot file varies, depending on the type of server. This section describes boot files for primary, secondary, and caching-only servers.

The server's initialization script, `/etc/init.d/inetsvc`, expects the name `/etc/named.boot` when it looks for the `in.named` daemon boot file. The script will not start the daemon if you name the boot file something else.

Code Example 12-1 shows a sample boot file for a primary server.

Code Example 12-1 Master Boot File for Primary Server

```
;
; Sample named.boot file for Primary Master Name Server
;
; type domain                                  source file or host
;
directory /var/named
cache      .                                     named.ca
primary             university.edu.              mydomain.zone
primary             32.128.in-addr.arpa.      mydomain.zone.rev
primary            0.0.127.in-addr.arpa          named.local
```

The entries in the file are explained below.

The `directory` *Line*

The `directory` line in the boot file designates the directory in which you want the name server to run:

```
directory     /var/named
```

This allows the use of relative path names for the files mentioned in the boot file or, later, with the `$INCLUDE` directive. This feature is especially useful if you have many files to be maintained, and you want to locate them all in one directory dedicated to that purpose.

If there is no `directory` line in the boot file, all file names listed in the boot file must be full path names.

The cache *Line*

A name server needs to know which servers are the authoritative name servers for the root zone. To do this, you have to list the addresses of these higher authorities.

All servers should have the following line in the boot file to find the root name servers:

```
cache       .          named.ca
```

The first field(`.`) indicates that the server will obtain root servers hints from the indicated file, in this case, `named.ca` (located in the directory `/var/named`).

The primary *Lines*

To set up a primary server, you must create a file that contains all the authoritative data for the zone. Then create a boot file that designates the server as a primary server and tells it where to find the authoritative data.

The following line in the boot file names the server and the data file:

```
primary      university.edu.        mydomain.zone
```

The first field designates the server as primary for the zone `university.edu`, as stated in the second field. The third field contains the name of the file from which authoritative data is read, in this case `mydomain.zone`.

The lines:

```
primary     32.128.in-addr.arpa.        mydomain.zone.rev
primary    0.0.127.in-addr.arpa.        named.local
```

indicate that the server is also a primary server for the zone `32.128.in-addr.arpa.` (that is, the reverse address domain for `university.edu`) and `0.0.127.in-addr.arpa.` (reverse address for the local host loopback). Data for them is to be found, respectively, in the files `mydomain.zone.rev` and `named.local`.

Code Example 12-2 is a sample boot file for a secondary server in the same domain as the above primary server.

Code Example 12-2 Sample Master Boot File for Secondary Server

```
;
; Sample named.boot file for secondary master name server
;
; type         domain                                    source file or host
;
directory /var/named
cache .named . ca
secondary university.edu.    128.32.0.4 128.32.0.10 123.32.136.22 mydomain.zone.zone
secondary 32.128.in-addr.arpa 128.32.0.4 128.32.0.10 128.32.136.22 purev.zone
primary   0.0.127.in-addr.arpa named.local
```

In appearance, this file is very similar to the boot file for the primary server; the main difference is to be found in the lines:

```
secondary  university.edu.      128.32.0.4  128.32.0.10  128.32.136.22  mydomain.zone.zone
secondary  32.128.in-addr.arpa   128.32.0.4   128.32.0.10   128.32.136.22   purev.zone
```

The word `secondary` establishes that this is a secondary server for the zone listed in the second field. It is to get its data from the listed servers (usually the primary server followed by one or more secondaries). Attempts to obtain data are made in the order in which the servers are listed. If there is a file name after the list of servers, as in the example above, data for the zone will be put into that file as a backup. When the secondary server is started, data is loaded from the backup file, if it exists. Then one of the servers is consulted to check whether the data is still up to date.

This ability to specify multiple secondary IP addresses allows for great flexibility in backing up a zone.

Note – A server may 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.

The interpretation of the other `secondary` line is similar to the above. Note also that although this machine is a secondary server for the domain `university.edu.` and `32.128.in-addr.arpa.`, it is a primary server for `0.0.127.in-addr.arpa.` (the local host). Code Example 12-3 is a sample boot file for a caching-only server.

Code Example 12-3 Sample Master Boot File for Caching-only Server

```
;
; Sample named.boot file for caching-only name server
;
; type     domain                              source file or host
;
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 caching-only server. What denotes a caching-only server is the absence of authority lines, such as `secondary` or `primary`, in the boot file. As explained on page 165, a caching-only server does not maintain any authoritative data; it simply handles queries and asks the hosts listed in the `in.named` file for the information needed.

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. This section describes the DNS data files and the resource records each file should contain. The section “Standard Resource Record Format” discusses standard resource record format, including an explanation of each resource record relevant to the DNS data files. It is followed by descriptions of the DNS data files.

Standard Resource Record Format

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

<name>	*<ttl>*	*class*	*record-type*	*record-specific-data*

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

The name Field

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

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

The ttl Field

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

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

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

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

The class Field

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

The record-type Field

The fourth field states the resource record *type*. There are many types of RR's; the most commonly used types are discussed in "Resource Record Types" on page 179.

The record-specific-data Field

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

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

Special Resource Record Characters

The following characters have special meanings:

Table 12-1 Special Resource Record Characters

Character	Definition
.	A free-standing dot in the name field refers to the current domain.
@	A free-standing @ in the name field denotes the current origin.
..	Two free-standing dots represent the null domain name of the root when used in the name field.
\X	Where *X* is any character other than a digit (0-9), quotes that character so that its special meaning does not apply. For example, you can use \. to place a dot character in a label.
\DDD	Where each *D* is a digit, this is the octet corresponding to the decimal number described by *DDD*. The resulting octet is assumed to be text and is not checked for special meaning.

Table 12-1 Special Resource Record Characters

Character	Definition
`()`	Use parentheses to group data that crosses a line. In effect, line terminations are not recognized within parentheses.
`;`	A semicolon starts a comment; the remainder of the line is ignored.
`*`	An asterisk signifies a wildcard.

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

Control Entries

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

`$INCLUDE`

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

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

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

$ORIGIN

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

Resource Record Types

The most commonly used types of resource records are listed in Table 12-2. They are usually entered in the order shown in Table 12-2, but that is not a requirement.

Table 12-2 Commonly Used Resource Record Types

Type	Description
SOA	start of authority
NS	name server
A	Internet address (name to address)
PTR	pointer (address to name)
CNAME	canonical name (nickname)
TXT	text information
WKS	well-known services
HINFO	host information
MX	mail exchanger

Code Example 12-5 on page 181 shows an example of a hosts file. It is presented here for illustration purposes only. Explanations of each field follow the code example. In that sample file @ indicates the current zone or origin and lines that begin with a semicolon (;) are comments.

Setting Up the `named.ca` *File*

The `named.ca` file contains the names and addresses of the root servers. 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. Code Example 12-4 is a sample `named.ca` file.

Code Example 12-4 Sample `named.ca` File

```
;
;Initial cache data for root domain servers.
;
; list of servers...
            99999999       IN          NS          NIC.DDN.MIL.
            99999999       IN          NS          A.ISI.EDU.
            99999999       IN          NS          TERP.UMD.EDU.
            99999999       IN          NS          C.NYSER.NET.
;
; ...and their addresses
NIC.DDN.MIL.    99999999    IN          A           26.0.0.73
C.NYSER.NET.    99999999    IN          A           192.33.4.12
NS.NASA.GOV.    99999999    IN          A           128.102.16.10
A.ISI.EDU.      99999999    IN          A           26.3.0.103
```

Setting Up the `hosts` *File*

The `hosts` file contains all the data about every machine in your zone. This information includes server names, IP addresses, host information (hardware and operating system information), canonical names and aliases, the services supported by a particular protocol at a specific address, and group and user information related to mail services. This information is represented in the records NS, A, CNAME, HINFO, WKS, PRT, and MX records. The file also includes the SOA record, which indicates the start of a zone and includes the name of the host on which the `hosts` data file resides.

Code Example 12-5 shows a sample `hosts` file.

Code Example 12-5 Sample `hosts` File

```
; sample hosts file
@                       IN          SOA     ourlima.Sample.Edu. root.sendai.Sample.Edu.
                                            101 ; Serial
                                            10800 ; Refresh
                                            1800 ; Retry
                                            3600000 ; Expire
                                            86400 ) ; Minimum
                        IN          NS      ourarpa.Sample.Edu.
                        IN          NS      ourlima.Sample.Edu.
ourarpa                 IN          A       128.32.0.4

                        IN          A       10.0.0.78
;                                               The HINFO field is a sample for syntax only
;                       IN          HINFO   3B2 UNIX
arpa                    IN          CNAME   ourarpa
seattle                 IN          A       128.32.0.6
;                       IN          HINFO   Sun 4/75 Solaris 2.5
ourseattle              IN          CNAME   seattle
sendai                  IN          A       128.32.7
                        IN          A       128.32.130.6
;                       IN          HINFO   Sun-4/75 Solaris 2.5
oursendai               IN          CNAME   sendai
ourlima                 IN          A       10.2.0.78
                        IN          A       128.32.0.10
;                       IN          HINFO   Sun-4/75 Solaris 2.5
                        IN          WKS     128.32.0.10 UDP syslog route timed domain
                        IN          WKS     128.32.0.10 TCP ( echo telnet
                                                discard rpc sftp
                                                uucp-path systat daytime
                                                netstat qotd nntp
                                                link chargen ftp
                                                auth time whois mtp
                                                pop rje finger smtp
                                                supdup hostnames
                                                domain
                                                nameserver )
lima                    IN          CNAME   ourlima
nairobi                 IN          A       128.32.131.119
;                       IN          HINFO   SPARC 2000 Solaris 2.5
nairobi                 IN          MX      0 sendai.Sample.Edu.
```

SOA— Start of Authority

Code Example 12-6 shows the syntax of a start-of-authority (SOA) resource record.

Code Example 12-6 SOA Record Format

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

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

name

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

class

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

SOA

This field is the type of this resource record.

origin

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

person-in-charge

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

serial

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

refresh

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

retry

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

expire

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

minimum

This field is the default number of seconds to be used for the time-to-live field on resource records that don't have a *ttl* specified.

There should only be one SOA record per zone. Code Example 12-7 is a sample SOA resource record.

Code Example 12-7 Sample SOA Resource Record

```
;name       class SOA     origin                          person-in-charge
Sample.edu.  IN          SOA      ourlima.Sample.Edu.     root.sendai.Sample.Edu.
                          101         ;Serial
                          7200        ;Refresh
                          3600        ;Retry
                          432000      ;Expire
                          86400)      ;Minimum
```

NS—Name Server

Code Example 12-8 shows the syntax of a name-server (NS) resource record:

Code Example 12-8 NS Record Format

```
<name>   <ttl>          class          NS          name-server-name
```

The name-server record lists by name a server responsible for a given domain. The *name* field lists the domain that is serviced by the listed name server. If no *name* field is listed, then it defaults to the last name listed. One NS record should exist for each primary and secondary master server for the domain. Code Example 12-9 is a sample NS resource record.

Code Example 12-9 Sample NS Resource Record

```
;<name>     <ttl>       class       NS          name-server-name
                        IN          NS          ourarpa.Sample.Edu.
```

A—Address

Code Example 12-10 shows the syntax of an address (A) resource record:

Code Example 12-10 Address Record Format

```
<name>       <ttl>          class          A          address
```

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

Code Example 12-11 Sample Address Record

```
;<name>    <ttl>  class      A          address
ourarpa               IN      A          128.32.0.4
                      IN      A          10.0.0.78
```

HINFO—Host Information

Code Example 12-12 shows the syntax of a host-information (HINFO) resource record:

Code Example 12-12 HINFO Record Format

```
<name>      <ttl>       class       HINFO       hardware        OS
```

The host-information resource record contains host-specific data. It lists the hardware and operating system that are running at the listed host. If you want to include a space in the machine name or in the entry in the *hardware* field, you must surround the entry with quotes. The *name* field specifies the name of the host. If no name is specified, it defaults to the last `in.named` host. One HINFO record should exist for each host. Code Example 12-13 is a sample HINFO resource record.

Code Example 12-13 Sample HINFO Resource Record

```
;<name>  <ttl>   class  HINFO   hardware      OS
                 IN     HINFO   Sun-4/80      UNIX
```

Caution – The HINFO field is considered a security hole and is seldom used.

WKS—Well-Known Services

Code Example 12-14 shows the syntax of a well-known services (WKS) resource record:

Code Example 12-14 WKS Record Format

<name>	*<ttl>*	*class*	*WKS*	*address*	*protocol-list-of-services*

The well-known services record describes the well-known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in the `services` database. Only one WKS record should exist per protocol per address.
Code Example 12-15 is an example of a WKS resource record.

Code Example 12-15 Sample WKS Resource Record

```
;<name>     <ttl>       class       WKS         address         protocol-list-of-services
                        IN          WKS         128.32.0.10     UDPwho route timed domain
                        IN          WKS         128.32.0.10     TCP (echo telnet
                                                                    discard rpc sftp
                                                                    uucp-path systat daytime
                                                                    netstat qotd nntp
                                                                    link chargen ftp
                                                                    auth time whots mtp
                                                                    pop rje finger smtp
                                                                    supdup hostnames
                                                                    domain
                                                                    nameserver)
```

The WKS record is optional. Most sites no longer provide this information.

CNAME—Canonical Name

Code Example 12-16 shows the syntax of a canonical-name (CNAME) resource record.

Code Example 12-16 CNAME Record Format

nickname	*<ttl>*	*class*	*CNAME*	*canonical-name*

The canonical-name resource record specifies a nickname for a canonical name. A nickname should be unique. All other resource records should be associated with the canonical name and not with the nickname. Do not create a nickname and then use it in other resource records. Nicknames are particularly useful during a transition period, when a machine's name has changed but you want to permit people using the old name to reach the machine. Code Example 12-17 is a sample CNAME resource record.

Code Example 12-17 Sample CNAME Resource Record

```
;nickname      <ttl>      class   CNAME           canonical-name
 oursendai                 IN      CNAME           sendai
```

PTR—Pointer Record

Code Example 12-18 shows the syntax for a pointer (PTR) resource record.

Code Example 12-18 PTR Record Format

```
special-name                <ttl>     class          PTR-real-name
```

A pointer record allows special names to point to some other location in the domain. In the example, PTR's are used mainly in the `in-addr.arpa.` records for the translation of an address (the special name) to a real name. PTR names should be unique to the zone. The PTR records Code Example 12-19 sets up reverse pointers for the special `in-addr.arpa.` domain.

Code Example 12-19 Sample PTR Resource Record

```
;special name                              <ttl>      class   PTR-real-name
  7.0                                                 IN      PTR sendai.university.edu.
  2.2.18.128.in-addr.arpa.                                IN      PTR blah.gull.com.
```

MX—Mail Exchanger

Code Example 12-20 shows the syntax for a mail-exchanger (MX) resource record.

Code Example 12-20 MX Record Format

```
name                <ttl>          class          MX     preference-value    mailer-exchanger
```

The mail-exchanger resource records are used to specify a machine that knows how to deliver mail to a domain or machines in a domain. There may be more than one MX resource record for a given name. In Code Example 12-21 on page 188, `Seismo.CSS.GOV.` (note the fully qualified domain name) is a mail gateway that knows how to deliver mail to `Munnari.OZ.AU`. Other machines on the network cannot deliver mail directly to `Munnari`. `Seismo` and `Munnari` may have a private connection or use a different transport medium. The *preference-value* field indicates the order a mailer should follow when there is more than one way to deliver mail to a single machine. The value 0 (zero) indicates the highest preference. If there is more than one MX resource record for the same name, records may or may not have the same *preference* value.

You can use names with the wildcard asterisk (*) for mail routing with MX records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a relay. In Code Example 12-21, all mail to hosts in domain `foo.com` is routed through `RELAY.CS.NET`. You do this by creating a wildcard resource record, which states that the mail exchanger for `*.foo.com` is `RELAY.CS.NET`. The asterisk will match any host or subdomain of foo.com, but it will not match `foo.com` itself.

Note – If the MX record contains both a wildcard and an explicit resource record, the explicit record is used.

Code Example 12-21 Sample MX Resource Record

```
;name                {ttl}      class      MX     preference-value   mailer-exchanger
Munnari.OZ.AU.                  IN         MX      0                  Seismo.CSS.GOV.
foo.com.                        IN         MX      10                 RELAY.CS.NET.
*.foo.com.                      IN         MX      20                RELAY.CS.NET.
```

Setting Up the `hosts.rev` *File*

The `hosts.rev` file sets up inverse mapping. It must contain the names of the primary and master name servers in your local domain, plus pointers to those servers and to other, non-authoritative name servers. The names of the primary and secondary master servers are indicated by NS records, and the pointers are indicated by PTR records. The file also needs an SOA record to indicate the start of a zone and the name of the host on which `hosts.rev` resides. Code Example 12-22 on page 189 is a sample `hosts.rev` file.

Code Example 12-22 Sample `hosts.rev` File

```
; sample hosts.rev file
@       IN  SOA ourhost.university.edu. root.sendai.university.edu.
              (
                101    ; Serial
                3600           ; Refresh
                300            ; Retry
                3600000 ; Expire
                3600 )         ; Minimum
            IN  NS      ourarpa.university.edu.
            IN  NS      ourhost.university.edu.
4.0         IN  PTR     ourarpa.university.edu.
6.0         IN  PTR     seattle.university.edu.
7.0         IN  PTR     sendai.university.edu
10.0        IN  PTR     ourhost.university.edu.
6.130       IN  PTR     sendai.university.edu.
```

Setting Up the `named.local` *File*

The `named.local` file sets up the local loopback interface for your name server. It must contain the host name of the machine, plus a pointer to the host name `localhost`, which represents the loopback mechanism. The server name is indicated in the NS resource record, and the pointer to `localhost` is indicated by the PTR record. The file must also include an SOA record, which

indicates the start of a zone and includes the name of the host on which the `named.local` data file reside. Code Example 12-23 is a sample `named.local` file.

Code Example 12-23 Sample `named.local` File

```
; sample named.local file
@   IN  SOA ourhost.university.edu. root.sendai.university.edu

            1                   ; Serial
            3600                ; Refresh
            300                 ; Retry
            3600000             ; Expire
            3600 )              ; Minimum
    IN      NS                  ourhost.university.edu.
1   IN      PTR                 localhost.
```

Modifying the Data Files

When you add or delete a host in one of the data files in the master DNS server or otherwise modify the data files, you must also change the serial number in the SOA resource record so the secondary servers modify their data accordingly; you should then inform `in.named` in the master server that it should reread the data files and update its internal database.

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

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

Note that all previously cached data is lost, and the caching process starts over again.

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

A Practical Example

You can now start building the files that an *imaginary* network would need. Assume that the network is composed of three networks, all with access to the Internet. Each network has a class C network number:

Table 12-3 Domain Configuration of Example Network—Class C

Name	Number
gull	223.100.100
falcon	223.100.101
owl	223.100.102

The names of the zones are also the names of the hosts that are being designated as the master servers.

Further assume that after careful consideration you decide that you want to set up DNS in the network so that each master server is the primary server for its zone and a secondary server for the other zones. All these assumptions result in the following tables:

Table 12-4 Domain Configuration of Example Network—gull Zone

Host Name	Function	Address
gull	primary	223.100.100.1
falcon	secondary	223.100.101.1
owl	secondary	223.100.102.1
	hosts	223.100.100.2-80

Table 12-5 Domain Configuration of Example Network—falcon Zone

Host Name	Function	Address
falcon	primary	223.100.101.1
gull	secondary	223.100.100.1
owl	secondary	223.100.102.1
	hosts	223.100.101.2-110

Table 12-6 Domain Configuration of Example Network—`owl` Zone

Host Name	Function	Address
`owl`	primary	`223.100.102.1`
`gull`	secondary	`223.100.100.1`
`falcon`	secondary	`223.100.101.1`
	hosts	`223.100.102.2-156`

Code Example 12-24 shows boot files for the three servers in the network.

Code Example 12-24 Example Network Server Boot Files

```
;
Boot file for server gull
directory       /var/named
cache           .                           named.root
primary         gull.com.                   gull.zone
primary         100.100.223.in-addr.arpa.   gull.revzone
primary         0.0.127.in-addr.arpa.       named.local
secondary       falcon.gull.com.            223.100.101.1 223.100.102.1 falcon.zone
secondary       owl.gull.com.               223.100.101.1 223.100.102.1 owl.zone
secondary       101.100.223.in-addr.arpa.   223.100.101.1 falcon.rev
secondary       102.100.223.in-addr.arpa.   223.100.102.1 owl.rev
```

```
;
; Boot file for server falcon
directory /var/named
cache       .                           named.root
primary     falcon.gull.com.            falcon.zone
primary     101.100.223.in-addr.arpa.   falcon.revzone
primary     0.0.127.in-addr.arpa.       named.local
secondary   gull.com.                   223.100.100.1 223.100.102.1 gull.zone
secondary   owl.gull.com.               223.100.100.1 223.100.102.1 owl.zone
secondary   100.100.223.in-addr.arpa.   223.100.100.1 gull.rev
secondary   102.100.223.in-addr.arpa.   223.100.102.1 owl.rev
```

```
;Boot file for server owl
;
directory      /var/name
cache          .                           named.root
primary        owl.gull.com.               owl.zone
primary        102.100.223.in-addr.arpa.   owl.revzone
primary        0.0.127.in-addr.arpa.       named.local
secondary      gull.com.                   223.100.100.1 223.100.102.1 gull.zone
secondary      falcon.gull.com.            223.100.100.1 223.100.101.1 falcon.zone
secondary      100.100.223.in-addr.arpa.   223.100.100.1 gull.rev
secondary      101.100.223.in-addr.arpa.   223.100.101.1 falcon.rev
```

The following are some sample `resolv.conf` files. Note that if the host in question is not running `in.named`, the local host address should not be used as a name server.

Code Example 12-25 Example `resolve.conf` Files

```
;resolv.conf file for server gull running in.named
;
domain         gull.com.
nameserver     127.0.0.1
nameserver     223.100.101.1
nameserver     223.100.102.1
```

```
; resolv.conf file for host in zone    gull not running in.named
;
domain         gull.com.
nameserver     223.100.100.1
nameserver     223.100.101.1
nameserver     223.100.102.1
```

```
;
; resolv.conf file for a host in zone falcon.gull not running in.named
;
domain         falcon.gull.com
nameserver     223.100.100.1
nameserver     223.100.101.1
nameserver     223.100.102.1
```

```
; resolv.conf file for a host in zone owl.gull not running in.named
;
domain         owl.gull.com.
nameserver     223.100.100.1
nameserver     223.100.101.1
nameserver     223.100.102.1
```

Code Example 12-26 shows sample `named.local` files:

Code Example 12-26 Example `named.local` Files

```
;
; named.local for server gull
;
@          IN  SOA     gull.com.      ralph.sysad.owl.gull.com.
(
               101            ;Serial
               10800          ;Refresh
               3600           ;Retry
               432000         ;Expire
               86400)         ;Minimum
           IN  NS             gull.com.
1          IN  PTR            localhost.
```

```
;
; named.local for server falcon
;
@          IN  SOA     falcon.gull.com.
ralph.sysad.owl.gull.com.(
               101            ;Serial
               10800          ;Refresh
               3600           ;Retry
               432000         ;Expire
               86400)         ;Minimum
           IN  NS             falcon.gull.com.
1          IN  PTR            localhost.
```

```
;
; named.local for server owl
;
@          IN  SOA     owl.gull.com.  ralph.sysad.owl.gull.com.
(
               101            ;Serial
               10800          ;Refresh
               3600           ;Retry
               432000         ;Expire
               86400)         ;Minimum
           IN  NS             owl.gull.com.
1          IN  PTR            localhost.
```

Code Example 12-27 shows the `hosts` file for server `gull`, followed by its `$INCLUDE` file.

Code Example 12-27 Example `hosts` File for Server `gull`

```
;
; gull zone hosts file for server gull
;
@                           IN         SOA     gull.com. ralph.sysad.owl.gull.com.
                                       101                ;Serial
                                       10800              ;Refresh
                                       3600               ;Retry
                                       432000             ;Expire
                                       86400)             ;Minimum
                            IN         NS                 gull.com.
                            IN         NS                 falcon.gull.com.
                            IN         NS                 owl.gull.com.
falcon.gull.com.            IN         NS                 falcon.gull.com.
                            IN         NS                 gull.com.
                            IN         NS                 owl.gull.com.
owl.gull.com.               IN         NS                 owl.gull.com.
                            IN         NS                 gull.com.
                            IN         NS                 falcon.gull.com.
gull.com.                   IN         MX                 10 gull.com.
*.gull.com.                 IN         MX                 10 gull.com.
; gull.com hosts
```

Code Example 12-28 Example `include` File for Server `gull`

```
$INCLUDE /var/named/hosts/gull
; hosts in gull zone as listed in /var/named/hosts/gull
gull                        A          223.100.100.1
                            A          10.1.0.56
                            MX         10                 gull.com.
falcon                      A          223.100.101.1
;                           HINFO      SPARC 10           Solaris 2.5
                            MX         10 gull.com.
                            WKS        223.100.101.1      UDP syslog timed domain
                            WKS        223.100.101.1      TCP (echo telnet
                                                          discard rpc sftp
                                                          uucp-path systat daytime
netstat
                                                          qotd nntp link chargen ftp auth
                                                          time whots mtp pop rje finger
                                                          smtp supdup hostnames
                                                          domain nameserver)
owl                         A          223.100.102.1
;                           HINFO      SPARC 5
                            MX         10 gull.com.
                            WKS        223.100.102.1      UDP syslog timed domain
                            WKS        223.100.102.1      TCP (echo telnet
                                                          discard sftp
                                                          uucp-path systat daytime
netstat
                                                          qotd nntp link chargen ftp auth
                                                          time whots mtp pop rje finger
                                                          smtp supdup hostnames
                                                          domain nameserver)
puma                        A          223.100.100.2
;                           HINFO      SPARC 5             Solaris 2.4
                            MX         10 gull.com.
tiger                       A          223.100.100.3
;                           HINFO      SPARC 20           Solaris 2.3
                            MX         10 gull.com.
lion                        A          223.100.100.4
;                           HINFO      SPARC 1000         Solaris 2.5
                            MX         10 gull.com.
; all other hosts follow, up to 223.100.100.80
```

Code Example 12-29 shows a `hosts` file for server `falcon`, followed by its `$INCLUDE` file.

Code Example 12-29 Example `hosts` and `include` File for Server `Falcon`

```
;
; falcon zone hosts file for server falcon
;
@                           IN          SOA      falcon.gull.com.ralph.sysad.
owl.gull.com.
                                        101              ;Serial
                                        10800            ;Refresh
                                        3600             ;Retry
                                        432000           ;Expire
                                        86400)           ;Minimum
                            IN          NS               falcon.gull.com.
                            IN          NS               gull.com.
                            IN          NS               owl.gull.com.
; gull.com hosts
$INCLUDE /var/named/hosts/falcon
; hosts in falcon zone as listed in /var/named/hosts/falcon
falcon                  A               223.100.101.1
;                       HINFO               SPARC 10          Solaris 2.5
                        MX                  10 gull.com.
machine1                CNAME               falcon.gull.com
machine2                A                   223.100.101.2
;                       HINFO               SPARC 10          Solaris 2.5
                        MX                  10 gull.com.
machine3                A                   223.100.101.3
;                       HINFO               SPARC 10          Solaris 2.5
                        MX                  10 gull.com.
machine4                A                   223.100.101.4
;                       HINFO               SPARC 5           Solaris 2.3
                        WKS                 223.100.101.4     UDP who route
                        WKS                 223.100.101.4     TCP (echo telnet
                                                              discard sftp
                                                              uucp-path systat daytime
netstat
                                                              ftp finger domain nameserver)
                        MX                  10 gull.com.

; all other hosts follow, up to 223.100.101.110
```

Code Example 12-30 is the sample file of reverse addresses for hosts in the zone `gull`. Note that the name of the domain is fully qualified, so that the addresses of the hosts (without the network address) is sufficient in this case:

Code Example 12-30 Example Reverse Address for Server Gull

```
; reverse address file for server gull, in /var/named/gull.revzone
100.100.223.in-addr.arpa.IN                  SOA        gull.com.ralph.sysad. owl.gull.com.
                                             101        ;Serial
                                             10800      ;Refresh
                                             3600       ;Retry
                                             432000     ;Expire
                                             86400)     ;Minimum
                       IN                      NS           gull.com.
1                                            PTR        gull.com.
2                                            PTR        puma.gull.com.
3                                            PTR        tiger.gull.com.
4                                            PTR        lion.gull.com.
; all other hosts follow, up to 223.100.100.80
```

The reverse address files for servers `falcon` and `owl` should be written in a manner similar to the above.

Setting Up a Root Server for a Local Network

If you are not connecting your local network to the Internet, you must set up primary and secondary name servers in the root-level domain on the local network. This is so all domains in the network have a consistent authoritative server to which to refer; otherwise, machines may not be able to resolve queries.

Since a single machine can be the primary domain name server for more than one machine, the easiest way to create a root domain name server is to have a server be the name server for all the domains that make up its own domain name. For example, if a server is named `x.sub.dom.`, then it should be designated the primary name server for ".", `dom.`, and `sub.dom`.

Since the root name server provides an authoritative name server at the root level of the network, all top-level domains should have their name-server records (`IN NS`) defined in the root domain.

Note – The root domain server name should be the primary name server for all top-level domains in the network.

12

Index

Symbols

#, comment character, 45
(), in DNS resource records, 178
*, in DNS resource records, 178, 188
+/- syntax compatibility, name service switch, 146 to 147
., *See* dots (.)
;, in DNS resource records, 178
@, in DNS resource records, 177
DDD, in DNS resource records, 177
X, in DNS resource records, 177

A

`-a` option
  `nisaddent` command, 128, 130, 134
  `nisgrpadm` command, 37 to 38, 81, 86, 119 to 120, 121
  `nispopulate` script, 33
A resource records (DNS)
  described, 179, 184
  `hosts` file, 180 to 181
  `named.ca` file, 180
access rights, *See* rights (NIS+)
adding (NIS+)
  DES credentials to root domain, 85 to 86
  LOCAL credentials to root domain, 84, 85
  non-root master and replica servers to admin group, 119 to 120
  principals to admin group, 23, 37 to 38, 86, 121
  replicas to existing domains, 109 to 111
  root master to admin group, 81 to 82
address resource records, *See* A resource records (DNS)
admin groups (NIS+)
  adding to
    non-root master and replica servers, 119 to 120
    principals, 23, 37 to 38, 86, 121
    root master, 81 to 82
  creating
    for non-root domains, 119
    for root domain, 81
  credential specification for, scripts and, 19, 33, 34
  defined, 26
  listing members, 82, 120
  naming
    for non-root domains, 116
    for root domain, 74
administrative domain (DNS), 154
`aliases` file, NIS+ aliases table and, 31

append option (NIS+)
populating tables with, 124, 128, 130, 134
ASCII files, *See* files
asterisk (*), in DNS resource records, 178, 188
at sign (@), in DNS resource records, 177
authentication (NIS+), 2
authoritative servers, *See* servers (DNS), master servers
authorization, *See also* rights (NIS+)
authorization (NIS+), 2
auto_home table (NIS+)
populating from files, 130
populating from NIS maps, 135
auto_master table (NIS+)
populating from files, 130
populating from NIS maps, 135
`automounter` information (NIS+)
populating tables from files and, 130
populating tables from NIS maps and, 135

B

`-B` option
`nisinit` command, 99 to 100
`rpc.nisd` command, 46, 76, 106, 108
Berkeley Internet Name Domain service, *See* `in.named` daemon
`BIND`, *See* `in.named` daemon
boot file (DNS), 170, 192, 193
broadcast initialization (NIS+), 99 to 100

C

`-c` option
`nisgrpadm` command, 81, 119
`nisinit` command, 99, 101 to 102
`nisping` command, 131, 136
cache (NIS+)
See also `nscd` daemon
client set up commands and, 95
root domain set up commands and, 72 to 73
root master server directory cache, listing contents, 78
starting, 83, 107
`cache` line (DNS boot file), 173
caching servers (DNS), 165
caching-only servers (DNS), 165, 175
“can’t list table: Server busy, Try Again.” message (`nisping` command), 38 to 39, 131, 136
canonical-name resource records, *See* CNAME resource records (DNS)
case sensitivity
domain names (NIS+), 32, 55
name server database (DNS), 177
changing
data files (DNS), 190
directory rights (NIS+), 78, 92, 119
domain names (NIS+)
clients, 96 to 98
root master server, 71
network password (NIS+), 121
`nisserver` script information, 28 to 29
root master server information (NIS+), 28 to 29
security levels (NIS+)
root master server, 84
scripts and, 26
table rights (NIS+), 139 to 140
checkpointing tables (NIS+), 23, 38 to 39, 131, 136
`chkey` command (NIS+), 86, 121
`class` field (DNS resource records), 176, 182
client (DNS)
overview, 156
client-only model (DNS), 156
clients (DNS), 167 to 168
client-only, 156
client-server, 156
resolver, 167 to 168
setting up, 167 to 168, 193 to 194

clients (NIS+), 39 to 43, 58 to 61, 89 to 103
converting to non-root replicas, 24, 56 to 58
converting to root master server replicas, 19, 23, 24, 44, 46 to 49, 50
converting to servers, 23, 42, 44 to 46, 50
credentials, 91, 93
disk space requirements, 5
DNS compatibility and, 15, 23, 39, 44, 45 to 46
DNS usage by, 145 to 146
domain name assignment, 93, 96 to 98
NIS compatibility and, 23, 26, 29, 44 to 46
`nsswitch.conf` file, 94 to 95
security considerations, 91 to 92, 96
setting up with commands, 89 to 103
initializing, 95, 98 to 102
initializing by broadcast method, 99 to 100
initializing by coldstart file method, 101 to 102
initializing by host name method, 100 to 101
overview, 89 to 90
setting up, 90 to 96
summary of commands, 103
setting up with `nisclient` script
creating additional clients, 42
initializing client machines for non-root domains, 20, 24, 58 to 60
initializing client machines for root domain, 19, 23, 39 to 43
initializing users as non-root domain clients, 20, 24, 60 to 61
initializing users as root domain clients, 19, 23, 42 to 43
limitations, 15
overview, 13, 14, 15
setting up with Solstice System Management Tools, 89
client-server model
NIS+, 2
CNAME resource records (DNS)
described, 179, 186
`hosts` file, 180
coldstart file (NIS+)
broadcast initialization method and, 99
cache manager and, 83
coldstart file initialization method and, 102
described, 77, 78
host name initialization method and, 101
listing contents, 78
location of, 78
coldstart file initialization (NIS+), 101 to 102
COM domain (Internet), 159
commands (NIS+)
See also scripts (NIS+); *specific commands*
client set up, 89 to 103
name service switch set up, 143 to 147
non-root domain set up, 113 to 122
overview, 3
root domain set up, 67 to 88
server set up, 105 to 112
summaries
client set up, 103
non-root domain set up, 122
root domain set up, 87
server set up, 112
table set up, 140 to 141
table set up, 123 to 141
comment lines
in DNS resource records, 178
in `/etc/init.d/rpc` file, 45
compatibility
See also DNS compatibility; NIS compatibility
+/- syntax, 146 to 147
`compatserver.log` file (NIS+), 109
configuration worksheets (NIS+), 7 to 8

control entries (DNS resource records), 178, 196, 197
converting (NIS+)
clients to servers, 23, 42, 44 to 46, 50
servers to non-root master servers, 23, 49, 52
servers to non-root replicas, 24, 56 to 58
servers to root replicas, 19, 23, 44, 46 to 49, 50
create rights, *See* rights (NIS+)
creating
admin groups (NIS+)
for non-root domains, 119
for root domain, 81
clients, *See* clients (DNS); clients (NIS+)
DES credentials (NIS+)
for clients, 91, 93
for root master server, 80 to 81
directories (NIS+)
non-root domain, 116 to 117
root, 74 to 75
subdirectories, 78 to 79, 117 to 118
table, 27
domains, *See* domains (DNS); domains (NIS+)
sample NIS+ namespace, 18, 20 to 24, 62 to 63
servers, *See* servers (DNS); servers (NIS+)
subdomains, *See* domains (DNS); domains (NIS+)
tables (NIS+)
for non-root domains, 117 to 118
for root domain, 78 to 79
`cred.org` table (NIS+), 80
credentials (NIS+)
See also public keys; `publickey` file; `publickey` map; security command for, 19, 80 to 81
DES
adding to root domain, 85 to 86
creating for clients, 91, 93
creating for root master server, 80 to 81
populating tables from files and, 130
populating tables from NIS maps and, 135
LOCAL
adding to root domain, 84, 85
populating tables from files and, 130
populating tables from NIS maps and, 135
other administrators'
adding to non-root domain, 120 to 121
adding to root domain, 85
populating tables from files and, 129
populating tables from NIS maps and, 135
script for, 14, 19, 33, 34
verifying root master credentials, 80 to 81
worksheet, 7

D

`-d` option
`nisaddent` command, 137
`nisclient` script, 40, 60
`nispopulate` script, 33
`nisserver` script, 25, 47, 50, 58
daemons
See also `in.named` daemon (DNS); NIS+ daemon; `rpc.nisd` command
`keyserv` (NIS+), 73, 95
`nscd` (NIS+), 72, 95
`rpc.nisd_resolv`, 106
`/data` directory, *See* `/var/nis/data` directory
data files (DNS), 170, 171, 175 to 188
described, 170, 171
`hosts` file, 171, 180 to 181, 195 to 197
`hosts.rev` file, 171, 189, 198
modifying, 190

named.ca file, 171, 180
named.local file, 171, 189, 190, 195
resource records
control entries, 178, 196, 197
special characters, 177
standard format, 175, 175 to 179
types, 179 to 188
sample, 180, 189, 190, 191 to 198
setting up, 175 to 188, 191 to 198
data.dict file (NIS+), 77, 78
caution, 75
name of, 75
database (NIS+)
checkpointing, 23, 38 to 39, 131, 136
described, 2
dbm files (NIS), populating NIS+ tables and, 132
defaultdomain file (NIS+), 97, 98
defaults
domain name (NIS+), 97
network (Secure-RPC) password (NIS+), 36, 37, 39
nisserver script, 24
non-root domain directory rights (NIS+), 119
time-to-live (DNS), 176, 183
DES credentials (NIS+)
adding to root domain, 85 to 86
creating for clients, 91, 93
creating for root master server, 80 to 81, 82 to 83
populating tables from files and, 130
populating tables from NIS maps and, 135
destroy rights, *See* rights (NIS+)
dictionary file (NIS+), 77, 78
.dir files (NIS), populating NIS+ tables and, 132
directories (NIS+)
See also specific directories
changing rights, 78, 92, 119
creating
non-root domain, 116 to 117
root, 74 to 75
subdirectories, 78 to 79, 117 to 118
table, 27
listing contents, 77
listing object properties, 77
rights worksheet, 7
root directory
creating, 74 to 75
root master server public key and, 82 to 83
verifying, 77
directory line (DNS boot file), 172
disk space requirements (NIS+), 4 to 5, 38 to 39
displaying, *See* listing
DNS, 151 to 199
case sensitivity, 177
clients, 167 to 168
domains, *See* domains (DNS)
enabling NIS+ client to use DNS, 145 to 146
Internet and, 160 to 161
mail delivery, 165 to 166
name resolution, 165
name servers, *See* servers (DNS)
name service switch and, 168
namespace, *See* namespace (DNS)
resolver, 167 to 168
sample configuration, 191 to 198
servers, *See* servers (DNS)
structure, 151
zones, 182
DNS compatibility
NIS+ commands and, 69, 75 to 76, 106, 107 to 108
NIS+ scripts and, 15, 23, 39, 44, 45 to 46, 58
Domain Name System, *See* DNS
domain names (DNS)
data file resource records and, 176, 177, 179, 184
overview, 160
domain names (NIS+)
assigning to clients, 93, 96 to 98

case sensitivity, 32, 55
changing
for clients, 96 to 98
for root master server, 71
checking, 70, 96
default, 97
`nisclient` specification of, 40, 60
`nispopulate` specification of, 33 to 34
`nisserver` specification of, 25, 28, 47, 58
requirements, 6, 25, 30 to 31, 70 to 71
`domainname` command (NIS+), 70, 96
domains (DNS)
defined, 154, 157
Internet and, 160 to 161
root domain, 165, 171, 180, 198 to 199
sample configuration, 191 to 192
subdomains, 156
domains (NIS+)
security considerations
non-root domains, 115
root domain, 24, 26, 69
server requirements and, 4
set up commands for
non-root domains, 113 to 122
root domain, 67 to 88
set up script for
non-root domains, 20, 23, 49 to 52, 58 to 60
root domain, 19, 23, 24 to 29
summary of commands
non-root domains, 19 to 20, 23, 122
root domain, 23, 62, 87
dots (.)
in DNS boot file `cache` line, 173
in DNS resource records, 177, 178, 182
NIS+ names and, 6, 31, 70, 71, 85, 93

E

EDU domain (Internet), 159
`EMULYP=` string (`/etc/init.d/rpc` file)
NIS+ commands and, 76, 108
NIS+ scripts and, 45 to 46, 48
environment variables (NIS+)
`NIS_GROUP`, 74, 116
`PATH`, 23, 25
`TMPDIR`, 32
error messages
"can't list table: Server busy, Try Again." (`nisping` command), 38 to 39, 131, 136
"Not Found" (DNS), 158
`parse error` warnings (`nispopulate` script), 37
"password differs from login password" (NIS+), 80, 85
"password incorrect" (NIS+), 80
"unable to create credential" (NIS+), 80
`/etc` files, populating tables from, *See* files, populating tables from
`/etc/.rootkey` file (NIS+)
clients, 96
root master server, 80
`/etc/defaultdomain` file (NIS+), 97, 98
`/etc/group` file (NIS+), +/- syntax compatibility, 146 to 147
`/etc/hosts` file (NIS+)
client set up and, 92
host name initialization method and, 101
`/etc/init.d/inetsvc` script (DNS), 170, 172
`/etc/init.d/rpc` file (NIS+)
commands and, 76, 108
scripts and, 45 to 46, 48
`/etc/named.boot` file (DNS), 170, 172 to 175
`/etc/nsswitch.conf` file
DNS and, 168
NIS+ commands and
+/- syntax compatibility, 146 to 147
clients and, 94 to 95
enabling NIS+ client to use DNS, 145 to 146

root master server and, 71 to 72
selecting alternate configuration file, 143 to 145
setting up switch, 143 to 147
NIS+ scripts and, 24
overview, 2, 144
/etc/passwd file (NIS+)
+/- syntax compatibility, 146 to 147
LOCAL credentials and, 84
passwd table and, 31
root domain set up and, 69
root master server set up and, 25
/etc/resolv.conf file (DNS), 69, 167 to 168, 193 to 194
/etc/shadow file (NIS+)
+/- syntax compatibility, 146 to 147
login password warning and, 85
table population and, 30, 31
expire field (SOA resource records), 183

F

-F option (nispopulate script), 33
files
See also data files (DNS); *specific files*
coldstart, *See* coldstart file (NIS+)
populating tables from
+/- syntax compatibility, 146 to 147
NIS+ commands for, 125 to 131
NIS+ scripts for, 5 to 7, 14, 20, 21, 23, 30 to 39, 52 to 56
switch-configuration, *See* /etc/nsswitch.conf file

G

GOV domain (Internet), 159
group file (NIS+), +/- syntax compatibility, 146 to 147
groups (NIS+)
See also admin groups (NIS+)
nisserver specification of, 26, 29
rights for, *See* rights (NIS+)
groups_dir directory (NIS+)
non-root domain, 117 to 118
root domain, 74, 78 to 79, 82 to 83
verifying existence of, 118

H

-h option
nisclient script, 40, 60
nispopulate script, 33
nisserver script, 47, 50, 58
-H option (nisinit command), 100 to 101
hardware field (HINFO resource records), 185
hierarchy
DNS namespace, 156 to 163
NIS+ namespace, 2
HINFO resource records (DNS)
described, 179, 185
hosts file, 180
host name initialization (NIS+), 100 to 101
host names (NIS+)
nisclient specification of, 40, 60
nispopulate specification of, 33
requirements, 6, 25, 30 to 31, 70 to 71
host-information resource records, *See* HINFO resource records (DNS)
hosts file (DNS)
described, 171
NIS+ hosts table and, 31
sample, 180, 195 to 197
setting up, 180 to 181
hosts file (NIS+), *See* /etc/hosts file (NIS+)
hosts.rev file (DNS), 171, 189, 198

I

-i option (nisclient script), 39, 40, 58, 60
iin.named daemon (DNS)
overview, 155
IN record class (DNS), 176
in.named daemon (DNS)

caution, 190
inetsvc script (DNS), 172
master servers and, 164
modifying data files and, 190
named.boot file and, 170, 172
named.ca file and, 171
secondary servers and, 164
in-addr.arpa domain (DNS), 171, 187
$INCLUDE command (DNS resource records), 178, 196, 197
inetd (DNS), in.named daemon and, 190
inetsvc script (DNS), 169, 170
/init.d/inetsvc script (DNS), 170, 172
/init.d/rpc file, *See* rpc file (NIS+)
initialization script (DNS), 169
initializing (NIS+)
commands for
clients, 95, 98 to 102
clients by broadcast method, 99 to 100
clients by coldstart file method, 101 to 102
clients by host name method, 100 to 101
scripts for
client machines for non-root domains, 24, 58 to 60
client machines for root domain, 19, 23, 40 to 42
users as non-root domain clients, 20, 24, 60 to 61
users as root domain clients, 19, 23, 42 to 43
INT domain (Internet), 159
Internet
See also IP address
DNS and, 160 to 161
registration, through Internet Service Provider, 160
registration, through InterNIC, 160
Internet Service Provider, 160
Internet-address resource records, *See* A resource records (DNS)
InterNIC, Internet registration services, 160, 168
interrupting
nispopulate script, 35
nisserver script, 27
inverse mapping (DNS), 171, 189, 198
IP address (DNS), 165 to 166, 168, 184
IP address (NIS+)
broadcast initialization method and, 99
host name initialization method and, 100
nisclient specification of, 40 to 41
nispopulate specification of, 33
root master server, 78

K

keylogin command (NIS+)
clients, 96
root master servers, 80
keyserv daemon (NIS+)
clients and, 95
root master server and, 73
keyserver (NIS+)
updating information for clients, 95
updating information for root master server, 80

L

-l option (nisgrpadm command), 82, 120
listing (NIS+)
admin group members, 82, 120
directory contents, 77
directory object properties, 77
root master server directory cache contents, 78
table contents, 79, 118
transaction log contents, 78, 108
LOCAL credentials (NIS+)
adding to root domain, 84, 85

populating tables from files and, 130
populating tables from NIS maps and, 135
local loopback interface (DNS), 171, 189 to 190
local networks (DNS), root server set up, 198 to 199
`localhost` address (DNS), 171, 189
login password (NIS+), 80, 85
logs, *See* transaction logs
loopback interface (DNS), 171, 189 to 190

M

`-m` option
`nisaddent` command, 124 to 125, 128, 134, 135
`nismkdir` command, 116 to 117
`-M` option (`nisserver` script), 50
mail exchange records (DNS), 165 to 166
mail-exchanger resource records, *See* MX resource records (DNS)
mapping (DNS), 152
maps (DNS), *See* zone files (DNS)
maps (NIS), *See* NIS maps
master servers (DNS), *See* servers (DNS), master servers
master servers (NIS+), *See* servers (NIS+), master servers
memory requirements (NIS+), 5
merge option (NIS+)
populating tables with, 124 to 125, 128, 134, 135
messages, *See* error messages
MIL domain (Internet), 159
minimum field (SOA resource records), 183
modify rights, *See* rights (NIS+)
modifying, *See* changing
MX resource records (DNS)
described, 179, 188
`hosts` file, 180

N

`name` field (DNS resource records)
A resource records, 185
described, 176
HINFO resource records, 185
NS resource records, 184
SOA resource records, 182
name resolution (DNS), 152, 165
name servers (DNS), *See* servers (DNS)
Name Service Cache Daemon, *See* `nscd` daemon (NIS+)
name service switch
DNS and, 168
NIS+ commands and
+/- syntax compatibility, 146 to 147
clients and, 94 to 95
enabling NIS+ client to use DNS, 145 to 146
root master server and, 71 to 72
selecting alternate configuration file, 143 to 145
setting up switch, 143 to 147
NIS+ scripts and, 24
overview, 2
`named.boot` file (DNS), 170, 172 to 175
`named.ca` file (DNS), 171, 180
`named.local` file (DNS), 171, 189, 190, 194, 195
names, *See* domain names (DNS); domain names (NIS+); host names (NIS+); naming
name-server resource records, *See* NS resource records (DNS)
namespace (DNS)
client set up, 167 to 168
server set up, 165, 169 to 199
structure, 156 to 163
namespace (NIS+)
defined, 1
planning layout of, 3 to 5, 19
preparing existing namespaces, 5 to 7
sample, 18, 20 to 24, 62 to 63

setting up with commands, 67 to 147
clients, 89 to 103
name service switch, 143 to 147
non-root domains, 113 to 122
root domain, 67 to 88
servers, 105
tables, 123 to 141
setting up with scripts, 13 to 15, 17 to 63
See also nisclient script; nispopulate script; nisserver script
initializing client machines for non-root domains, 20, 24, 58 to 60
initializing client machines for root domain, 19, 23, 39 to 43
initializing users as non-root domain clients, 20, 24, 60 to 61
initializing users as root domain clients, 19, 23, 42 to 43
limitations, 14 to 15
non-root master servers, 20, 23, 49, 52
non-root replica servers, 24, 56 to 58
overview, 2, 13 to 15, 17 to 20
populating non-root master servers, 20, 23, 52, 56
populating root master servers, 19, 23, 30 to 39
root master server, 19, 23, 24 to 29
root replica servers, 19, 23, 44, 46 to 49
sample namespace, 18, 20 to 24, 62 to 63
summary of commands, 22 to 24, 62 to 63
structure, 2
naming
See also domain names (DNS); domain names (NIS+); host names (NIS+)
admin groups (NIS+)
non-root domain, 116
root domain, 74
NET domain (Internet), 159
netgroup table (NIS+)
+/- syntax compatibility and, 147
Network Information Services Plus, *See* NIS+
network password (NIS+), 36, 37, 39, 41, 121
networks, local, *See* local networks (DNS)
nickname resource records, *See* CNAME resource records (DNS)
NIS compatibility
+/- syntax compatibility, 146 to 147
defined, 26
NIS+ commands and
non-root domains, 114, 117 to 118
overview, 68
root domain, 69, 75 to 76, 78 to 79
servers, 69, 106, 107 to 108
NIS+ scripts and, 23, 26, 29, 44 to 46
`/nis` directory, *See* `/var/nis` directory
NIS maps
disk space requirements, 5
populating tables from
+/- syntax compatibility, 146 to 147
NIS+ commands for, 131 to 136
NIS+ scripts for, 5 to 7, 14, 20, 21, 23, 30 to 39, 52 to 56
transferring NIS+ tables into, 136 to 137
NIS+
admin groups, *See* admin groups (NIS+)
cache manager, 72 to 73, 83, 95, 107
case sensitivity, 32, 55
clients, *See* clients (NIS+)
commands, *See* commands (NIS+)
credentials, *See* credentials (NIS+)

daemons, *See* daemons
database, *See* database (NIS+)
disk space requirements, 4 to 5, 38 to 39
DNS compatibility, *See* DNS compatibility
DNS usage by NIS+ clients, 145 to 146
domain names and, *See* domain names (NIS+)
environment variables, 23, 25, 32
groups, *See* groups (NIS+)
host names and, *See* host names (NIS+)
memory requirements, 5
name service switch and, *See* name service switch
namespace, *See* namespace (NIS+)
NIS compatibility, *See* NIS compatibility
overview, 1 to 2
prerequisites, 3
replicas, *See* servers (NIS+), replicas
scripts, *See* scripts (NIS+)
security, *See* security (NIS+)
servers, *See* servers (NIS+)
setting up
See also commands (NIS+); scripts (NIS+)
configuration worksheets, 7 to 8
overview, 2 to 3
planning layout, 3 to 5, 19
preparation for, 3
structure, 2
swap space requirements, 5, 38 to 39, 131
tables, *See* tables (NIS+)
update propagation, 2
NIS+ daemon
See also `rpc.nisd` command
restarting with security level 2, 84
starting with security level 0, 75 to 76
verifying running of, 78, 109
`nis_cachemgr` command, 83, 107
`NIS_COLD_START` file, *See* coldstart file (NIS+)
`NIS_GROUP` environment variable, 74, 116
`NIS_SHARED_DIRCACHE` file, 83
`nisaddcred` command
clients, 93
non-root domain, 120
root domain, 80 to 81, 84, 85
`nisaddent` command, 128 to 129, 133 to 134
`-a` option (append), 124, 128, 130, 134
`-d` option (NIS+ table to ASCII file), 137
`-m` option (merge), 124 to 125, 128, 134, 135
`-r` option (replace), 124, 128, 134
`-t` option (nonstandard table), 130, 135
`-y` option (NIS domain), 134, 135
`-Y` option (NIS map), 135
`niscat` command
credential verification, 80 to 81
object property listing, 77, 79, 81
table entry verification, 38, 129
`nischmod` command, 78, 92, 119, 139
`nisclient` script, 39 to 43, 58 to 61
`-d` option (domain name), 40, 60
`-h` option (host name), 40, 60
`-i` option (initialize), 39, 40, 58, 60
initializing client machines for non-root domains, 24, 58 to 60
initializing client machines for root domain, 19, 23, 40 to 42
initializing users as non-root domain clients, 20, 24, 60 to 61
initializing users as root domain clients, 19, 23, 42 to 43
limitations, 15
overview, 13, 14, 15
prerequisites to running, 39 to 40, 42 to 43, 59, 60 to 61
`-u` option (user), 43, 61
uses for, 19, 20, 23, 24
using, 39 to 43, 58 to 61
`-x` option (run without executing), 14
`nisgrpadm` command

-a option (adding), 37 to 38, 81, 86, 119 to 120, 121
-c option (creating), 81, 119
-l option (listing), 82, 120
uses for, 19, 23
nisinit command, 74 to 75, 98 to 102
-B option (broadcast method), 99 to 100
-C option (coldstart file method), 101 to 102
-c option, 99, 101, 102
-H option (host name method), 100 to 101
-r option (root master), 74 to 75
nislog command, 78, 108
nisls command, 77
nismkdir command
cautions, 111
-m option (non-root domains), 116 to 117
-s option (add replica server), 110, 116 to 117
nisping command, 23, 38 to 39, 111, 131, 136
nisplus password, 36, 37, 39
nispopulate script, 30 to 39, 52 to 56
-a option (IP address), 33
-d option (domain name), 33
-F option (data from files), 33
-h option (machine name), 33
insufficient /tmp space and, 32, 54
interrupting, 35
non-root master servers, 20, 23, 52, 56
overview, 13, 14
-p option (search path), 33
parse error warnings, 37
prerequisites to running, 30 to 32, 53 to 55
rerunning, 35
root master servers, 19, 23, 30 to 39
uses for, 19, 23
using, 30 to 39
-x option (run without executing), 14
-Y option (data from NIS maps), 33
-y option (NIS domain name), 33 to 34
nisserver script, 24 to 29
changing incorrect information, 28 to 29
-d option (domain name), 25, 47, 50, 58
defaults, 24
-h option (client machine), 47, 50, 58
interrupting, 27
limitations, 14 to 15
-M option (non-root master server), 50
name service switch and, 24
non-root master set up, 20, 23 to 24, 49 to 52
non-root replica set up, 24, 56 to 58
overview, 13, 14 to 15
prerequisites to running, 25, 46 to 47, 57
-R option (replica server), 47, 58
-r option (root master server), 25
rerunning, 27
root master set up, 19, 23, 24 to 29
root replica set up, 19, 23, 44, 46 to 49
uses for, 19, 23, 24
using, 24 to 29
-x option (run without executing), 14
nissetup command
described, 27
non-root domain subdirectory and table creation, 117 to 118
root domain subdirectory and table creation, 78 to 79
-Y option (NIS compatibility), 68, 78 to 79, 114, 117
nisshowcache command, 78
nistbladm command, 140
nisupdkeys command, 82
nobody class (NIS+)
See also rights (NIS+)
NIS compatibility and, 114, 118
non-root domain names (NIS+)
requirements, 70 to 71

non-root domains (NIS+)
initializing client machines
commands for, 95, 98 to 102
script for, 24, 58 to 60
initializing client users, 20, 24, 60 to 61
name requirements, 6, 25, 30 to 31, 70 to 71
security considerations, 115
set up commands for, 113 to 122
set up script for, 20, 23, 49 to 52, 58 to 60
summary of commands, 20, 23 to 24, 122
non-root master servers (NIS+)
set up commands for, 44, 46, 105 to 109
set up script for, 20, 23, 49, 52
table populating script, 20, 23, 52, 56
non-root replica servers (NIS+)
adding to admin groups, 119 to 120
adding to existing domains, 109
set up commands for, 105
set up script for, 24, 56 to 58
"Not Found" message (DNS), 158
NS resource records (DNS)
described, 179, 184
`hosts` file, 180 to 181
`hosts.rev` file, 189
`named.ca` file, 180
`named.local` file, 189, 190
`nscd` daemon (NIS+)
clients and, 95
root master server and, 72
`nsswitch.conf` file, *See* `/etc/nsswitch.conf` file

O

`-o` option (`niscat` command), 77, 79, 81
object properties, *See* rights (NIS+)
options (NIS+ commands), *See specific commands*; *specific options*
ORG domain (Internet), 159
`org_dir` directory
non-root domains, 117 to 118
root domain, 74, 78 to 79, 82 to 83
verifying existence of, 118
`$ORIGIN` command (DNS resource records), 179
origin field (SOA resource records), 182
owner rights, *See* rights (NIS+)

P

`-p` option
`nisaddcred` command, 84, 93, 120
`nispopulate` script, 33
`-P` option (`nisaddcred` command), 84, 120, 93
`.pag` files (NIS), populating NIS+ tables and, 132
parentheses, in DNS resource records, 178
`parse error` warnings (`nispopulate` script), 37
passwd column (NIS+)
limiting access to, 137 to 140
`passwd` file, *See* `/etc/passwd` file (NIS+)
passwd table (NIS+)
limiting access to passwd column of, 137 to 140
populating by script, 30, 31
populating from files, 130
"password differs from login password" message (NIS+), 80, 85
"password incorrect" message (NIS+), 80
passwords (NIS+)
See also security
giving new password to keyserver, 80
limiting access to passwd column, 137 to 140
login password, 80, 85
network password, 36, 37, 39, 41, 121
root password, 28, 41, 80
Secure-RPC password, 36, 37, 39, 41
`PATH` environment variable (NIS+), 23, 25
periods (.), *See* dots (.)

permissions, *See* rights (NIS+)
person-in-charge field (SOA resource records), 183
pinging, *See* `nisping` command
planning NIS+ layout, 3 to 5, 19
plus (+) / minus (-) syntax compatibility name service switch, 146 to 147
pointer resource records, *See* PTR resource records (DNS)
populating tables, *See* tables (NIS+), populating
pound sign (#), comment character, 45
preference-value field (MX resource records), 188
`primary` lines (DNS boot file), 173 to 175
primary name servers (DNS), 164, 172 to 175, 198 to 199
private key, of root master server (NIS+), 80
propagating updates (NIS+), 2
properties, *See* rights (NIS+)
`ps` command (NIS+), 78, 109
PTR resource records (DNS)
  described, 179, 187
  `hosts` file, 180
  `hosts.rev` file, 189
  `named.local` file, 189, 190
public keys (NIS+)
  broadcast initialization method and, 99
  host name initialization method and, 100
  root master server, 78, 80, 82 to 83
`publickey` file, populating tables from files and, 129
`publickey` map, populating tables from NIS maps and, 135

R

`-r` option
  `keylogin` command, 80, 96
  `nisaddent` command, 128, 134
  `nisinit` command, 74 to 75
  `nisserver` script, 25
`-R` option (`nisserver` script), 47, 58
read rights, *See* rights (NIS+)
`record-type` field (DNS resource records), 177
refresh field (SOA resource records), 183
replace option (NIS+)
  populating tables with, 124, 128, 134
replicas, *See* servers (NIS+), replicas
`resolv.conf` file (DNS), 69, 167 to 168, 193 to 194
resolver (DNS), 167 to 168
resolving host names, by NIS+ clients requiring DNS, 39
resource records (DNS)
  control entries, 178, 196, 197
  special characters, 177
  standard format, 175, 175 to 179
  types of, 179 to 188
`resource record-specific data` field (DNS resource records), 177
retry field (SOA resource records), 183
rights (NIS+)
  See also security
  changing for directories, 78, 92, 119
  changing for tables, 139 to 140
  listing for directories, 77
  NIS compatibility and, 114, 118
  worksheet, 7 to 8
root directory (NIS+)
  creating, 74 to 75
  root master server public key and, 82 to 83
  verifying, 77
root domain (NIS+)
  security considerations, 24, 26, 69
  set up commands for, 67 to 88
    admin groups, 74, 81
    cache manager, 83
    DES credentials, 80 to 81, 85 to 86
    LOCAL credentials, 84

other administrators' credentials, 85
overview, 67 to 70
passwords, 80
prerequisites, 69 to 70
public keys, 82 to 83
subdirectories, 78 to 79
tables, 78 to 79
set up script for, 19, 23, 24 to 29
summary of commands, 23, 62, 87
root domain clients (NIS+)
set up script for, 19, 23, 39 to 43
root domain name servers (DNS), 153, 165, 171, 180, 198 to 199
root master servers (NIS+)
credentials, 80 to 81, 82 to 83
Internet address, 78
private key, 80
public key, 78, 80, 82 to 83
secret key, 80
set up commands for, 67 to 88
set up script for, 19, 23, 24 to 29
summary of commands, 23, 62, 87
switch-configuration file, 71 to 72
table populating script, 30 to 39
root password (NIS+), 28, 41, 80
root replicas (NIS+)
adding to existing domains, 109
set up commands, 105
set up script, 19, 23, 44, 46 to 49, 50
`root.dir` directory object (NIS+), 77
`root.object` file (NIS+), 74, 77
`.rootkey` file (NIS+)
clients, 96
root master server, 80
`rootmaster` server (NIS+), 22
`rpc` file (NIS+)
commands and, 76, 108
scripts and, 45 to 46, 48
RPC netname (NIS+), 85, 93
RPC password, *See* Secure-RPC password (NIS+)
`rpc.nisd` command, 44 to 46, 76
See also NIS+ daemon
`-B` option (DNS compatibility), 46, 76, 106, 108
function of, 19, 23
prerequisites to running, 44 to 45, 76
`-S` option (security level), 76, 84
swap space requirements and, 5
using, 44 to 46, 76, 84
`-Y` option (NIS compatibility), 45 to 46, 75 to 76, 106, 107 to 108
`rpc.nisd_resolv` daemon, 106

S

`-s` option (`nismkdir` command), 110, 116 to 117
`-S` option (`rpc.nisd` command), 76, 84
scripts (DNS), server initialization script, 169
scripts (NIS+), 13 to 15, 17 to 63
See also nisclient script; nispopulate script; nisserver script
caution, 13
changing incorrect information, 28 to 29
interrupting, 27, 35
limitations, 14 to 15
overview, 2, 13 to 15, 17 to 20
rerunning, 27, 35
running without executing, 14
summary of commands, 22 to 24, 62 to 63
secondary name servers (DNS), 164, 174 to 175, 198
secret key, root master servers (NIS+), 80
secure RPC netname (NIS+), 85, 93
Secure-RPC password (NIS+), 36, 37, 39, 41
security (NIS+)
See also credentials (NIS+); rights (NIS+)
authentication, 2
authorization, 2
changing levels
root master server, 84

scripts and, 26
client set up and, 91 to 92, 96
levels of, 26, 84, 106
non-root domains, 115
overview, 2, 69
passwords
giving new password to keyserver, 80
limiting access to passwd column, 137 to 140
login password, 80, 85
network password, 36, 37, 39, 41, 121
root password, 28, 41, 80
Secure-RPC password, 36, 37, 39, 41
public keys
See also `publickey` file; `publickey` map
broadcast initialization method and, 99
host name initialization method and, 100
root master server, 78, 80, 82 to 83
rights worksheets, 7 to 8
root domain set up and, 24, 26, 69
server set up and, 106
table populating, 125, 132
semicolon (;), in DNS resource records, 178
serial field (SOA resource records), 183, 190
servers (DNS), 164 to 165, 169 to 199
boot file, 170, 172 to 175, 193
caching, 165
caching-only, 165, 175
data files, 170, 171, 175 to 188
described, 170, 171
`hosts` file, 171, 180 to 181, 195 to 197
`hosts.rev` file, 171, 189, 198
modifying, 190
`named.ca` file, 171, 180
`named.local` file, 171, 189, 190, 195
resource record control entries, 178
resource record special characters, 177
resource record standard format, 175, 175 to 179
resource record types, 179 to 188
sample, 180, 189, 190, 198
setting up, 175, 188, 198
inverse mapping and, 171, 189
local loopback interface, 171, 189 to 190
local networks, 198 to 199
master servers
boot files, 172 to 175
defined, 164
primary, 164, 172 to 175, 198 to 199
root domain, 165, 171, 180, 198 to 199
secondary, 164, 174 to 175, 198
overview, 155
setting up, 169 to 170
boot file, 170, 172 to 175
data files, 170, 171, 175, 188
example, 191 to 198
update propagation, 2
zones and, 165
servers (NIS+), 24 to 29, 44 to 58, 67 to 88, 105 to 112
checkpointing and, 38 to 39
converting clients to, 23, 42, 44 to 46, 50
converting to non-root master servers, 23, 49, 52
converting to non-root replicas, 24, 56 to 58
converting to root replicas, 19, 23, 44, 46 to 49, 50
determining requirements, 4
DNS compatibility, 23, 39, 44, 45 to 46
master servers
defined, 2

non-root domain set up commands, 113 to 122
non-root domain set up script, 20, 23, 49 to 52, 58 to 60
non-root master added to admin groups, 119 to 120
non-root master set up commands, 44 to 46, 105 to 109
non-root master set up script, 20, 23, 49 to 52
planning, 3, 4, 7
root domain set up commands, 67 to 88
root domain set up script, 19, 23, 24 to 29
root master added to admin groups, 81 to 82
root master credentials, 80 to 81, 82 to 83
root master Internet address, 78
root master `nsswitch.conf` file, 71 to 72
root master private key, 80
root master public key, 78, 80, 82 to 83
root master secret key, 80
root master summary of commands, 23, 62, 87
table populating script for non-root master servers, 20, 23, 52, 56
table populating script for root master servers, 19, 23, 30 to 39
update propagation from, 2
memory requirements, 5
NIS compatibility, 23, 24, 26, 29, 44 to 46
overview, 2
purposes of, 44
replicas
adding to existing domains, 109 to 111
converting clients to master server replicas, 24
converting servers to root replicas, 23, 46 to 49
defined, 2
domains spanning multiple subnets and, 110
maximum number per domain, 4
non-root replica set up script, 24, 56 to 58
non-root replicas added to admin groups, 119 to 120
planning, 4, 7
root replica set up script, 19, 23, 44, 46 to 49, 50
update propagation to, 2
security considerations, 106
setting up with commands, 67 to 88, 105 to 112
non-root master servers, 23, 44, 46, 105
non-root replicas, 105
root master server, 67 to 88
root replicas, 105 to 111
summary of commands, 112
setting up with `nisserver` script
limitations, 14 to 15
NIS compatibility and, 24, 26
non-root masters, 49 to 52
non-root replicas, 24, 56 to 58
overview, 13, 14 to 15
root master, 19, 23, 24 to 29
root replicas, 19, 23, 44, 46 to 49
security and, 24, 26
tables and, 2
worksheet, 7
`services` database (DNS), 186
set up commands (NIS+), *See* commands (NIS+)
set up scripts (NIS+), *See* scripts (NIS+)
shadow column (NIS+ passwd table), 130
`shadow` file (NIS+), *See* `/etc/shadow` file (NIS+)
shadow table (NIS+), 30, 130

SOA resource records (DNS)
described, 179, 182 to 184
`hosts` file, 180
`hosts.rev` file, 189
modifying data files and, 190
`named.local` file, 189, 190
Solaris 1.x clients
NIS and DNS compatibility and, 69
transferring NIS+ tables into NIS maps, 136 to 137
Solstice System Management Tools
client set up and, 89
table set up and, 123
temporary credential creation and, 85
space requirements (NIS+), 4 to 5, 38 to 39
special characters (DNS resource records), 177
start-of-authority resource records, *See* SOA resource records (DNS)
subdirectories, *See* directories (NIS+)
subdomains (DNS)
See also domains (DNS)
subdomains (NIS+), *See* non-root domains (NIS+)
subnets
replica servers and (NIS+), 110
summaries, *See* commands (NIS+), summaries; scripts (NIS+), summary of commands
SunOS 4.x clients
DNS compatibility and, 44, 45 to 46
swap space requirements (NIS+)
checkpointing tables and, 38 to 39, 131
overview, 5
switch-configuration file, *See* `/etc/nsswitch.conf` file
system files, *See* files

T

`-t` option (`nisaddent` command), 130, 135
tables (NIS+)
See also passwd table (NIS+); shadow table (NIS+)
checkpointing, 23, 38 to 39, 131, 136
creating
for non-root domains, 117 to 118
for root domain, 78 to 79
dictionary file for, 77, 78
directory creation for, 27
listing contents, 79, 118
populating methods, 123 to 124
populating with commands, 123 to 136
See also `nisaddent` command
append option, 124, 128, 130, 134
checkpointing, 131, 136
from files, 125 to 131
merge option, 124 to 125, 128, 134, 135
from NIS maps, 131 to 136
replace option, 124, 128, 134
security considerations, 125, 132
summary of commands, 140 to 141
populating with scripts
checkpointing, 23, 38 to 39
from files, 5 to 7, 14, 19, 21, 23, 30 to 39, 52 to 56
from NIS maps, 5 to 7, 14, 19, 21, 23, 30 to 39, 52 to 56
non-root master server, 20, 23, 52 to 56
root master server, 19, 23, 30 to 39
summary of commands, 19, 23
private, 15
rights changes, 124 to 140
rights worksheet, 8
servers and, 2
summary of commands, 19, 23, 140 to 141
transferring into NIS maps, 136 to 137
verifying, 129
temporary space, *See* `/tmp` directory
text-information resource records (DNS), 179

time-to-live field (DNS resource records), 176, 183
/tmp directory, nispopulate script and, 32, 54
TMPDIR environment variable (NIS+), 32
trans.log file (NIS+), 77, 78, 108, 109
  cautions, 75
  name of, 75
transaction logs (NIS+)
  disk space requirements, 5
  listing contents, 78, 108
  root master server, 77, 78
  servers, 108, 109
transferring NIS+ tables into NIS maps, 136 to 137
ttl field (DNS resource records), 176, 183
TXT resource records (DNS), 179

U

-u option
  nisclient script, 43, 61
  nistbladm command, 140
"unable to create credential" message (NIS+), 80
update propagation (NIS+), 2
users
  initializing NIS+ non-root domain client users, 20, 24, 60 to 61
  initializing NIS+ root domain client users, 19, 23, 42 to 43
/usr/lib/nis/nisshowcache command, 78
/usr/lib/nis/nisupdkeys command, 82

V

/var/nis directory (NIS+)
  cautions, 75
  cleaning out
    for clients, 95
    for root master server, 73
  clients configured as servers and, 45
  disk space requirements, 5
  name of, 75
  NIS_COLD_START file, 78
  NIS_SHARED_DIRCACHE file, 83
  Solaris earlier versions, 75
  trans.log file, 108, 109
/var/nis/data directory (NIS+), 74, 108, 109, 118
  cautions, 6, 75
  name of, 75
/var/nis/*hostname* directory (NIS+), 75
/var/nis/*hostname*.dict file (NIS+), 75
/var/nis/*hostname*.log file (NIS+), 75
/var/yp directory (NIS+), disk space requirements, 5
variables, *See* environment variables
verifying (NIS+)
  groups_dir directory, 118
  NIS+ daemon running, 78, 109
  org_dir directory, 118
  root directory, 77
  root master credentials, 80 to 81
  tables, 129
viewing, *See* listing

W

warning messages, *See* error messages
well-known services resource records, *See* WKS resource records (DNS)
wildcard character (*), in MX resource records, 188
wiz.com domain (NIS+), 22
WKS resource records (DNS)
  described, 179, 186
  hosts file, 180
worksheets (NIS+), 7 to 8
workstations (NIS+), changing domain name for, 96 to 98
world rights, *See* rights (NIS+)
woz.com domain (NIS+), 70

X

`-x` option (NIS+ scripts), 14
`.xfr` file extension (NIS+), 128

Y

`-Y` option
    `nisaddent` command, 135
    `nispopulate` script, 33
    `nissetup` command, 68, 78 to 79, 114, 117
    `rpc.nisd` command, 45 to 46, 75 to 76, 106, 107 to 108
`-y` option
    `nisaddent` command, 134, 135
    `nispopulate` script, 33 to 34
YP compatibility, *See* NIS compatibility
`ypcat` command (NIS+), 147

Z

zone files (DNS), 162
zone transfer, 164
zones (DNS), 162 to 165, 182

Copyright 1995 Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100 USA.

Tous droits réservés.Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie et la décompliation. Aucune partie de ce produit ou de sa documentation associée ne peuvent Être reproduits sous aucune forme, par quelque moyen que ce soit sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il en a.

Des parties de ce produit pourront etre derivees du système UNIX®, licencié par UNIX Systems Laboratories Inc., filiale entierement detenue par Novell, Inc. ainsi que par le système 4.3. de Berkeley, licencié par l'Université de Californie. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fourmisseurs de Sun.

LEGENDE RELATIVE AUX DROITS RESTREINTS : l'utilisation, la duplication ou la divulgation par l'administation americaine sont soumises aux restrictions visées a l'alinéa (c)(1)(ii) de la clause relative aux droits des données techniques et aux logiciels informatiques du DFAR 252.227- 7013 et FAR 52.227-19.

Le produit décrit dans ce manuel peut Être protege par un ou plusieurs brevet(s) americain(s), etranger(s) ou par des demandes en cours d'enregistrement.

MARQUES
Sun, Sun Microsystems, le logo Sun, Solaris, Solstice, AdminTool, AdminSuite sont des marques deposées ou enregistrées par Sun Microsystems, Inc. aux Etats-Unis et dans certains autres pays. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays, et exclusivement licenciée par X / Open Company Ltd. OPEN LOOK est une marque enregistrée de Novell, Inc., PostScript et Display PostScript sont des marques d'Adobe Systems, Inc.

Toutes les marques SPARC sont des marques deposées ou enregitrées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. SPARCcenter, SPARCcluster, SPARCompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARstation, SPARCstorage, SPARCworks, microSPARC, microSPARC II et UltraSPARC sont exclusivement licenciées a Sun Microsystems, Inc. Les produits portant les marques sont basés sur une architecture développée par Sun Microsytems, Inc.

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

Le système X Window est un produit du X Consortium, Inc.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" SANS GARANTIE D'AUCUNE SORTE, NI EXPRESSE NI IMPLICITE, Y COMPRIS, ET SANS QUE CETTE LISTE NE SOIT LIMITATIVE, DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DES PRODUITS A REPONDRE A UNE UTILISATION PARTICULIERE OU LE FAIT QU'ILS NE SOIENT PAS CONTREFAISANTS DE PRODUITS DE TIERS.

CETTE PUBLICATION PEUT CONTENIR DES MENTIONS TECHNIQUES ERRONEES OU DES ERREURS TYPOGRAPHIQUES. DES CHANGEMENTS SONT PERIODIQUEMENT APPORTES AUX INFORMATIONS CONTENUES AUX PRESENTES, CES CHANGEMENTS SERONT INCORPORES AUX NOUVELLES EDITIONS DE LA PUBLICATION. SUN MICROSYSTEMS INC. PEUT REALISER DES AMELIORATIONS ET / OU DES CHANGEMENTS DANS LE(S) PRODUIT(S) ET / OU LE(S) PROGRAMME(S) DECRITS DANS DETTE PUBLICATION A TOUS MOMENTS.