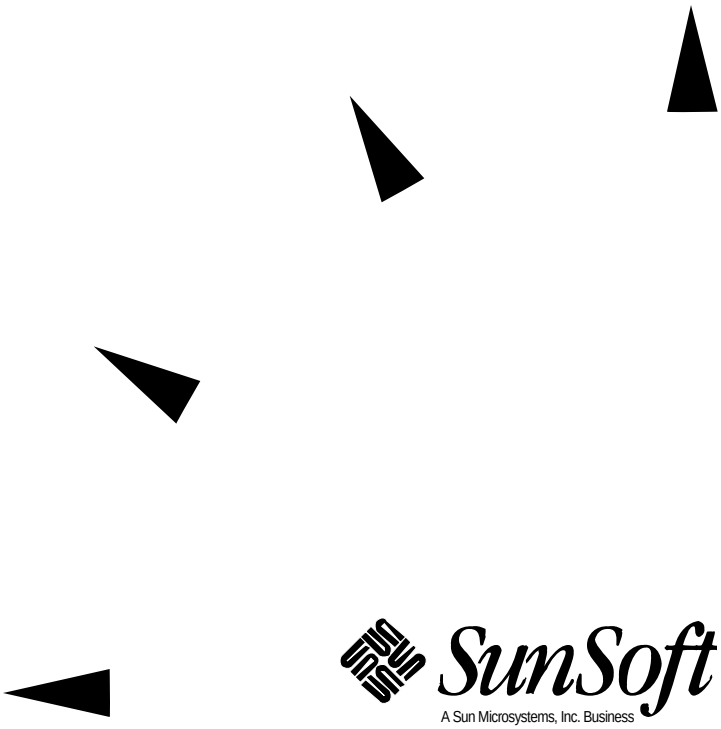
Name Services Configuration Guide



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

SunSoft
A Sun Microsystems, Inc. Business

Contents

Tables

Figures

Preface

The *Name Services Configuration Guide* describes how to configure the Network Information Service Plus (NIS+) and the Domain Name Service (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 manual is for system and network administrators who want to set up a basic network using NIS+ and or DNS. It assumes the reader is an experienced system administrator. For NIS+ customizing information and detailed administration instructions, see *Name Services Administration Guide.* For information on making the transition from NIS to NIS+, see the *NIS+ Transition Guide.*

Although this manual introduces some concepts relevant to NIS+ and DNS, it makes no attempt to explain networking fundamentals or to 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.

How This Book Is Organized

This book is divided into three parts, Part 1—NIS+ Concepts, Part 2—NIS+ Configuration Tutorial, and Part 3—DNS Concepts. Part 1 discusses basic name service and NIS+ concepts. Part 2 contains a tutorial on how to use the NIS+ scripts to configure an NIS+ namespace. Part 3 describes DNS structure and how to configure DNS clients and servers. There is an appendix containing templates for NIS+ planning worksheets, and a glossary.

Chapter 1, "Overview of Name Services," gives a quick description of name services, Network Information Service (NIS), Network Information Service Plus (NIS+) and Domain Name Service (DNS).

Part 1 — NIS+ Concepts

Chapter 2, "Understanding Name Services," describes what name services do.

Chapter 3, "Understanding the NIS+ Namespace," describes the basic structure of an Network Information Service Plus namespace.

Chapter 4, "Understanding NIS+ Tables and Information," describes the structure and contents of the NIS+ tables.

Chapter 5, "Understanding the Name Service Switch," describes the software and files that determine which information sources the name services use.

Part 2 — NIS+ Configuration Tutorial

Chapter 6, "Getting Started with NIS+," describes the NIS+ scripts and the minimum requirements of an NIS+ namespace.

Chapter 7, "Setting Up NIS+," takes you step-by-step through the configuring of an NIS+ namespace using the NIS+ scripts.

Part 3 — DNS Concepts

Chapter 8, "DNS Structure,"describes the structure of the Domain Name Service.

Chapter 9, "Setting Up DNS Clients," describes how to configure a DNS client.

Chapter 10, "Setting Up DNS Servers," describes how to configure a DNS server.

Appendices

Appendix A, "Pre-Setup Worksheets," contains blank worksheets that you can use to determine your domain and server requirements.

"Glossary" is a list of words and phrases found in this book and their definitions.

Related Books

You can consult the following for more information on NIS+ and DNS. These books are also part of the Solaris 2.4 System and Network Administration manual set:

- *Name Services Administration Guide*—Describes how to administer a running NIS+ namespace, modify its security level, and otherwise customize your namespace.
- *NIS+ Transition Guide*—Describes how to make the transition from NIS to NIS+.
- *Network Interfaces Programmer's Guide*—Describes the application programming interfaces for networks including NIS+.
- *man Pages(1M): System Administration Commands*—Contains the man pages for the NIS+ commands.
- *Administration Application Reference Manual*—Describes the administration tool window interface for modifying the data in NIS+ tables.
- *TCP/IP Network Administration Guide*—Describes the TCP/IP network protocol and how to use the TCP/IP software.

Additional books not part of the Solaris 2.4 manual set:

- *DNS and Bind* by Cricket Liu and Paul Albitz, O'reilly & Associates, Inc., 1992.
- *Managing NFS and NIS* by Hal Stern, O'reilly & Associates, Inc., 1991.

What Typographic Changes and Symbols Mean

Table P-1 describes the type changes and symbols 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. `system% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`system% 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.
Code samples are included in boxes and may display the following:		
`%`	UNIX C shell prompt	`system%`
`$`	UNIX Bourne and Korn shell prompt	`system$`
`#`	Superuser prompt, all shells	`system#`

In the AnswerBook® on-line documentation tool, double-clicking on a cross reference takes you to the page on which it begins.

Overview of Name Services 1

This chapter briefly discusses what name services are, the Domain Name Service (DNS), the Network Information Service (NIS), and the Network Information Service Plus (NIS+). Part 1 of this book discusses all these topics in detail.

Directions for setting up a DNS namespace are in Part 3. Directions for setting up an NIS+ namespace are in Chapter 7, “Setting Up NIS+.” See “Glossary” for definitions of terms and acronyms you don’t recognize.

What Is a Name Service?

Name services store information that users, workstations, and applications must have to communicate across the network. Without a name service, each workstation would have to maintain its own copy of this information. This information includes machine addresses, user names, passwords and network

access permissions, in some cases. The information may be stored in files or database tables. Centrally locating this data makes it easier to administer large networks.

What Is DNS?

DNS, the Domain Name Service, is the name service provided by the Internet for TCP/IP networks. It was developed so that workstations on the network could be identified with common names instead of Internet addresses. Domain Name Service performs naming between hosts *within* your local administrative domain and *across* domain boundaries.

The collection of networked workstations that use DNS are referred to as the DNS *namespace*. The DNS namespace can be divided into a hierarchy of *domains*. A DNS domain is simply a group of workstations. Each domain is supported by two or more *name servers*: a principal server and one or more secondary servers. Each server implements DNS by running a daemon called `in.named`. On the client's side, DNS is implemented through the "resolver." The resolver's function is to resolve users' queries; to do that, it queries a name server, which then returns either the requested information or a referral to another server.

What Is NIS?

NIS, Network Information Service, was developed independently of DNS and had a slightly different focus. Whereas DNS focused on making communication simpler by using workstation names instead of addresses, NIS focused on making network administration more manageable by providing centralized control over a variety of network information. As a result, NIS stores information not only about workstation names and addresses, but also about users, the network itself, and network services. This collection of network *information* is referred to as the NIS *namespace*.

NIS uses a client-server arrangement similar to DNS. Replicated NIS servers provide services to NIS clients. The principal servers are called *master* servers, and for reliability, they have backup, or *replica* servers. Both master and replica servers use the NIS information retrieval software and both store NIS maps.

What Is NIS+?

NIS+ (pronounced en-eye-ess plus) is a network name service similar to the network information service (NIS) but with more features. NIS+ is not an extension of NIS, but 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 a location where all workstations on a network can have access to it. This configuration of network information is referred to as the NIS+ *namespace*.

The NIS+ namespace is hierarchical, and is similar in structure to the UNIX® directory file system. The hierarchical structure allows an NIS+ namespace to be configured to conform to the logical hierarchy of an organization. The namespace's layout of information is unrelated to its *physical* arrangement. Thus, an NIS+ namespace can be divided into multiple domains that can be administered autonomously. Clients 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. Various security levels, including none at all, can be set.

Solaris clients use the name service switch (`/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.

1

What NIS + Can Do for You

NIS+ has some major advantages over NIS:

- Secure data access
- Hierarchical and decentralized network administration
- Very large namespace administration
- Access to resources across domains
- Incremental updates

With the security system described in “What Is NIS+?,” you can control a particular user’s access to an individual entry in a particular table. This approach to security helps to keep the system secure and administration tasks to be more broadly distributed without risking damage to the entire NIS+ namespace or even to an entire table.

The NIS+ hierarchical structure allows for multiple domains in one namespace. Division into domains makes administration easier to manage. Individual domains can be administered completely independently, thereby relieving the burden on system administrators who would otherwise each be responsible for very large namespaces. As mentioned above, the security system in combination with decentralized network administration allows for a sharing of administrative work load.

Even though domains may be administered independently, all clients can be granted permission to access information across all domains in a namespace. Since a client can only see the tables in its own domain, the client can only have access to tables in other domains by explicitly addressing them.

Incremental updates mean faster updates of information in the namespace. Since domains are administered independently, changes to master server tables only have to be propagated to that master’s replicas and not to the entire namespace. Once propagated, these updates are visible to the entire namespace immediately.

How NIS+ Differs from NIS

NIS+ differs from NIS in several ways. It has many new features and the terminology for similar concepts is different. Look in "Glossary" if you see a term you don't recognize. Table 1-1 gives an overview of the major differences between NIS and NIS+.

Table 1-1 Differences Between NIS and NIS+

NIS	NIS+
Flat domains—no hierarchy	Hierarchical layout—data stored in different levels in the namespace
Data stored in 2 column maps	Data stored in multi-column tables
Uses no authentication	Uses DES authentication
Single choice of network information source	Name service switch—lets client choose information source: NIS, NIS+, DNS, or local `/etc` files
Updates delayed for batch propagation	Incremental updates propagated immediately

See Chapter 2, "Understanding Name Services" and *NIS+ Transition Guide* for more information on these changes and new features.

1

Part 1 — NIS+ Concepts

This part of the manual focuses on the structure of NIS+. It has four chapters.

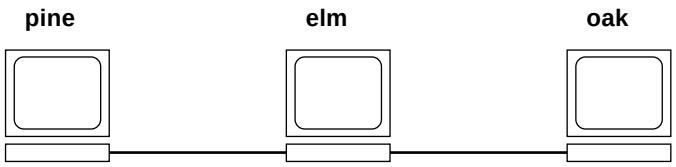
Understanding Name Services 2

This chapter describes the purpose of name services (also called “network information services”), points out their major features and benefits, and compares three of them: DNS, NIS, and NIS+.

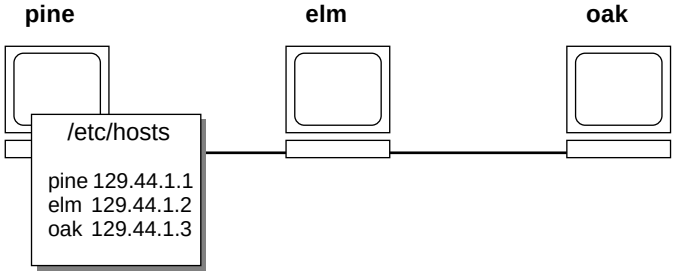
Purpose of Name Services

Name services store information that users, workstations, and applications must have to communicate across the network. Without a name service, each workstation would have to maintain its own copy of this information.

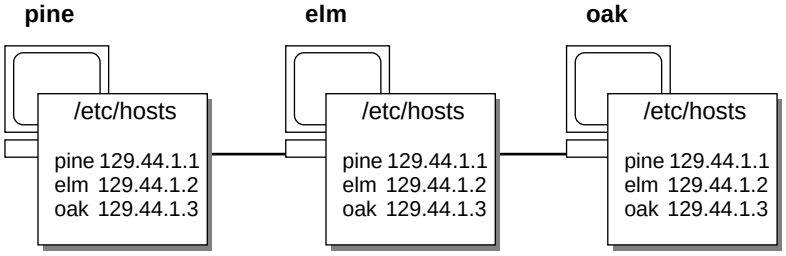
For example, take a simple network of three workstations, `pine`, `elm`, and `oak`:



Before `pine` can send a message to either `elm` or `oak`, it must know their network addresses. For this reason, it keeps a file, `/etc/hosts`, that stores the network address of every workstation in the network, including itself.

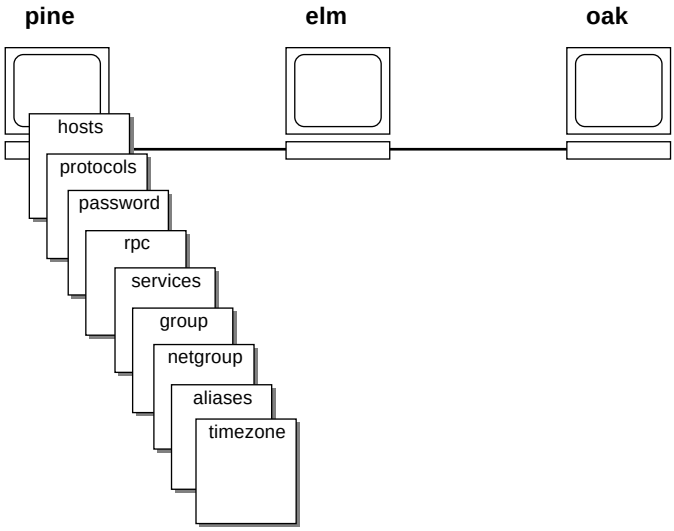


Likewise, in order for `elm` and `oak` to communicate with `pine` or with each other, they must keep similar files.



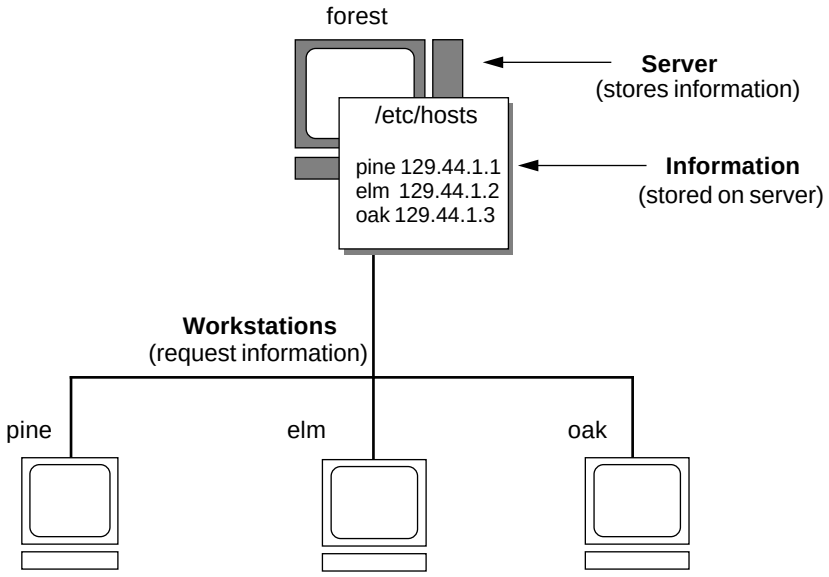
Addresses are not the only network information that workstations need to store. They also need to store security information, mail information, information about their Ethernet interfaces, about network services, about groups of users allowed to use the network, about services offered on the

network, and so on. As networks offer more services, the list grows. As a result, each workstation may need to keep an entire set of files similar to `/etc/hosts`:



As this information changes, administrators must keep it current on every workstation in the network. In a small network this is simply tedious, but on a medium or large network, the job becomes not only time-consuming, but nearly unmanageable.

A network information service solves this problem. It stores network information on servers and provides it to any workstation that asks for it:



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

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

Although the chief purpose of a network information service is to centralize information, another is to simplify network names. A network information service enables workstations to be identified by common names instead of numerical addresses. (This is why these services are sometimes called "name services.") This makes communication simpler because users don't have to remember and try to enter cumbersome numerical addresses like "129.44.3.1." Instead, they can use descriptive names like Sales, Lab1, or Arnold.

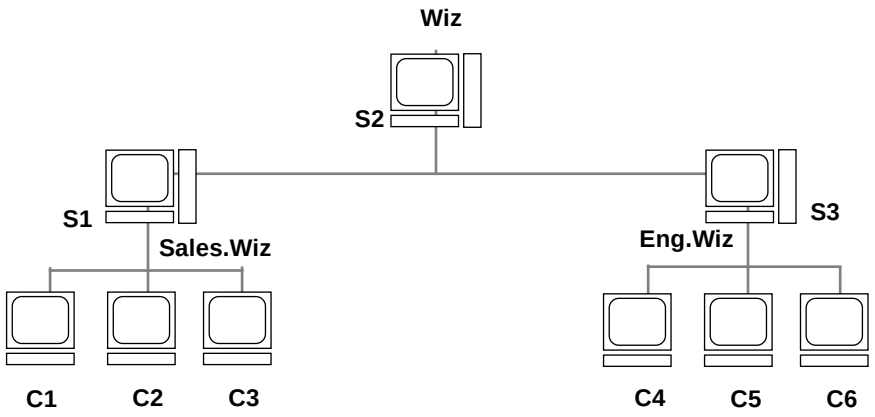
For example, assume that a fictitious company called Wizard, Inc. has set up a network and connected it to the Internet. The Internet has assigned Wizard, Inc. the network number of 129.44.0.0. Wizard, Inc. has two divisions, Sales and Eng, so its network is divided into two subnets, one for each division. Each subnet has its own address:

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

So, instead of addressing mail or other network communications to 129.44.1.0, they could be addressed simply to Wiz. Instead of addressing them to 129.44.2.0 or 129.44.3.0, they could be addressed to Sales.Wiz or Eng.Wiz.

Names are also more flexible than physical addresses. While physical networks tend to remain stable, the organizations that use them tend to change. A network information service can act as a buffer between an organization and its physical network. This is because a network information service is mapped to the physical network, not hard-wired to it. For example,

assume that the Wiz network is supported by three servers, S1, S2, and S3, and that two of those servers, S1 and S3, support clients:

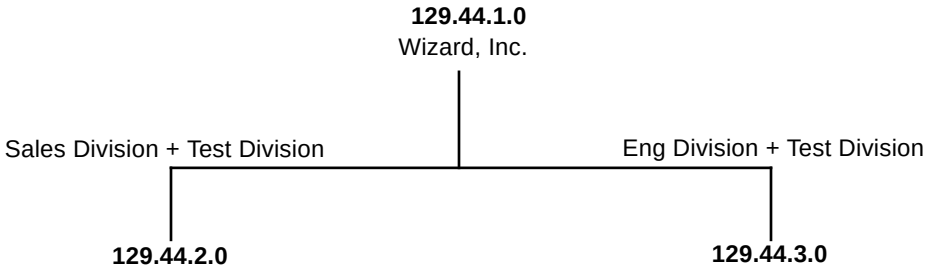


Clients C1, C2, and C3 would obtain their network information from server S1. Clients C4, C5, and C6 would obtain it from server S3. The resulting network is summarized in Table 2-1. (The table is a generalized representation of that network but does not resemble an actual network information map.)

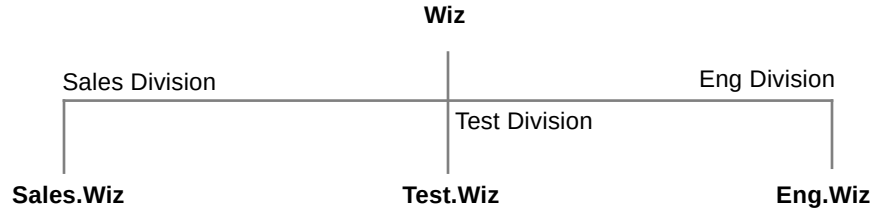
Table 2-1 Representation of Wiz Network

Network Address	Network Name	Server	Clients
129.44.1.0	Wiz	S1	
129.44.2.0	Sales.Wiz	S2	C1, C2, C3
129.44.3.0	Eng.Wiz	S3	C4, C5, C6

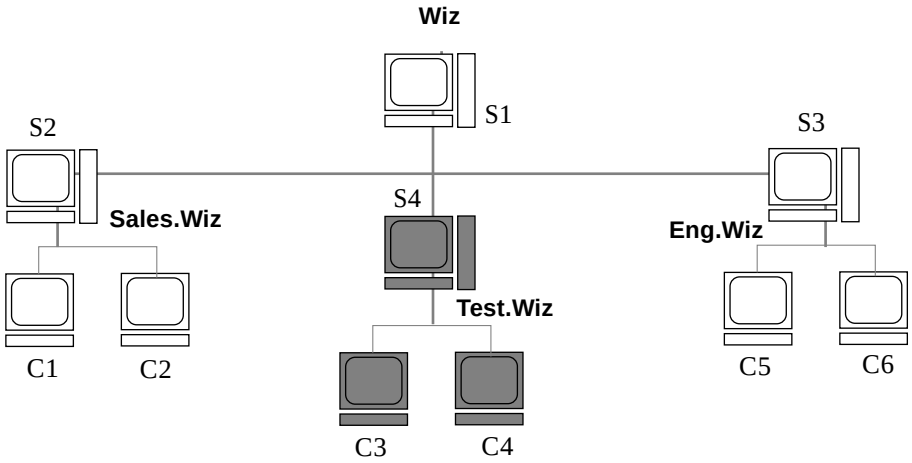
Now assume that Wizard, Inc. created a third division, Testing, which borrowed some resources from the other two divisions, but did not create a third subnet. The physical network would then no longer parallel the corporate structure:



Traffic for the Test Division would not have its own subnet, but would instead be split between 129.44.2.0 and 129.44.3.0. However, with a network information service, the Test Division traffic could have its own dedicated network:



Thus, when an organization changes, its network information service can simply change its mapping:



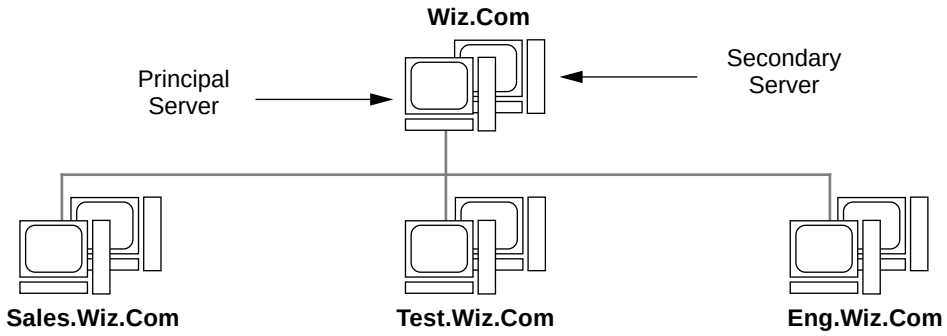
Now clients C1 and C2 would obtain their information from server S2; C3 and C4 from server S4; and C5 and C6 from server S3.

Subsequent changes in the Wizard Inc., organization would continue to be accommodated by changes to the “soft” network information structure without reorganizing the “hard” network structure.

Overview of DNS

DNS, the Domain Naming Service, is the name service provided by the Internet for TCP/IP networks. It was developed so that workstations on the network could be identified with common names instead of Internet addresses.

The collection of networked workstations that use DNS are referred to as the DNS *namespace*. The DNS namespace can be divided into a hierarchy of *domains*. A DNS domain is simply a group of workstations. Each domain is supported by two or more *name servers*: a principal server and one or more secondary servers:



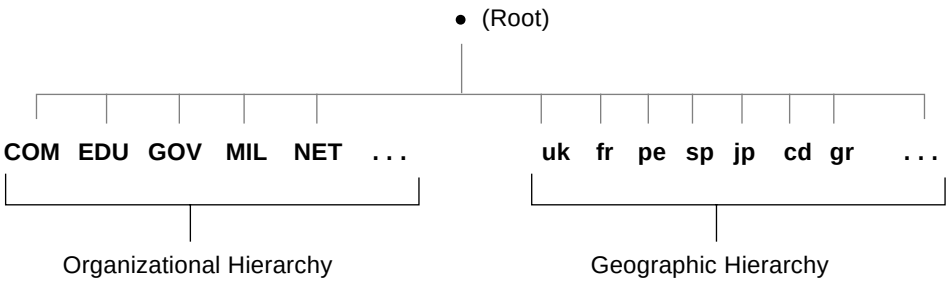
Both principal and secondary servers run the DNS software and store the names and addresses of the workstations in the domain. Principal servers store the original information and secondary servers store copies.

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" message to the client. If the domain is valid, the server routes the request down to the server that supports that domain.

2

DNS and the Internet

DNS is the network information service used by the Internet. The Internet is a vast network that connects many smaller networks across the world. Organizations with networks of any size can join the Internet by applying for membership in two domain hierarchies: an organizational one and a geographical one.



The Organizational hierarchy divides its namespace into the top-level domains listed in Table 2-2.

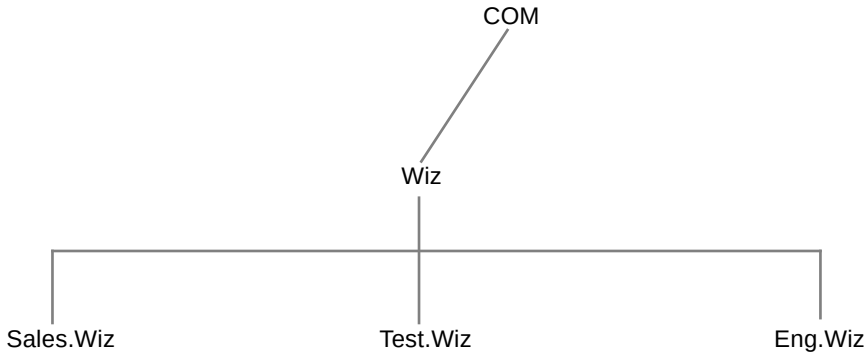
Table 2-2 Internet Organizational Domains

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.

A site using DNS can use any top-level names it prefers, but if it wants to connect to the Internet, it cannot use any of the organizational or geographic names reserved by the Internet's top-level domains.

Networks that join the Internet append their Internet domain name to their own names. For example, if the Wiz domain from the previous example joined the Internet, it would be placed in the "COM" domain.



Thus the full Internet names of the Wiz domains would be:

```
Wiz.COM
Sales.Wiz.COM
Test.Wiz.COM
Eng.Wiz.COM
```

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

```
boss.Wiz.COM
neverhome.Sales.Wiz.COM
quota.Sales.Wiz.COM
lab.Test.Wiz.COM
worknights.Eng.Wiz.COM
```

The Internet regulates administration of its domains by granting each domain authority over the names of its workstations, and expecting each domain to delegate authority to the levels below it. Thus, the COM domain has authority over the names of the workstations 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 workstations in its domain and approves the formation of the Sales.Wiz.COM, Test.Wiz.COM, and Eng.Wiz.COM domains.

2

DNS Name Resolution and Mail Delivery

DNS provides two principal services: it translates hostnames to internet protocol (IP) addresses (and also addresses to names) and it helps mail agents deliver mail along the Internet.

The process of translating names to addresses (and addresses to names) is called *name resolution*. To accomplish this, DNS stores the names and IP addresses of all the workstations in each domain in a set of maps, called *zone files.* One type of zone file stores IP addresses by name. When someone attempts a remote procedure such as `ftp` or `telnet`, it provides the name of the remote workstation. DNS looks up the name in the zone file and converts (or *resolves*) it into its IP address. The IP address is sent along with the remote procedure so the receiving workstation can know who sent the request. This enables the receiving workstation to reply without having to be a DNS client.

Another type of zone file stores workstation names by IP address. It uses them to convert IP addresses to workstation names, a process called *reverse resolution*. Reverse resolution is used primarily to verify the identity of the workstation that sent a message or to authorize remote operations on a local workstation (remote operations are usually authorized per IP addresses, which are more stable than workstation names).

To deliver mail across the Internet, DNS uses *mail exchange records.* Many organizations don't allow mail that comes across the Internet to be delivered directly to workstations within the organization. Instead, they use a central mailhost (or a set of mailhosts) to intercept incoming mail messages and route them to their recipients.

The purpose of a mail exchange record is to identify the mailhost that services each workstation. Therefore, a mail exchange record lists the DNS domain names of remote organizations and either the IP address or the name of its corresponding mailhost. For example:

DNS Domain	**Mailhost**
International.Com.	129.44.1.1
Sales.Wiz.Com.	SalesWizMailer
Eng.Wiz.Com.	EngWizMailer
Fab.Com.	FabMailer

When the mail agent receives a request to send mail to another domain, it parses the name of the recipient backwards and looks for a match in the table. For example, if it receives a request to send mail to neverhome.Sales.Wiz.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 Wiz.Com; since there is none, it continues looking. The next entry it looks for is Sales.Wiz.Com. As you can see in the table above, the mailhost for that domain is SalesWizMailer. Because that is a workstation name, the mail agent asks DNS to resolve it. When DNS provides that mailhost's IP address, the mail agent sends the message.

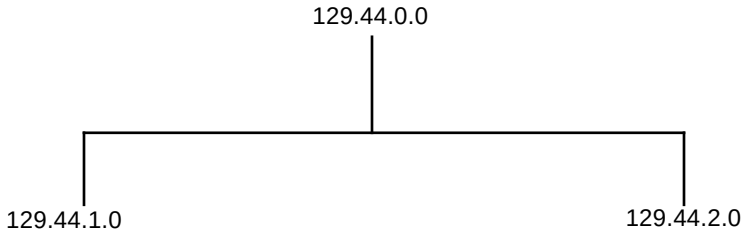
If, instead of the mailhost name, the mail exchange record had specified an IP address, the mail agent would have sent the message directly to that address, since it would have needed no name resolution from DNS.

Overview of NIS

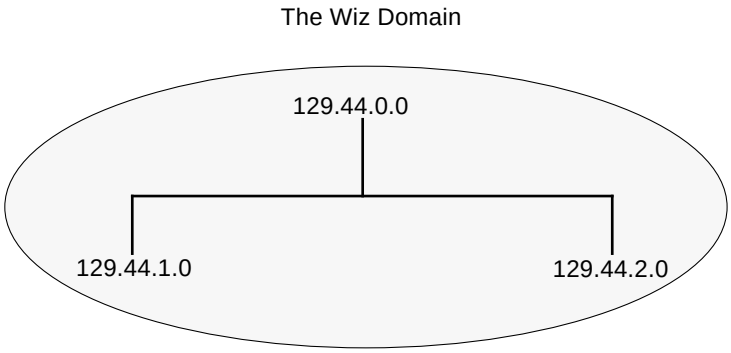
NIS was developed independently of DNS and had a slightly different focus. Whereas DNS focused on making communication simpler by using workstation names instead of addresses, NIS focused on making network administration more manageable by providing centralized control over a variety of network information. As a result, NIS stores information not only about workstation names and addresses, but also about users, the network itself, and network services. This collection of network *information* is referred to as the NIS *namespace*.

NIS uses a client-server arrangement similar to DNS. Replicated NIS servers provide services to NIS clients. The principal servers are called *master* servers, and for reliability, they have backup, or *replica* servers. Both master and replica servers use the NIS information retrieval software and both store NIS maps.

NIS, like DNS, uses domains to arrange the workstations, users, and networks in its namespace. However, it does not use a domain hierarchy; an NIS namespace is flat. Thus, this physical network:



would be arranged into one NIS domain:



An NIS domain can't be connected directly to the Internet. However, organizations that want to use NIS and be connected to the Internet can combine NIS with DNS. They use NIS to manage all local information and DNS for hostname resolution. NIS provides special client routines for this purpose ("DNS forwarding"). When a client needs access to any type of information except IP addresses, the request goes to the client's NIS server. When a client needs name resolution, the request goes to the DNS server. From the DNS server, the client has access to the Internet in the usual way.

NIS Maps

Like DNS, NIS stores information in a set of files called maps, instead of zones. However, NIS maps were designed to replace UNIX `/etc` files, as well as other configuration files, so they store much more than names and addresses. As a result, the NIS namespace has a large set of maps, as shown in Table 2-3 on page 23.

NIS maps are essentially bi-column tables. One column is the key and the other column is information about the key. NIS finds information for a client by searching through the keys. Thus, some information is stored in several maps because each map uses a different key. For example, the names and addresses of workstations are stored in two maps: `hosts.byname` and `hosts.byaddr`. When a server has a workstation's name and needs to find its address, it looks in the `hosts.byname` map. When it has the address and needs to find the name, it looks in the `hosts.byaddr` map.

Table 2-3 NIS Maps

NIS Map	Description
`bootparams`	Lists the names of the diskless clients and the location of the files they need during booting
`ethers.byaddr`	Lists the Ethernet addresses of workstations and their corresponding names
`ethers.byname`	Lists the names of workstations and their corresponding Ethernet addresses
`group.bygid`	Provides membership information about groups, using the group id as the key
`group.byname`	Provides membership information about groups, using the group name as the key
`hosts.byaddr`	Lists the names and addresses of workstations, using the address as the key
`hosts.byname`	Lists the names and addresses of workstations, using the name as the key
`mail.aliases`	Lists the mail aliases in the namespace and all the workstations that belong to them
`mail.byaddr`	Lists the mail aliases in the namespace, using the address as the key

Table 2-3 NIS Maps *(Continued)*

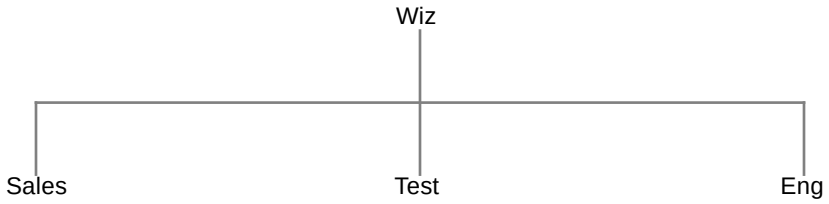
NIS Map	Description
`netgroup`	Contains netgroup information, using group name as the key
`netgroup.byhost`	Contains information about the netgroups in the namespace, using workstation names as the key
`netgroup.byuser`	Contains netgroup information, using user as the key
`netid.byname`	Contains the secure RPC netname of workstations and users, along with their UIDs and GIDs
`netmasks.byaddr`	Contains network masks used with IP subnetting, using address as the key
`networks.byaddr`	Contains the names and addresses of the networks in the namespace, and their Internet addresses
`networks.byname`	Contains the names and addresses of the networks in the namespace, using the names as the key
`passwd.byname`	Contains password information, with username as the key
`passwd.byuid`	Contains password information, with userid as the key
`protocols.byname`	Lists the network protocols used
`protocols.bynumber`	Lists the network protocols used, but uses their number as the key
`publickey.byname`	Contains public and secret keys for secure RPC
`rpc.bynumber`	Lists the known program name and number of RPC's
`services.byname`	Lists the available Internet services
`ypservers`	Lists the NIS servers in the namespace, along with their IP addresses

Overview of NIS+

NIS+ was designed to replace NIS. NIS addresses the administration requirements of client-server computing networks prevalent in the 1980s. At that time client-server networks did not usually have more than a few hundred clients and a few multipurpose servers. They were spread across only a few remote sites, and since users were sophisticated and trusted, they did not require security.

However, client-server networks have grown tremendously since the mid-1980's. They now range from 100-10,000 multi-vendor clients supported by 10-100 specialized servers located in sites throughout the world, and they are connected to several "untrusted" public networks. In addition, the information they store changes much more rapidly than it did during the time of NIS. The size and complexity of these networks required new, autonomous administration practices. NIS+ was designed to address these requirements.

The NIS namespace, being flat, centralizes administration. Because networks in the 90's require scalability and decentralized administration, the NIS+ namespace was designed with hierarchical domains, like those of DNS:



This design enables NIS+ to be used in a range of networks, from small to very large. It also allows the NIS+ service to adapt to the growth of an organization. For example, if a corporation divided itself into two divisions, its NIS+ namespace could be divided into two domains that could be administered autonomously. Just as the Internet delegates administration of domains *downward*, NIS+ domains can be administered more or less independently of each other.

Although NIS+ uses a domain hierarchy similar to that of DNS, an NIS+ domain is much more than a DNS domain. A DNS domain only stores name and address information about its clients. An NIS+ domain, on the other hand, is a collection of *information* about the workstations, users, and network services in a portion of an organization.

Although this division into domains makes administration more autonomous and growth easier to manage, it does not make information harder to access. Clients have the same access to information in other domains as they would have had under one umbrella domain. A domain can even be administered from within another domain.

The NIS+ client-server arrangement is similar those of NIS and DNS in that each domain is supported by a set of servers. The principal server is called the *master* server, and the backup servers are called *replicas*. Both master and

replica servers run NIS+ server software and both maintain copies of NIS+ tables. Tables store information in NIS+ the way maps store information in NIS. The principal server stores the original tables, and the backup servers store copies.

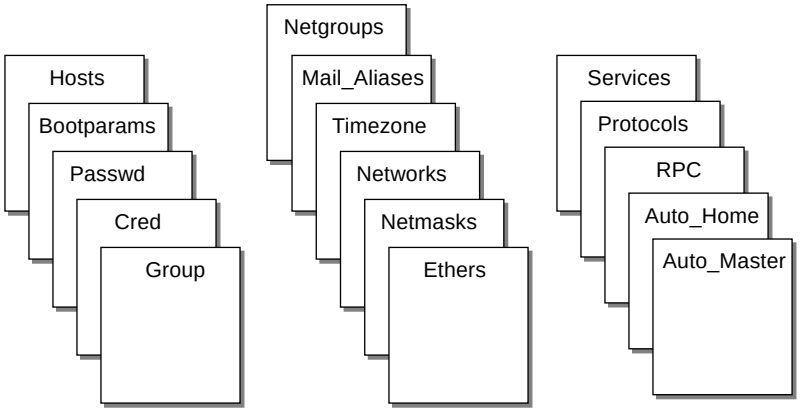
However, NIS+ uses an updating model that is completely different from the one used by NIS. Since at the time NIS was developed, the type of information it would store changed infrequently, NIS was developed with an update model that focused on stability. Its updates are handled manually and, in large organizations, can take more than a day to propagate to all the replicas. Part of the reason for this is the need to remake and propagate an entire map every time any information in the map changes.

NIS+, however, accepts *incremental* updates to the replicas. Changes must still be made on the master server, but once made they are automatically propagated to the replica servers and immediately made available to the entire namespace. You don't have to "make" any maps or wait for propagation.

Details about NIS+ domain structure, servers, and clients, are provided in Chapter 3, "Understanding the NIS+ Namespace".

An NIS+ domain can be connected to the Internet via its NIS+ clients, using the name service switch, described below. The client, if it is also a DNS client, can set up its switch configuration file to search for information in either DNS zone files or NIS maps — in addition to NIS+ tables.

NIS+ stores information in *tables* instead of maps or zone files. NIS+ provides 16 types of predefined, or *system*, tables:



Each table stores a different type of information. For instance, the Hosts table stores information about workstation addresses, while the Password table stores information about users of the network.

NIS+ tables provide two major improvements over the maps used by NIS. First, an NIS+ table can be searched by any column, not just the first column (sometimes referred to as the "key"). This eliminates the need for duplicate maps, such as the `hosts.byname` and `hosts.byaddr` maps used by NIS. Second, the information in NIS+ tables can be accessed and manipulated at three levels of granularity: the table level, the entry level, and the column level. NIS+ tables — and the information stored in them — are described in Chapter 4, "Understanding NIS+ Tables and Information".

NIS+ Security

NIS+ protects the structure of the namespace, and the information it stores, by the complementary processes of *authorization* and *authentication*. First, every component in the namespace specifies the type of operation it will accept and from whom. This is authorization. Second, NIS+ attempts to *authenticate* every request for access to the namespace. Once it identifies the originator of the request, it can find out whether the component has authorized that particular operation for that particular individual. Based on its authentication and the component's authorization, NIS+ carries out or denies the request for access. A full description of this process is provided in *Name Services Administration Guide*.

NIS+ and the Name Service Switch

NIS+ works in conjunction with a separate program called the *name service switch*. The name service switch, sometimes referred to as "the switch," enables Solaris 2.x-based workstations to obtain their information from more than one name service; specifically, from local or `/etc` files, from NIS maps, from DNS zone files, or from NIS+ tables. The switch not only offers a choice of sources, but allows a workstation to specify different sources for different *types* of information. A complete description of the switch software and its associated files is provided in Chapter 5, "Understanding the Name Service Switch".

2

NIS+ and Solaris 1.x

Although NIS+ is provided with the Solaris 2.4 package, it can be used by workstations running the Solaris 1.x software in two different ways:

- NIS-compatibility mode
- Solaris 1.x Distribution

NIS-Compatibility Mode

NIS+ provides an *NIS-compatibility mode*. The NIS-compatibility mode enables an NIS+ server running Solaris 2.4 to answer requests from NIS clients while continuing to answer requests from NIS+ clients. NIS+ does this by providing two service interfaces. One responds to NIS+ client requests, while the other responds to NIS client requests.

This mode does not require any additional setup or changes to NIS clients. In fact, NIS clients are not even aware that the server that is responding isn't an NIS server — except for some differences including: the NIS+ server running in NIS-compatibility mode does not support the `ypupdate` and `ypxfr` protocols and thus it cannot be used as a replica or master NIS server. For more information on NIS-compatibility mode see *NIS+ Transition Guide*.

Note – In Solaris 2.3 and later releases, the NIS-compatibility mode *supports* DNS forwarding. In Solaris 2.2, support for DNS forwarding is available as a *patch*. The DNS forwarding patch is *not* available in Solaris 2.0 and 2.1 releases.

Two more differences need to be pointed out. One is that instructions for setting up a server in NIS-compatibility mode are slightly different than those used to set up a standard NIS+ server. For details, see Chapter 7, "Setting Up NIS+". The other is that NIS-compatibility mode has security implications for tables in the NIS+ namespace. Since the NIS client software does not have the capability to provide the credentials that NIS+ servers expect from NIS+ clients, all their requests end up classified as *unauthenticated*. Therefore, to allow NIS clients to access information in NIS+ tables, those tables must provide access rights to unauthenticated requests. This is handled automatically by the utilities used to set up a server in NIS-compatibility mode, as described in Part II. However, to understand more about the authentication process and NIS-compatibility mode, read *NIS+ Transition Guide* and the chapter on security in *Name Services Administration Guide*.

Solaris 1.x Distribution

NIS+ provides a separate package called the *Solaris 1.x Distribution*, which enables workstations running Solaris 1.x to operate as NIS+ servers without having to upgrade to Solaris 2.x. This use of NIS+ is not recommended, however. It is better to upgrade to Solaris 2.x before making the transition to NIS+. See *NIS+ Transition Guide* for more information on how to make the transition from NIS to NIS+.

The NIS+ Solaris 1.x Distribution consists of the NIS+ daemon, all the NIS+ commands, the NIS+ client libraries, and a README file. It is delivered in a `tar` file, `NISPLUS.TAR`, included in the Solaris 2.4 CD-ROM. To transfer the distribution from the CD-ROM to a Solaris 1.x-based workstation, first mount the CD-ROM, then transfer the `NISPLUS.TAR` file using the `tar` command. If you have network access to the Solaris 1.x Distribution, you can `ftp` or `rcp` it. Instructions for installing it are provided in the README file.

NIS+ Administration Commands

NIS+ provides a full set of commands for administering a namespace. They are described in *Name Services Administration Guide.* Table 2-4, below, summarizes them.

Table 2-4 NIS+ Namespace Administration Commands

Command	Description
`nisaddcred`	Creates credentials for NIS+ principals and stores them in the Cred table.
`nisaddent`	Adds information from `/etc` files or NIS maps into NIS+ tables.
`nis_cachemgr`	Starts the NIS+ Cache Manager on an NIS+ client.
`niscat`	Displays the contents of NIS+ tables.
`nischgrp`	Changes the group owner of an NIS+ object.
`nischmod`	Changes an object's access rights.
`nischown`	Changes the owner of an NIS+ object.
`nischttl`	Changes an NIS+ object's time-to-live value.

Table 2-4 NIS+ Namespace Administration Commands *(Continued)*

Command	Description
`nisdefaults`	Lists an NIS+ object's default values: domain name, group name, workstation name, NIS+ principal name, access rights, directory search path, and time-to-live.
`nisgrep`	Searches for entries in an NIS+ table.
`nisgrpadm`	Creates or destroys an NIS+ group, or displays a list of its members. Also adds members to a group, removes them, or tests them for membership in the group.
`nisinit`	Initializes an NIS+ client or server.
`nisln`	Creates a symbolic link between two NIS+ objects.
`nisls`	Lists the contents of an NIS+ directory.
`nismatch`	Searches for entries in an NIS+ table.
`nismkdir`	Creates an NIS+ directory and specifies its master and replica servers.
`nispasswd`	Changes password information stored in the NIS+ Passwd table.
`nisrm`	Removes NIS+ objects (except directories) from the namespace.
`nisrmdir`	Removes NIS+ directories and replicas from the namespace.
`nissetup`	Creates `org_dir` and `groups_dir` directories and a complete set of (unpopulated) NIS+ tables for an NIS+ domain.
`nisshowcache`	Lists the contents of the NIS+ shared cache maintained by the NIS+ Cache Manager.
`nistbladm`	Creates or deletes NIS+ tables, and adds, modifies or deletes entries in an NIS+ table.
`nisupdkeys`	Updates the public keys stored in an NIS+ object.

NIS+ API

The NIS+ application programmer's interface (API) is a group of functions that can be called by an application to access and modify NIS+ objects. The NIS+ API has 54 functions that fall into nine categories:

- Object manipulation functions (`nis_names`)
- Table access functions (`nis_tables`)
- Local name functions (`nis_local_names`)
- Group manipulation functions (`nis_groups`)
- Application subroutine functions (`nis_subr`)
- Miscellaneous functions (`nis_misc`)
- Database access functions (`nis_db`)
- Error message display functions (`nis_error`)
- Transaction log functions (`nis_admin`)

The functions in each category are summarized in the *Network Interfaces Programmer's Guide*. The category names match the names by which they are grouped in the NIS+ man pages.

What Next?

The remainder of this book shifts focus from name services in general toward NIS+ and DNS in particular. NIS is no longer mentioned except when discussing how to use it with NIS+. Before attempting to set up NIS+, be sure you understand the information presented in the remaining chapters of Part I:

- Chapter 3, "Understanding the NIS+ Namespace," describes NIS+ directories, domains, servers, clients, and the NIS-compatibility mode.
- Chapter 4, "Understanding NIS+ Tables and Information," describes NIS+ tables and the information in each of the 16 system tables.
- Chapter 5, "Understanding the Name Service Switch," describes the switch.

2

Understanding the NIS+ Namespace 3

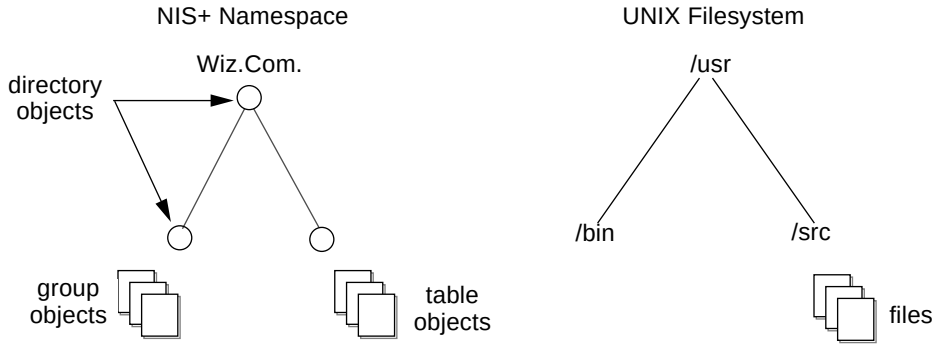
The NIS+ service is designed to conform to the shape of the organization that installs it, wrapping itself around the bulges and corners of almost any network configuration. This is implemented through the NIS+ *namespace.* This chapter describes the structure of the NIS+ namespace, the servers that support it, and the clients that use it. It has the following sections:

Structure of the NIS+ Namespace

The NIS+ namespace is the arrangement of information stored by NIS+. The namespace can be arranged in a variety of ways to suit the needs of an organization. For example, if an organization had three divisions, its NIS+ namespace would likely be divided into three parts, one for each division. Each part would store information about the users, workstations, and network

services in its division, but the parts could easily communicate with each other. Such an arrangement would make information easier for the users to access and for the administrators to maintain.

Although the arrangement of an NIS+ namespace can vary from site to site, all sites use the same structural components: directories, tables, and groups. These components are called NIS+ *objects.* NIS+ objects can be arranged into a hierarchy that resembles a UNIX file system. For example, the illustration below shows, on the left, a namespace that consists of three directory objects, three group objects, and three table objects; on the right it shows a UNIX file system that consists of three directories and three files:

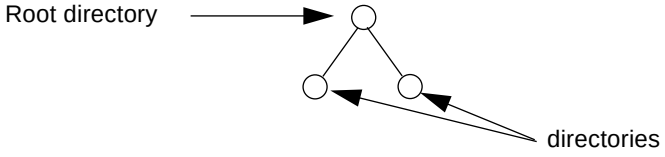


Although an NIS+ namespace resembles a UNIX file system, it has four important differences:

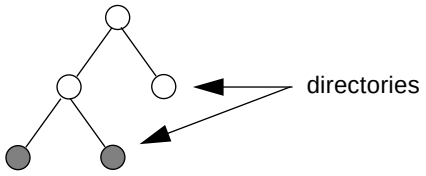
- Although both use directories, the other objects in an NIS+ namespace are tables and groups, not files.
- The NIS+ namespace is administered only through NIS+ administration commands (listed in Table 2-4 on page 29) or graphical user interfaces (GUIs) designed for that purpose (Administration Tool); it cannot be administered with standard UNIX file system commands or GUIs.
- The names of UNIX file system components are separated by slashes (`/usr/bin`), but the names of NIS+ namespace objects are separated by dots (Wiz.Com.).
- The "root" of a UNIX file system is reached by stepping through directories from right to *left* (`/usr/src/file1`), while the root of the NIS+ namespace is reached by stepping from left to *right* (Sales.Wiz.Com.).

Directories

Directory objects are the skeleton of the namespace. When arranged into a tree-like structure, they divide the namespace into separate parts. You may want to visualize a directory hierarchy as an upside-down tree, with the root of the tree at the top, and the leaves toward the bottom. The topmost directory in a namespace is the *root* directory. If a namespace is flat, it has only one directory, but that directory is nevertheless the root directory. The directory objects beneath the root directory are simply called “directories:”

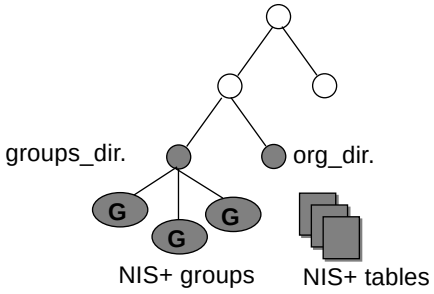


A namespace can have several levels of directories:



When identifying the relation of one directory to another, the directory beneath is called the *child* directory, and the directory above is called the *parent* directory.

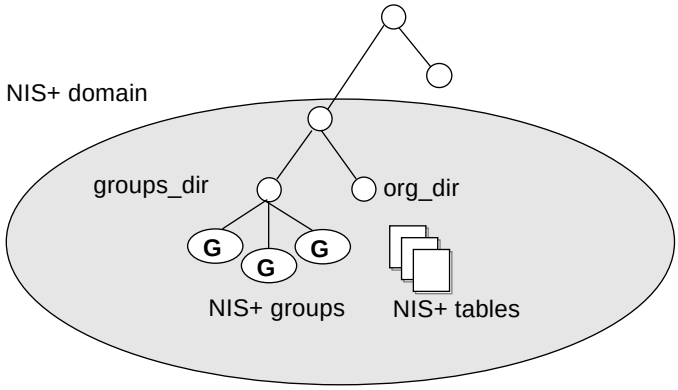
Whereas UNIX directories are designed to hold UNIX files, NIS+ directories are designed to hold NIS+ objects: other directories, tables and groups. Any NIS+ directory that stores NIS+ groups is named `groups_dir`. Any directory that stores NIS+ system tables is named `org_dir`.



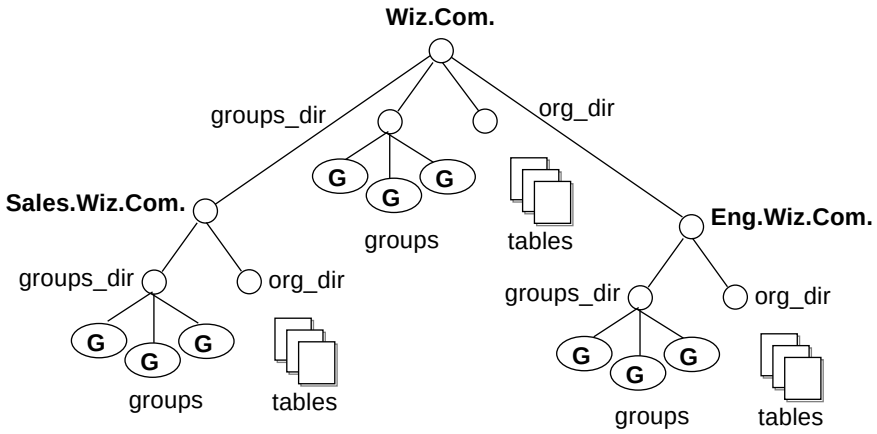
Technically, you can arrange directories, tables, and groups into any structure that you like. However, NIS+ directories, tables, and groups in a namespace are normally arranged into configurations called *domains*. Domains are designed to support separate portions of the namespace. For instance, one domain may support the Sales Division of a company, while another may support the Engineering Division.

Domains

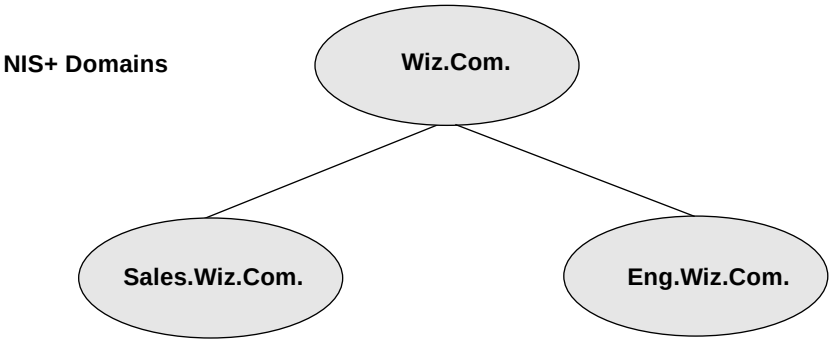
An NIS+ domain consists of a directory object, its `org_dir` directory, its `groups_dir` directory, and a set of NIS+ tables.



NIS+ domains are not *tangible* components of the namespace. They are simply a convenient way to *refer* to sections of the namespace that are used to support real-world organizations. Take the Wizard Corporation from Chapter 2 as an example. As you recall, at one point it had a Sales Division and an Engineering division. To support those divisions, its NIS+ namespace would most likely be arranged into three major directory groups, with a structure that looked like this:



Instead of referring to such a structure as three directories, six subdirectories, and several additional objects, referring to it as three domains is more convenient:

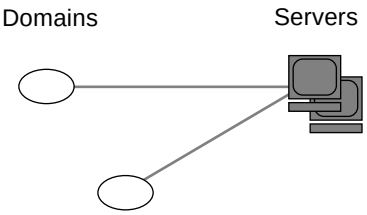


Part 2 of this manual describes how to configure domains.

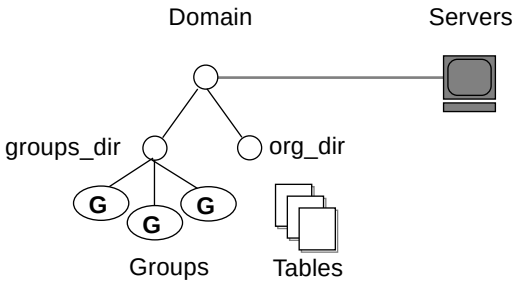
3

Servers

Every NIS+ domain is supported by a set of NIS+ *servers*. The servers store the domain's directories, groups, and tables, and answer requests for access from users, administrators, and applications. Each domain is supported by only one set of servers. However, a single set of servers can support more than one domain:



Remember that a domain is not an object, but only refers to a collection of objects. Therefore, a server that supports a domain is not actually associated with the domain, but with the domain's main *directory*:



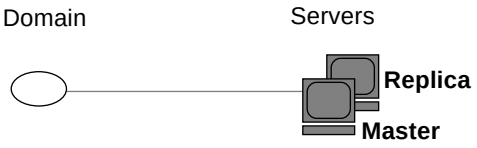
This connection between the server and the directory object is established during the process of setting up a domain. Although instructions are provided in Part 2, one thing is important to mention now: when that connection is established, the directory object stores the name and IP address of its server. This information is used by clients to send requests for service, as described later in this section.

Any Solaris 2.4-based workstation can be an NIS+ server. The software for both NIS+ servers and clients is bundled together into the Solaris 2.4 release. Therefore, any workstation that has the Solaris 2.4 software installed can

become a server or a client, or both. What distinguishes a client from a server is the *role it is playing.* If a workstation is providing NIS+ service, it is acting as an NIS+ server. If it is requesting NIS+ service, it is acting as an NIS+ client.

Because of the need to service many client requests, a workstation that will act as an NIS+ server might be configured with more computing power and more memory than the average client. And, because it needs to store NIS+ data, it might also have a larger disk. However, other than hardware to improve its performance, a server is not inherently different from an NIS+ client.

Two types of servers support an NIS+ domain: a master and its replicas:



The master server of the root domain is called the *root master* server. A namespace has only one root master server. The master servers of other domains are simply called master servers. Likewise, there are root replica servers and regular replica servers.

Both master and replica servers store NIS+ tables and answer client requests. The master, however, stores the master copy of a domain's tables. The replicas store only duplicates. The administrator loads information into the tables in the master server, and the master server propagates it to the replica servers.

This arrangement has two benefits. First, it avoids conflicts between tables because only one set of master tables exists; the tables stored by the replicas are only copies of the masters. Second, it makes the NIS+ service much more *available.* If either the master or a replica is down, another server can act as a backup and handle the requests for service.

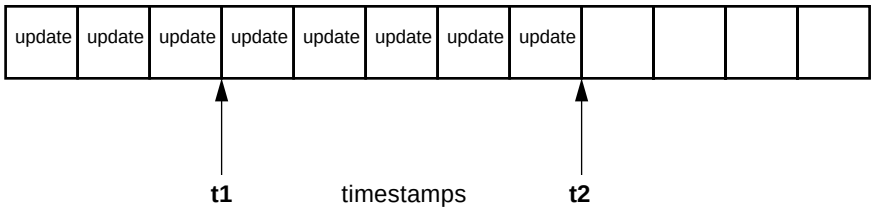
How Servers Propagate Changes

An NIS+ master server implements updates to its objects immediately; however, it tries to "batch" several updates together before it propagates them to its replicas. When a master server receives an update to an object, whether a directory, group, link, or table, it waits about two minutes for any other

updates that may arrive. Once it is finished waiting, it stores the updates in two locations: on disk and in a *transaction log* (it has already stored the updates in memory).

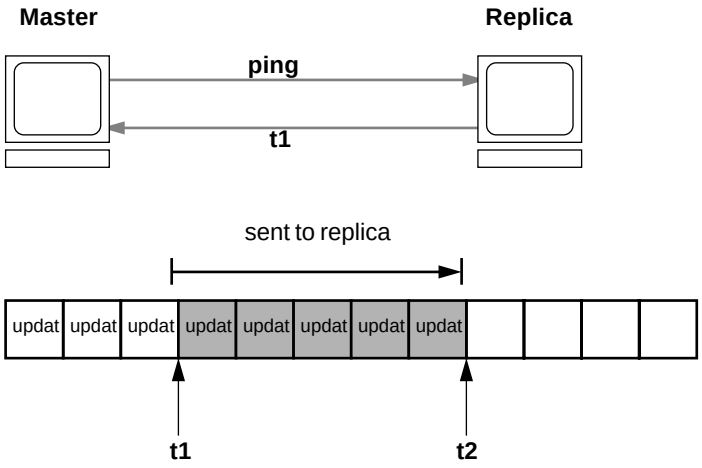
The transaction log is used by a master server to store changes to the namespace until they can be propagated to replicas. A transaction log has two primary components: updates and timestamps.

Transaction Log



An update is an actual copy of a changed object. For instance, if a directory has been changed, the update is a complete copy of the directory object. If a table entry has been changed, the update is a copy of the actual table entry. The timestamp indicates the time at which an update was made by the master server.

After recording the change in the transaction log, the master sends a message to its replicas, telling them that it has updates to send them. Each replica replies with the timestamp of the last update it received from the master. The master then sends each replica the updates it has recorded in the log since the replica's timestamp:



When the master server updates *all* its replicas, it clears the transaction log. In some cases, such as when a new replica is added to a domain, the master receives a timestamp from a replica that is before its earliest timestamp still recorded in the transaction log. If that happens, the master server performs a full *resynchronization*, or "resync." A resync downloads all the objects and information stored in the master down to the replica. During a resync, both the master and replica are busy. The replica cannot answer requests for information; the master can answer read requests but cannot accept update requests. Both respond with a "`Server Busy - Try Again`" message.

Clients

An NIS+ client is a workstation that has been set up to receive NIS+ service. Setting up an NIS+ client consists of establishing security credentials, making it a member of the proper NIS+ groups, verifying its home domain, verifying its switch configuration file and, finally, running the NIS+ initialization script. (Complete instructions are provided in Part 2.)

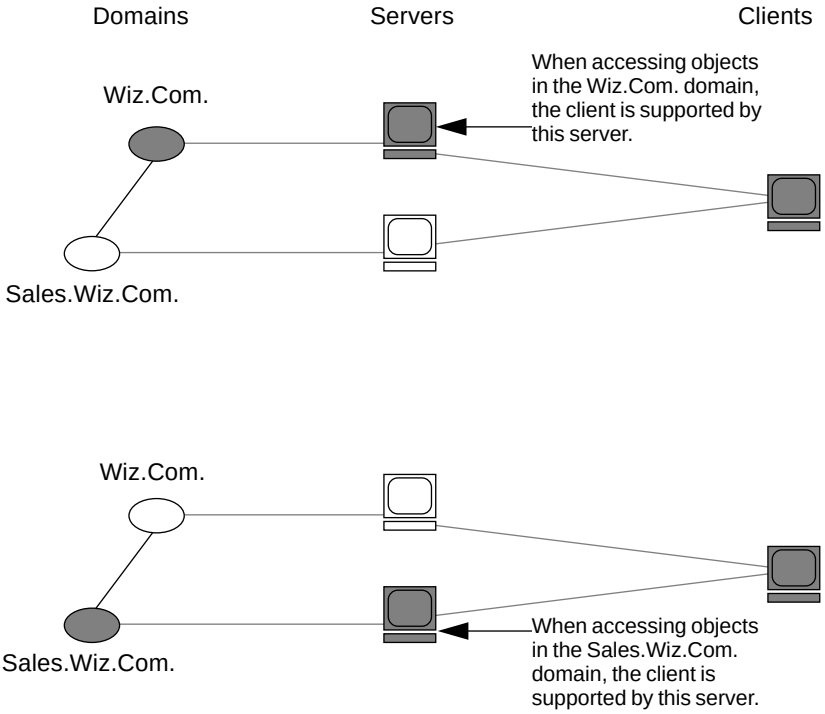
An NIS+ client can access any part of the namespace, subject to security constraints. In other words, if it has been authenticated and has been granted the proper permissions, it can access information or objects in any domain in the namespace.

Although a client can access the entire namespace, a client *belongs* to only one domain, which is referred to as its *home* domain. A client's home domain is usually specified during installation, but it can be changed or specified later. All the information about a client, such as its IP address and its credentials, is stored in the NIS+ tables of its home domain.

There is a subtle difference between being an NIS+ client and being listed in an NIS+ table. Entering information about a workstation into an NIS+ table does not automatically make that workstation an NIS+ client. It simply makes information about that workstation available to all NIS+ clients. That workstation cannot request NIS+ service unless it is actually set up as an NIS+ client.

Conversely, making a workstation an NIS+ client does not enter information about that workstation into an NIS+ table. It simply allows that workstation to receive NIS+ service. If information about that workstation is not explicitly entered into the NIS+ tables by an administrator, other NIS+ clients will not be able to get it.

When a client requests access to the namespace, it is actually requesting access to a particular domain in the namespace. Therefore, it sends its request to the server that supports the domain it is trying to access. Here is a simplified representation:



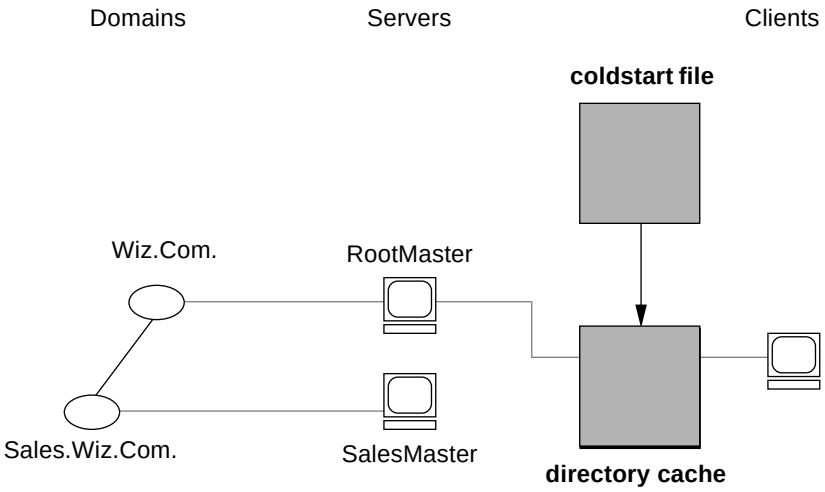
How does the client know which server that is? By a simple method of trial and error. Beginning with its home server, the client tries one server, then another, until it finds the right one. When a server cannot answer the client's request, it sends the client information to help locate the right server. Over time, the client builds up its own cache of information and becomes more efficient at locating the right server. The next section describes this process.

The Coldstart File and Directory Cache

When a client is initialized, it is given a *coldstart file*. The coldstart file gives a client a copy of a directory object that it can use as a starting point for contacting servers in the namespace. The directory object contains the address,

public keys, and other information about the master and replica servers that support the directory. Normally, the coldstart file contains the directory object of the client's home domain.

A coldstart file is used only to initialize a client's *directory cache*. The directory cache, managed by an NIS+ facility called the *cache manager*, stores the directory objects that enable a client to send its requests to the proper servers.



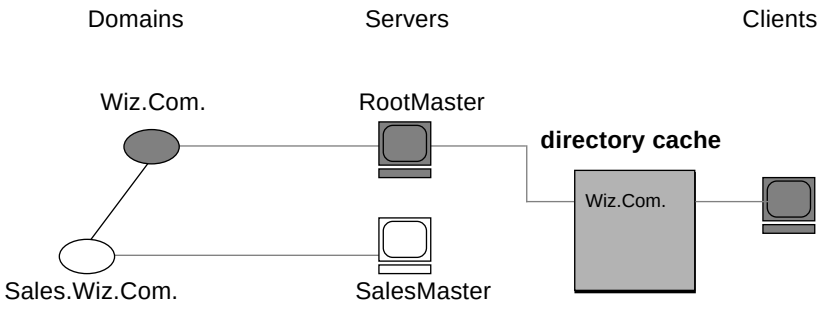
By storing a copy of the namespace's directory objects in its directory cache, a client can know which servers support which domains. (To view the contents of a client's cache, use the `nisshowcache` command, described in *Name Services Administration Guide*.) Here is a simplified example:

Domain	Directory Name	Supporting Server	IP Address
Wiz.Com.	Wiz.Com.	RootMaster	129.44.1.1
Sales.Wiz.Com	Sales.Wiz.Com.	SalesMaster	129.44.2.1
Eng.Wiz.Com.	Eng.Wiz.Com.	EngMaster	129.44.3.1
Intl.Sales.Wiz.Com.	Intl.Sales.Wiz.Com.	IntlSalesMaster	129.44.2.11

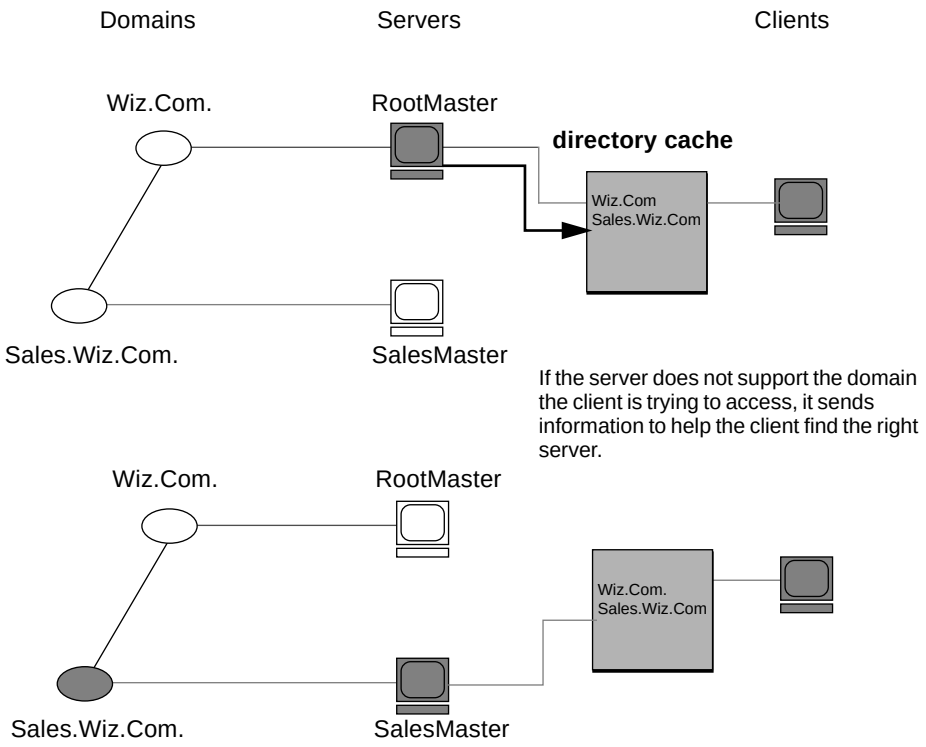
To keep these copies up-to-date, each directory object has a *time-to-live* field. Its default value is 12 hours. If a client looks in its directory cache for a directory object and finds that it has not been updated in the last 12 hours, the cache

manager obtains a new copy of the object. You can change a directory object's time-to-live value with the `nischttl` command, as described in *Name Services Administration Guide.* However, keep in mind that the longer the time to live, the higher the likelihood that the copy of the object will be out of date; and the shorter the time to live, the greater the network traffic and server load.

How does the directory cache accumulate these directory objects? As mentioned above, the coldstart file provides the first entry in the cache. Therefore, when the client sends its first request, it sends the request to the server specified by the coldstart file. If the request is for access to the domain supported by that server, the server answers the request.



If the request is for access to another domain (for example, Sales.Wiz.Com.), the server tries to help the client locate the proper server. If the server has an entry for that domain in its own directory cache, it sends a copy of the domain's directory object to the client. The client loads that information into its directory cache for future reference and sends its request to that server.



In the unlikely event that the server does not have a copy of the directory object the client is trying to access, it sends the client a copy of the directory object for its own home domain, which lists the address of the server's parent. The client repeats the process with the parent server, and keeps trying until it finds the proper server or until it has tried all the servers in the namespace. What the client does after trying all the servers in the domain is determined by the instructions in its name service switch configuration file. See Chapter 5, "Understanding the Name Service Switch," for details.

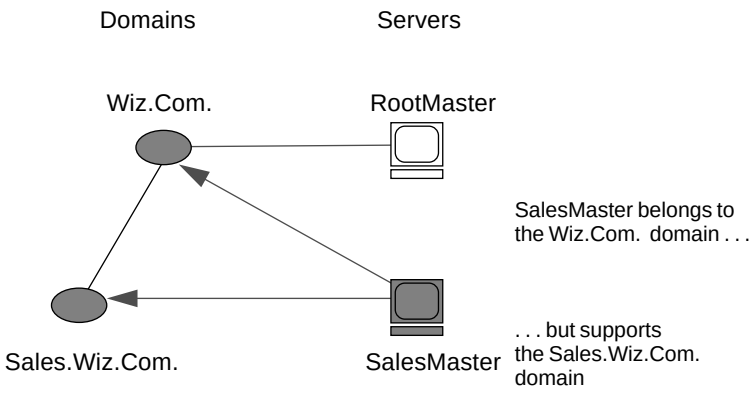
Over time, the client accumulates in its cache a copy of all the directory objects in the namespace and thus, the IP addresses of the servers that support them. When it needs to send a request for access to another domain, it can usually find the name of its server in its directory cache and send the request directly to that server.

An NIS+ Server Is Also a Client

An NIS+ server is also an NIS+ client. In fact, before you can set up a workstation as a server (as described in Part 2 of this manual), you must initialize it as a client. The only exception is the root master server, which has its own unique setup process.

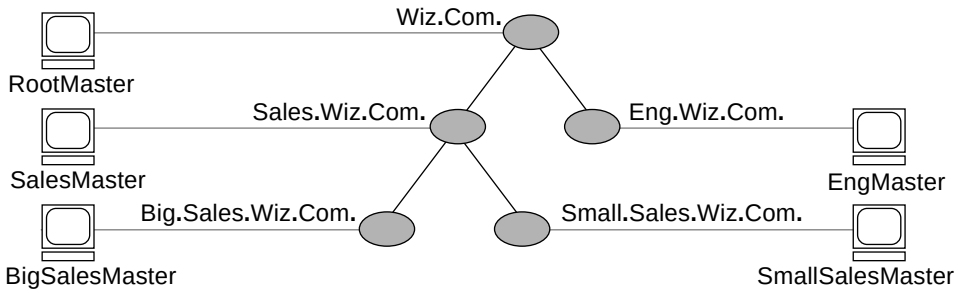
This means that in addition to *supporting* a domain, a server also *belongs* to a domain. In other words, by virtue of being a client, a server has a home domain. Its host information is stored in the Hosts table of its home domain, and its DES credentials are stored in the Cred table of its home domain. Like other clients, it sends its requests for service to the servers listed in its directory cache.

An important point to remember is that — except for the root domain — a server's home domain is the *parent* of the domain the server supports:



In other words, a server supports clients in one domain, but is a *client* of another domain. A server cannot be a client of a domain that it supports, with the exception of the root domain. Because they have no parent domain, the servers that support the root domain belong to the root domain itself.

For example, consider the following namespace:



The chart lists which domain each server supports and which domain it belongs to:

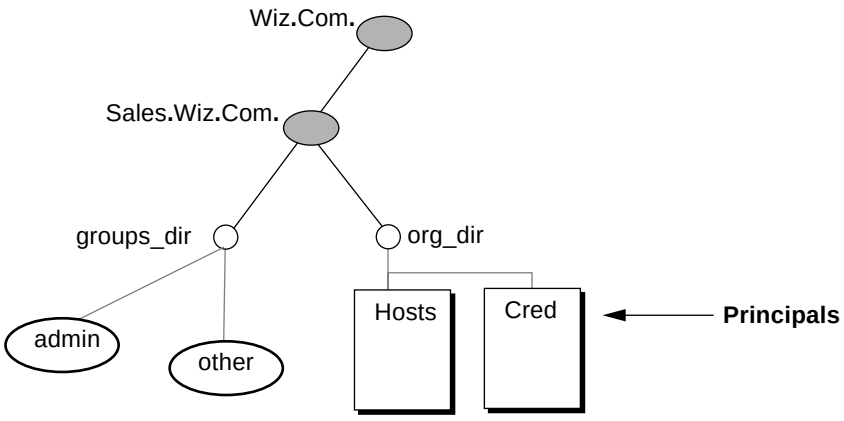
Server	Supports	Belongs to
RootMaster	Wiz.Com.	Wiz.Com.
SalesMaster	Sales.Wiz.Com.	Wiz.Com.
BigSalesMaster	Big.Sales.Wiz.Com.	Sales.Wiz.Com.
SmallSalesMaster	Small.Sales.Wiz.Com.	Sales.Wiz.Com.
EngMaster	Eng.Wiz.Com.	Wiz.Com.

Naming Conventions

Objects in an NIS+ namespace can be identified with two types of names: *partially qualified* and *fully-qualified.* A partially qualified name, also called a *simple* name, is simply the name of the object or any portion of the fully-qualified name. If during any administration operation you type the partially qualified name of an object or principal, NIS+ will attempt to expand the name into its fully qualified version. For details, see "NIS+ Name Expansion" on page 53.

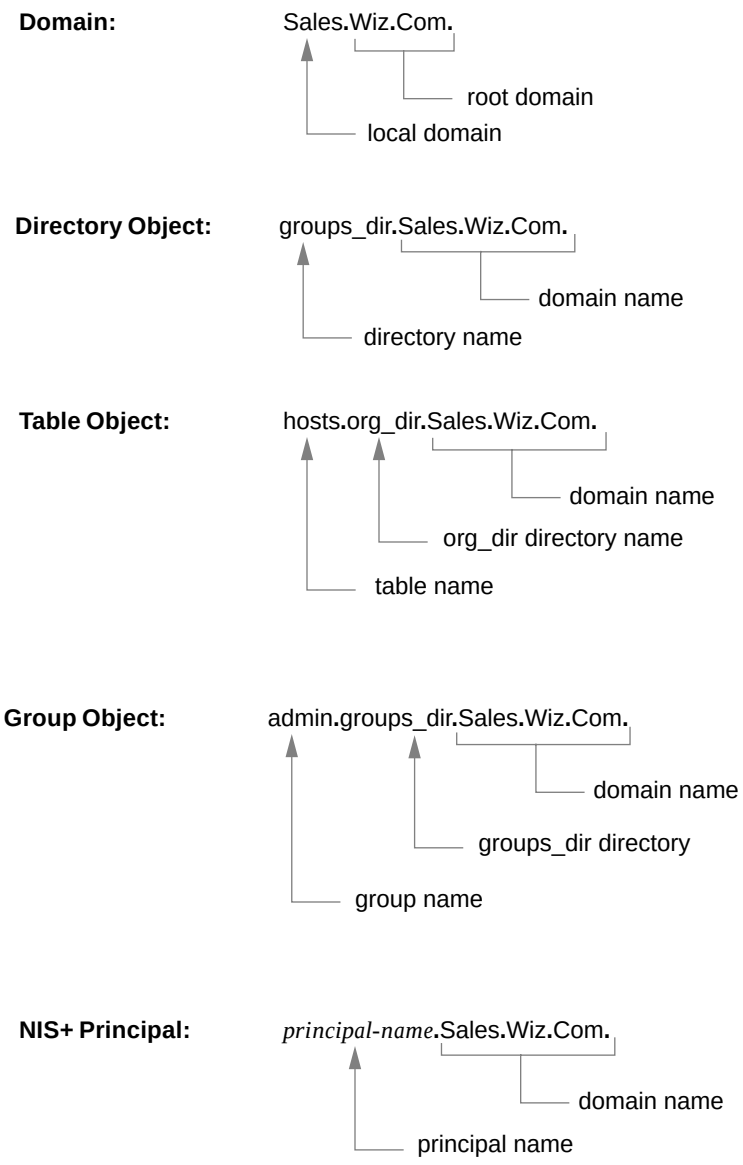
A fully-qualified name is the complete name of the object, including all the information necessary to locate it in the namespace, such as its parent directory, if it has one, and its complete domain name, including a trailing dot.

This varies among different types of objects, so the conventions for each type, as well as for NIS+ principals, is described separately. This namespace will be used as an example:



The fully-qualified names for all the objects in this namespace, including NIS+ principals, are summarized in Figure 3-1 on page 50.

Figure 3-1 Fully-Qualified Names of Namespace Components



For Domains

A fully-qualified domain name is formed from left to right, starting with the local domain and ending with the root domain. For example:

Wiz.Inc.	Wiz.Com.
Sales.Wiz.Inc.	Sales.Wiz.Com.
Intl.Sales.Wiz.Inc.	Intl.Sales.Wiz.Com.

The first line above shows the name of the root domain. The root domain must always have at least two labels and must end in a dot. The second label can be an Internet domain name, such as Com. The second and third lines above show the names of lower-level domains.

For Directory Objects

A directory's simple name is simply the name of the directory object. Its fully qualified name consists of its simple name plus the fully qualified name of its domain (which always includes a trailing dot):

groups_dir	(simple name)
groups_dir.Eng.Wiz.Com.	(fully-qualified name)

If you set up an unusual hierarchy in which several layers of directories do not form a domain, be sure to include the names of the intermediate directories. For example:

```
lowest_dir.lower_dir.low_dir.MyStrangeDomain.Com.
```

The simple name is normally used from within the same domain, and the fully-qualified name is normally used from a remote domain. However, by specifying search paths in a domain's NIS_PATH environment variable, you can use the simple name from remote domains (see "NIS+ Name Expansion" on page 53).

For Tables and Groups

Fully-qualified table and group names are formed by starting with the object name and appending the directory name, followed by the Fullyqualified domain name. Remember that all system table objects are stored in an `org_dir` directory and all group objects are stored in a `groups_dir` directory. (If you create your own NIS+ tables, you can store them anywhere you like.) Here are some examples of group and table names:

admin.groups_dir.Wiz.Inc.	admin.groups_dir.Wiz.Com.
admin.groups_dir.Sales.Wiz.Inc.	admin.groups_dir.Sales.Wiz.Com.
hosts.org_dir.Wiz.Inc.	hosts.org_dir.Wiz.Com.
hosts.org_dir.Sales.Wiz.Inc.	hosts.org_dir.Sales.Wiz.Com.

For Table Entries

To identify an entry in an NIS+ table, you need to identify the table object and the entry within it. This type of name is called an *indexed* name. It has the following syntax:

[*column=value*, *column=value*, . . .], *table-name*

Column is the name of the table column. *Value* is the actual value of that column. *Table-name* is the fully-qualified name of the table object. Here are a few examples of entries in the Hosts table:

```
[addr=129.44.2.1,name=pine],hosts.org_dir.Sales.Wiz.Com.
[addr=129.44.2.2,name=elm],hosts.org_dir.Sales.Wiz.Com.
[addr=129.44.2.3,name=oak],hosts.org_dir.Sales.Wiz.Com.
```

You can use as few column-value pairs inside the brackets as required to uniquely identify the table entry.

Some NIS+ administrative commands accept variations on this syntax. For details, see the `nistbladm`, `nismatch`, and `nisgrep` commands in *Name Services Administration Guide.*

For NIS+ Principals

NIS+ principal names are sometimes confused with secure RPC netnames. Both types of names are described in the security chapter in *Name Services Administration Guide.* However, one difference is worth pointing out now because it can cause confusion: NIS+ principal names *always* end in a dot and secure RPC netnames *never* do:

```
olivia.Sales.Wiz.Com.        ( NIS+ principal name )
unix.olivia@Sales.Wiz.Com ( secure RPC netname )
```

Also, even though credentials for principals are stored in a Cred table, neither the name of the Cred table nor the name of the `org_dir` directory is included in the principal name.

Accepted Symbols

You can form namespace names from any printable character in the ISO Latin 1 set. However, the names cannot start with these characters:

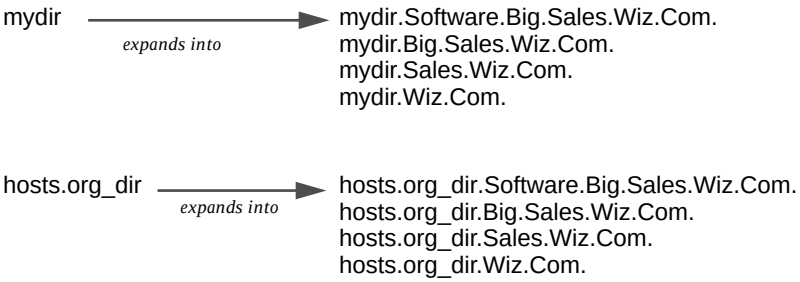
```
@       < >       +         [ ]         -       /
=       .         ,          :          ;
```

To use a string, enclose it in double quotes. To use a quote sign in the name, quote the sign too (for example, to use `ol'yeller`, type `ol"'"yeller`). To include white space (as in John Smith), use double quotes within single quotes, like this:

```
'"John Smith"'
```

NIS+ Name Expansion

Entering fully-qualified names with your NIS+ commands can quickly become tedious. To ease the task, NIS+ provides a name expansion facility. When you enter a partially qualified name, NIS+ attempts to find the object by looking for it under different directories. It starts by looking in the default domain. This is the home domain of the client from which you type the command. If it does not find the object in the default domain, NIS+ searches through each of the default domain's parent directories in ascending order until it finds the object. It stops after reaching a name with only two labels. Here are some examples (assume you are logged onto a client that belongs to the "Software.Big.Sales.Wiz.Com." domain).

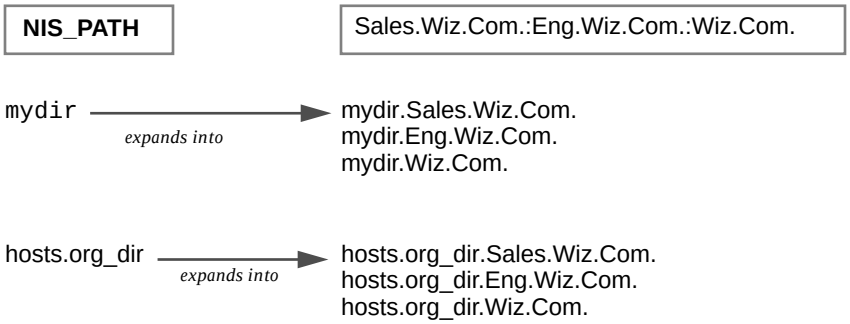


NIS_PATH Environment Variable

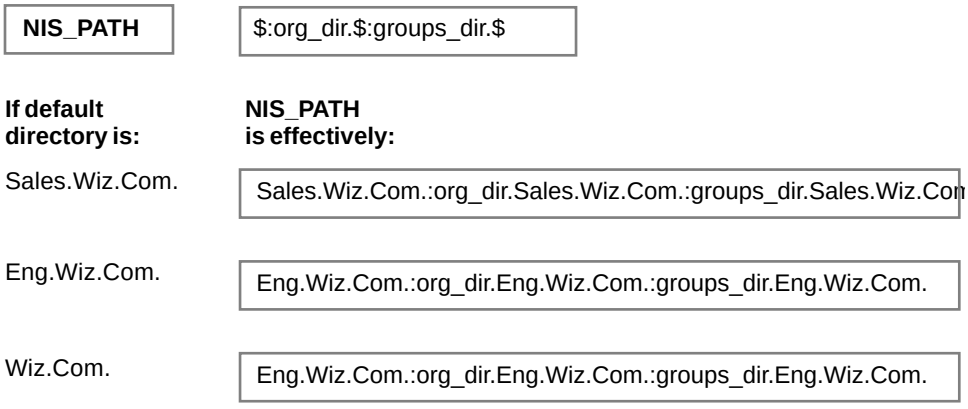
You can change or augment the list of directories NIS+ searches through by changing the value of the environment variable `NIS_PATH`. `NIS_PATH` accepts a list of directory names separated by colons:

```
setenv NIS_PATH directory1:directory2:directory3...
NIS_PATH=directory1:directory2:directory3...;export NIS_PATH
```

NIS+ searches through these directories from left to right. For example:



The `NIS_PATH` variable accepts a special symbol: `$`. You can append the `$` symbol to a directory name or add it by itself. If you append it to a directory name, NIS+ appends the default directory to that name. For example:



If you use the `$` sign by itself (for example, `org_dir.$:$`), NIS+ performs the standard name expansion described earlier: start looking in the default directory and proceed through the parent directories. In other words, the default value of `NIS_PATH` is `$`.

Understanding NIS+ Tables and Information

4

NIS+ stores a wide variety of network information in tables. This chapter describes the structure of those tables and provides a brief overview of how they can be set up

NIS+ Table Structure

NIS+ tables provide several features not found in simple text files or maps. They have a column-entry structure, they accept search paths, they can be linked together, and they can be set up in several different ways. NIS+ provides 16 preconfigured system tables, and you can also create your own tables. Table 4-1 lists the preconfigured NIS+ tables.

Table 4-1 NIS+ Tables

Table	Information in the Table
Hosts	Network address and hostname of every workstation in the domain
Bootparams	Location of the root, swap, and dump partition of every diskless client in the domain
Password	Password information about every user in the domain.
Cred	Credentials for principals who belong to the domain

Table 4-1 NIS+ Tables *(Continued)*

Table	Information in the Table
Group	The group password, group id, and members of every UNIX group in the domain
Netgroup	The netgroups to which workstations and users in the domain may belong
Mail_aliases	Information about the mail aliases of users in the domain
Timezone	The timezone of every workstation in the domain
Networks	The networks in the domain and their canonical names
Netmasks	The networks in the domain and their associated netmasks
Ethers	The ethernet address of every workstation in the domain
Services	The names of IP services used in the domain and their port numbers
Protocols	The list of IP protocols used in the domain
RPC	The RPC program numbers for RPC services available in the domain
Auto_home	The location of all user's home directories in the domain
Auto_master	Automounter map information

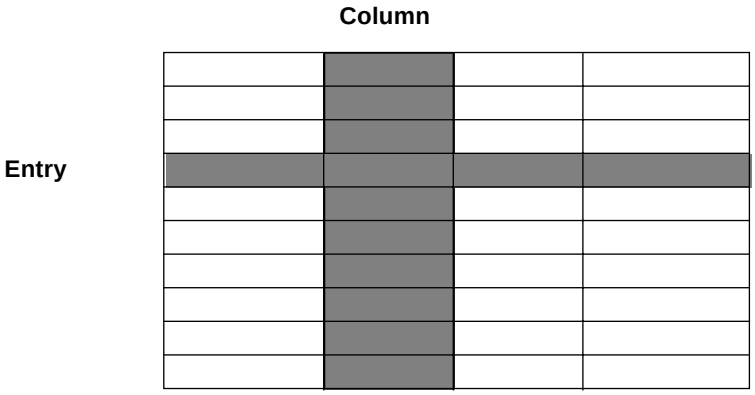
These tables store a wide variety of information, ranging from user names to Internet services. Most of this information is generated during a setup or configuration procedure. For instance, an entry in the Password table is created when a user account is set up. An entry in the Hosts table is created when a workstation is added to the network. And an entry in the Networks table is created when a new network is set up.

Since this information is generated from such a wide field of operations, much of it is beyond the scope of this manual. However, as a convenience, *Name Services Administration Guide* summarizes the information contained in each column of the tables, providing details only when necessary to keep things from getting confusing, such as when distinguishing groups from NIS+ groups and netgroups. For thorough explanations of the information, consult Solaris 2.4, system and network administration manuals.

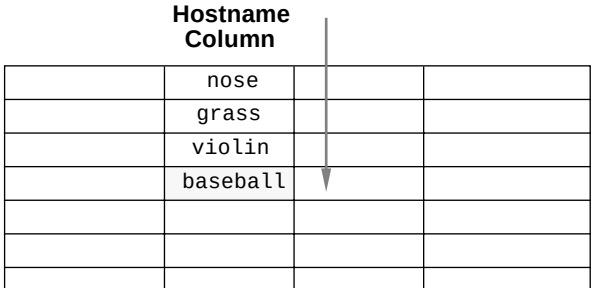
The Cred table, because it contains only information related to NIS+ security, is described in the security chapter in *Name Services Administration Guide.*

Columns and Entries

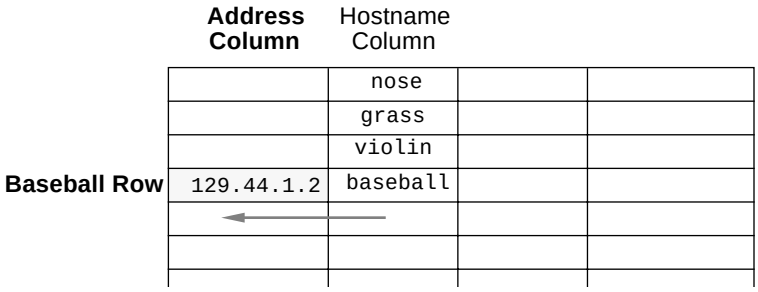
Although NIS+ tables store different types of information, they all have the same underlying structure; they are each made up of rows and columns (the rows are called “entries,” or “entry objects”):



A client can access information by a key, or by any column that is searchable. For example, to find the network address of a workstation named “baseball,” a client could look through the hostname column until it found “baseball”



it then would move along the baseball entry to find its network address:



Because a client can access table information at any level, NIS+ provides security mechanisms for all three levels. For instance, an administrator could assign Read rights to everyone for a table at the object level, Modify rights to the owner at the column level, and Modify rights to the group at the entry level. Details about table security are provided in *Name Services Administration Guide*.

Search Paths

A table contains information only about its *local* domain. For instance, tables in the Wiz.Com. domain contain information only about the users, clients, and services of the Wiz.Com. domain. The tables in the Sales.Wiz.Com. domain store information only about the users, clients, and services of the Sales.Wiz.Com. domain. And so on.

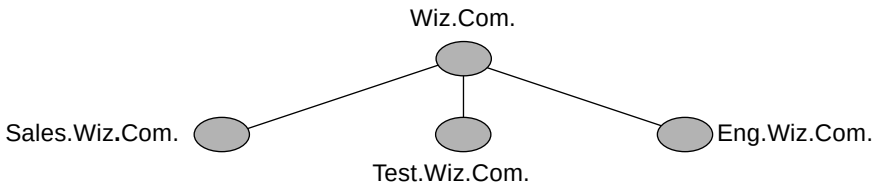
If a client in one domain tries to find information that is stored in another domain, it has to provide a fully qualified name. As described in "NIS+ Name Expansion" if the `NIS_PATH` environment variable is set up properly, the NIS+ service will do this automatically.

Every NIS+ table can also specify a *search path* that a server will follow when looking for information. The search path is an ordered list of NIS+ tables, separated by colons:

table:*table*:*table*. . .

The table names in the search path don't have to be fully qualified; they can be expanded just like names entered in the command line. When a server cannot find information in its local table, it returns the table's search path to the client. The client uses that path to look for the information in every table named in the search path, in order, until it finds the information or runs out of names.

Here is an example that demonstrates the benefit of search paths. Assume the following domain hierarchy:



The Hosts table of the lower three domains have the following contents:

Sales.Wiz.Com.		**Test.Wiz.Com.**		**Eng.Wiz.Com.**	
127.0.0.1	localhost	127.0.0.1	localhost	127.0.0.1	localhost
129.44.2.10	vermont	129.44.4.10	nebraska	129.44.3.10	georgia
129.44.2.11	maine	129.44.4.11	oklahoma	129.44.3.11	florida
129.44.2.12	cherry	129.44.4.12	corn	129.44.3.12	orange
129.44.2.13	apple	129.44.4.13	wheat	129.44.3.13	potato
129.44.2.14	mailhost	129.44.4.14	mailhost	129.44.3.14	mailhost

Assume now that a user logged onto a client in the Sales.Wiz.Com. domain wants to log in remotely to another client. If that user does not provide a fully qualified name, it can only remotely log on to five workstations: `vermont`, `maine`, `cherry`, `apple`, and the mailhost.

Now assume that the search path of the Hosts table in the Sales.Wiz.Com. domain listed the Hosts tables from the Test.Wiz.Com. and Eng.Wiz.Com. domains:

search path—hosts.org_dir.Test.Wiz.Com.:hosts.org_dir.Eng.Wiz.Com.

Now a user in the Sales.Wiz.Com. domain can enter something like `rlogin oklahoma`, and the NIS+ server will find it. It will first look for `oklahoma` in the local domain, but when it does not find a match, it will look in the Test.Wiz.Com. domain. How does the client know how to find the Test.Wiz.Com. domain? As described in Chapter 3, "Understanding the NIS+ Namespace", the information is stored in its directory cache. If it is not stored in its directory cache, the client will obtain the information by following the process described in Chapter 3, "Understanding the NIS+ Namespace".

There is a slight drawback, though, to specifying a search path. If the user were to enter an incorrect name, such as `rlogin potatoe`, the server would need to look through three tables — instead of just one — before returning an error message. If you set up search paths throughout the namespace, an operation may end up searching through the tables in 10 domains instead of just 2 or 3. Another drawback is a performance loss from having many clients contact more than one set of servers when they need to access NIS+ tables.

You should also be aware that since "mailhost" is often used as an alias, when trying to find information about a specific mailhost, you should use its fully qualified name (for example, `mailhost.Sales.Wiz.Com.`), or NIS+ will return *all* the mailhosts it finds in all the domains it searches through.

You can specify a table's search path by using the `-p` option to the `nistbladm` command, as described in *Name Services Administration Guide.*

Ways to Set Up Tables

The second part of this book provides complete step-by-step instructions for setting up NIS+ tables using the NIS+ scripts, but here is an overview of the process. Setting up NIS+ tables involves three or four tasks:

1. Creating the `org_dir` directory
2. Creating the system tables
3. Creating non-system tables (optional)
4. Populating the tables with information

As described in Chapter 3, "Understanding the NIS+ Namespace", NIS+ system tables are stored under an `org_dir` directory. So, before you can create any tables, you must create the `org_dir` directory that will hold them. You can do this in three ways.

- use the `nisserver` script
- use the `nismkdir` command
- use the `/usr/lib/nis/nissetup` utility

The `nisserver` script creates the appropriate directories and a full set of system tables. The `nismkdir` command simply creates the directory. The `nissetup` utility creates the `org_dir` and `groups_dir` directories and a full set of system tables.

The `nisserver` script, the recommended way, is described in Part 2. The `nissetup` and `nismkdir` utilities are described in *Name Services Administration Guide.*

Another benefit of the `nissetup` utility is its capability to assign the proper access rights to the tables of a domain whose servers are running in NIS-compatibility mode. When entered with the `-Y` flag, it assigns Read permissions to the "Nobody" category of the objects it creates, allowing NIS clients, who are unauthenticated, to get information from the domain's NIS+ tables.

The 16 NIS+ system tables and the type of information they store are described in *Name Services Administration Guide.* To create them, you could use one of the three ways mentioned above. The `nistbladm` utility also creates and modifies NIS+ tables. You could conceivably create all the tables in a namespace with the `nistbladm` command, but you would have to type much more and you would have to know the correct column names and access rights. A much, much easier way is to use the `nisserver` script.

To create a non-system table—that is, a table that has not been preconfigured by NIS+—use the `nistbladm` command.

You can populate NIS+ tables in three ways: from NIS maps, from ASCII files (such as `/etc` files), and manually.

If you are upgrading from the NIS service, you already have most of your network information stored in NIS maps. You *don't* have to re-enter this information manually into NIS+ tables. You can transfer it automatically with the `nispopulate` script or the `nisaddent` utility.

If you are not using another network information service, but maintain network data in a set of `/etc` files, you *don't* have to re-enter this information either. You can transfer it automatically, also using the `nispopulate` script or the `nisaddent` utility.

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 NIS+ tables. You can do this with the `nistbladm` command. You can also do it by entering all the information for a particular table into an *input file*—which is essentially the same as an `/etc` file—and then transferring the contents of the file with the `nispopulate` script or the `nisaddent` utility.

How Tables Are Updated

When a domain is set up, its servers receive their first versions of the domain's NIS+ tables. These versions are stored on disk, but when a server begins operating, it loads them into memory. When a server receives an update to a table, it immediately updates its memory-based version of the table. When it receives a request for information, it uses the memory-based copy for its reply.

Of course, the server also needs to store its updates on disk. Since updating disk-based tables takes time, all NIS+ servers keep *log* files for their tables. The log files are designed to temporarily store changes made to the table, until they can be updated on disk. They use the table name as the prefix and append `.log`. For example:

```
hosts.log
bootparams.log
password.log
```

You should update disk-based copies of a table on a daily basis so that the log files don't grow too large and take up too much disk space. This process is called *checkpointing*. To do this, use the `nisping -C` command.

Understanding the Name Service Switch 5

The name service switch, referred to as the “switch,” is not really part of NIS+, but it enables NIS+ clients (actually, clients of `getXXbyYY()` routines) to obtain their network information from one or more of these *sources*: NIS+ tables, NIS maps, the DNS hosts table, and local `/etc` files. This chapter describes the switch and what it can do. It has four sections

About the Name Service Switch

An NIS+ client can obtain its information from one or more of the switch’s sources in place of, or in addition, to NIS+ tables. For example, an NIS+ client could obtain its hosts information from an NIS+ table, its group information from NIS maps, and its password information from a local `/etc` file. Plus, it could specify the conditions under which the switch must use each source (see “Search Criteria” on page 65).

These choices are specified in a special configuration file called `nsswitch.conf`. This file is automatically loaded into every workstation’s `/etc` directory by the Solaris 2.4 software, along with three alternate versions:

- `/etc/nsswitch.nisplus`

- `/etc/nsswitch.nis`
- `/etc/nsswitch.files`

These alternate files contain the default switch configurations used by the NIS+ service, NIS, and local files. (They are described later in this section.) No default file is provided for DNS, but you can edit any of these files to use DNS.

When the Solaris 2.4 software is first installed on a workstation, the installer must select the workstation's default naming service: NIS+, NIS, or local files. During the installation, the corresponding configuration file is copied into the `/etc/nsswitch.conf` file.

You can change the sources of information used by an NIS+ client by creating your own customized configuration file and copying it over `/etc/nsswitch.conf`. Its syntax is described below, and instructions are provided in *Name Services Administration Guide.*

Format of the `nsswitch.conf` *File*

The `nsswitch.conf` file is essentially a list of 15 types of information and their sources, not necessarily in this order:

```
aliases:                    source(s)
bootparams:                 source(s)
ethers:                     source(s)
group:                      source(s)
hosts:                      source(s)
netgroup:                   source(s)
netmasks:                   source(s)
networks:                   source(s)
passwd: (includes shadow)   source(s)
protocols:                  source(s)
publickey:                  source
rpc:                        source(s)
services:                   source(s)

automount:                  source(s)

sendmailvars                source(s)
```

The information for the Auto_home and Auto_master tables is combined into one category, called “automount.” The timezone table does not use the switch, so it is not included in the list. Possible sources are listed in Table 5-1:

Table 5-1 Possible Switch Sources

Source	Description
files	A local file stored in the client’s `/etc` directory (for example, `/etc/passwd`)
nisplus	An NIS+ table
nis	An NIS map
compat	Only for the Password and Group entries, supports the old-style “+” or “-” syntax in the `/etc/passwd`, `/etc/shadow`, and `/etc/group` files.
dns	DNS, but only for the hosts entry.

If an information type has only one source, the switch searches for the information in that source only. (If it does not find the information, it stops searching and returns a status message. The status message is passed to the library routine that requested the information. What the routine does with the status message varies from routine to routine.)

If a table has more than one source, the switch starts by searching for the information in the first source. If it does not find the information there, it tries the next source. The switch continues searching through the sources until it has tried them all.

If the switch still does not find the information, it stops searching and returns a status message. However, you can specify a different course of action, such as continuing to search for the information, done with *search criteria*.

Search Criteria

The switch searches through the sources one at a time. If it finds the information it is looking for in the first source, it returns a successful status message and passes the information to the library routine that asked for it. If the switch does *not* find the information, it returns one of three unsuccessful

status messages, depending on the reason for not finding the information, and moves to the next source. The four possible status messages are listed in Table 5-2:

Table 5-2 Switch Status Messages

Status	Meaning
SUCCESS	The requested entry was found in the source
UNAVAIL	The source is not responding or is unavailable
NOTFOUND	The source responded with “No such entry”
TRYAGAIN	The source is busy; it might respond next time

You can instruct the switch to respond to status messages with either of these two *actions* shown in Table 5-3:

Table 5-3 Responses to Switch Status Messages

Action	Meaning
return	Stop looking for the information
continue	Try the next source, if there is one

Default Search Criteria

The switch’s default search criteria are the same for every source. Described in terms of the status messages listed above, they are:

- SUCCESS=return
- UNAVAIL=continue
- NOTFOUND=continue
- TRYAGAIN=continue

You can change the default search criteria for any source, using the `STATUS=action` syntax shown above. For example:

```
hosts:     nis
networks:  nis [NOTFOUND=return] files
protocols: nis [NOTFOUND=return] files
```

In the second line of the example above, when the switch searches for information in NIS maps and gets a NOTFOUND status message, instead of searching through the second source, it stops looking. It would search through files only if the NIS service was unavailable.

What if the Syntax is Wrong?

Client library routines contain compiled-in default entries that are used if an entry in the `nsswitch.conf` file is either missing or syntactically incorrect. These entries are the same as the default `nsswitch.conf` file.

The name service switch assumes that the spelling of table and source names is correct. If you misspell a table or source name, the switch uses the default values instead.

Default `nsswitch.conf` *File*

The default `nsswitch.conf` file shipped with Solaris 2.4 is actually a copy of the `nsswitch.nis` file, described below. You can change it to the NIS+ version by copying the `nsswitch.nisplus` file over the `/etc/nswitch.conf` file.

The switch provides three alternate configuration files in addition to the default `/etc/nsswitch.conf` file. Each is described below.

`nsswitch.nisplus` *File*

The `nsswitch.nisplus` configuration file specifies NIS+ as the primary source for all information except passwd, group, automount, and aliases. For those files, the primary source is local `/etc` files and the secondary source is an NIS+ table. The `[NOTFOUND=return]` search criterion instructs the switch

to stop searching the NIS+ tables if it receives a “No such entry” message from them. It searches through local files only if the NIS+ server is unavailable. Here is a copy of the file with all the comments stripped out:

```
passwd:     files nisplus
group:      files nisplus

hosts:      nisplus [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
```

DNS Forwarding for NIS+ Clients

NIS+ clients do *not* have implicit DNS forwarding capabilities like NIS clients do. Instead, they take advantage of the switch. To provide DNS forwarding capabilities to an NIS+ client, change its hosts entry to:

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

The nsswitch.nis *File*

The nsswitch.nis configuration file is almost identical to the NIS+ configuration file, except that it specifies NIS maps in place of NIS+ tables.

```
passwd:     files nis
group:      files nis

hosts:      nis [NOTFOUND=return] files
services:   nis [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
protocols:  nis [NOTFOUND=return] files
rpc:        nis [NOTFOUND=return] files
ethers:     nis [NOTFOUND=return] files
netmasks:   nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey:  nis [NOTFOUND=return] files

netgroup:   nis

automount:  files nis
aliases:    files nis
```

Because the search order for passwd and group is `files nis`, you don't need to place the "+" entry in the /etc/passwd and /etc/group files.

DNS Forwarding for NIS Clients

If an NIS client is using the DNS forwarding capability of a NIS-compatible NIS+ server, its nsswitch.conf file should *not* have the following syntax for the hosts file:

```
hosts:  nis dns files
```

Since DNS forwarding automatically forwards host requests to DNS, the syntax shown above would cause the NIS+ server to forward unsuccessful requests to the DNS servers twice, impacting performance.

To take best advantage of DNS forwarding, use the default syntax for the nsswitch.nis file, as shown in the box above.

5

The `nsswitch.files` *File*

The `nsswitch.files` configuration file specifies local `/etc` files as the only source of information for the workstation.

```
passwd:     files
group:      files
hosts:      files
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
bootparams: files
publickey:  files

netgroup:   files

automount:  files
aliases:    files
services:   files
```

There is no “files” source for netgroup, so the client simply won’t use it.

Part 2 — Nis+ Configuration Tutorial

This part of the manual describes how to use the NIS+ scripts in a tutorial that shows you how to configure a small namespace that includes one subdomain. It has two chapters.

Getting Started with NIS+ 6

This chapter discusses the information you need to assemble before you start NIS+. It also describes the NIS+ scripts and what they will and will not do.

Before You Start NIS+

Before you start to set up NIS+ at your site, you need to do some planning and to know certain information about the machines at your site. In addition, you must have at least one system already running at your site that contains 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.)

6

Planning Your NIS+ Layout

To plan the structure of your NIS+ namespace:

- 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

Detailed explanations of these tasks are in Part I of this book. Appendix A, "Pre-Setup Worksheets" contains blank worksheets that you can use to help plan your NIS+ 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 tips and insights. The NIS+ scripts easily enable you to start NIS+ with data from NIS maps. Chapter 7, "Setting Up NIS+" shows you how to use the NIS+ scripts to create a NIS+ namespace from either system files or NIS maps.

You don't have to do any planning to run through the tutorial in Chapter 7, "Setting Up NIS+". You just need a few networked machines with which to practice. Be sure to plan your site's hierarchy before you move from the tutorial to setting up your real NIS+ namespace.

Determining Server and System Space 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:

1. Assign one master server per domain in the hierarchy.
2. Add at least one replica server for each domain. (A replica can answer requests when the master is unavailable).
3. 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 upon four factors:

- Disk space consumed by the Solaris 2.4 software
- Disk space for `/var/nis` (and `/var/yp`)
- Amount of memory
- Swap space required for NIS+ processes

The Solaris 2.4 software can consume over 220 megabytes (Mbytes) of disk space (including the OpenWindows™ software), depending on how much of it you install. (This is an estimate; for exact numbers, see *SPARC: Installing Solaris Software* or *x86: Installing Solaris Software*.) 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 Kilobytes of disk space per client of the domain. For example, if a domain has 1,000 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 1,000 clients, allocate 15 to 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.

6

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`(1M) process size because the server process forks during certain operations. See "How to Configure a Client as an NIS+ Server" and the `rpc.nisd`(1M) man page for more information.

About the NIS+ Scripts

The three NIS+ scripts—`nisserver`(1M), `nispopulate`(1M), and `nisclient`(1M)—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 6-1 describes what each script does.

Table 6-1 NIS+ scripts

NIS+ Script	What It Does
`nisserver`(1M)	Sets up the root master, non-root master and replica servers with level 2 security (DES)
`nispopulate`(1M)	Populates NIS+ tables in a specified domain from their corresponding system files or NIS maps
`nisclient`(1M)	Creates NIS+ credentials for hosts and users; initializes NIS+ hosts and users

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`(1M), `nispopulate`(1M), and `nisclient`(1M) man pages for complete descriptions of these commands and their options. Chapter 7, "Setting Up NIS+," 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 running them with 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 rather than reading it now.

The `nisserver`(1M) 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 7, "Setting Up NIS+" for how to use the `nisgrpadm` (1M) 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`(1M) daemon on remote servers, which is required to complete server installation

  See Chapter 7, "Setting Up NIS+" for how to use the `rpc.nisd` (1M) command instead of one of the NIS+ scripts to change NIS+ client machines into non-root servers.

The `nisclient`(1M) script does not set up an NIS+ client to resolve hostnames using DNS. You need to explicitly set DNS for clients that require this option.

See *Name Services Administration Guide* for information on how to perform any of the above tasks with individual commands.

6

Setting Up NIS+ 7

This chapter shows you how to set up a basic NIS+ namespace using the scripts `nisserver` (1M), `nispopulate` (1M), and `nisclient` (1M) in combination with a few NIS+ commands. Setting up NIS+ will be much simpler if you use these scripts than if you use the NIS+ commands individually; procedures using commands are described in *Name Services Administration Guide*. The NIS+ setup scenarios described below use the scripts' default values. See *Name Services Administration Guide* for information on how to customize your setup or if you want to set up an NIS+ namespace by hand. See "Glossary" for definitions of terms and acronyms you don't recognize.

This chapter provides step-by-step procedures for the following tasks:

7

See the `nisserver`(1M), `nispopulate`(1M), and `nisclient`(1M) man pages for complete descriptions of the scripts.

It is strongly suggested that you *not* use the small sample NIS+ namespace described in this tutorial as a basis for your actual NIS+ namespace. You are advised to 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 + Set up Overview

Table 7-1 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 7-1 Recommended NIS+ Setup Procedure Overview

Activity	Description	Script/NIS+ Commands
Set Up Root Domain	Create the root domain. Set up and initialize the root master server. Create the root domain admin group.	`nisserver`(1M)
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`(1M) `nisgrpadm`(1M) `nisping`(1M)
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`(1M)
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`(1M)
Set Up Root Replicas	Designate one or more of the servers you just set up as a replica of the root domain.	`nisserver`(1M)
Set Up Non-Root Domains	Create a new domain. Designate previously enabled server as its master. Create its admin group and admin credentials.	`nisserver`(1M)
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`(1M)

Table 7-1 Recommended NIS+ Setup Procedure Overview *(Continued)*

Activity	Description	Script/NIS+ Commands
Set Up 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`(1M)
Set Up Servers for Lower Level	If the namespace will have lower-level domains, convert some of the second level domain's clients into the servers that will later become masters and replicas of the lower levels.	`nisserver`(1M)
Set Up Other Domains	Set up other domains, whether at the same level as the new domain or beneath it, by repeating activities 6-9.	

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

Script Prerequisites

Before you use the scripts to create your actual namespace you have to:

1. Plan your NIS+ layout.
2. Choose a root domain name.
3. Choose a root server machine.
4. Choose your client machines.

Note – The machine that will be designated the root server must be up and running and you must have superuser access to it.

To create the sample namespace, you only need to do steps 3 and 4 above. The tutorial does the NIS+ layout planning for you and chooses a domain name.

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 the Network Information Service (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* 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 to have 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 (for the wiz.com. domain)
- Four clients:
  - The first client will become a root replica (for the wiz.com. domain)
  - The second client will become a master server for a new subdomain (for the subwiz.wiz.com. domain)
  - The third client will become a non-root replica server of the new subdomain (for the subwiz.wiz.com. domain)
  - The fourth client will remain solely a client of the root domain (wiz.com.)
- Two clients 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 7-1 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.

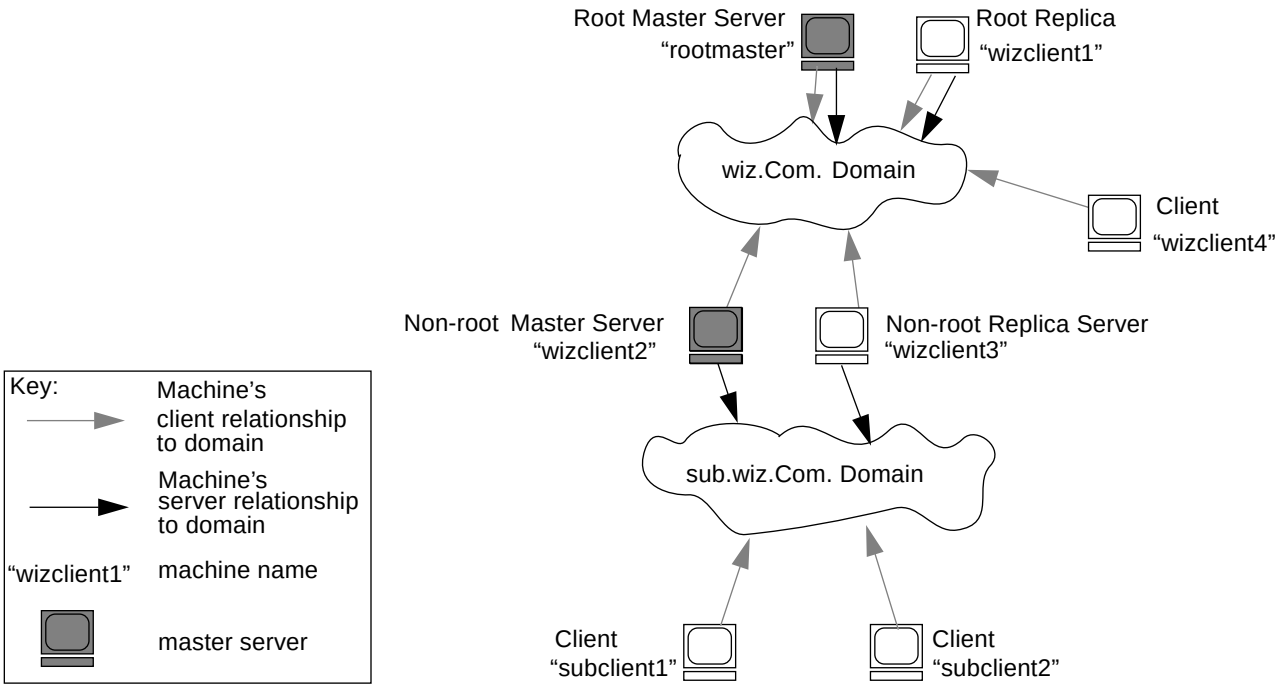


Figure 7-1 Sample NIS+ domain

Summary of NIS+ Scripts Command Lines

Table 7-2 contains the generic sequence of NIS+ scripts and commands you will use to create the NIS+ domains shown in Figure 7-1. 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 7-2 as a quick

reference guide to the appropriate command lines. Table 7-3 on page 123 is a summary of the actual commands with the appropriate variables that you will type to create the sample NIS+ namespace.

Table 7-2 NIS+ Domains Set Up Command Lines Summary

Purpose	On Which Machine	Command Line
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*`.` **`-h`** *NIS_servername*`\` **`-a`** *NIS_server_ipaddress* **`-y`** *NIS_domain* 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 (& DNS) compatibility	Client machine as superuser	**`rpc.nisd`** or **`rpc.nisd -Y`** or **`rpc.nisd -Y -B`**
Convert a server to a root replica	Root master server as superuser	**`nisserver -R -d`** *domain*`.` **`-h`** *clientname*
Convert a server to a non-root master server	Root master server as superuser	**`nisserver -M -d`** *newsubdomain*`.`*domain*`.` **`-h \`** *clientmachine*
Populate the new master server tables from files or from NIS maps	New subdomain master server as superuser	**`nispopulate -F -p`** */subdomaindirectory* **`-d \`** *newsubdomain*`.`*domain*`.` or **`nispopulate -Y -d`** *newsubdomain*`.`*domain*`.` **`-h \`** *NIS_servername* **`-a`** *NIS_server_ipaddress* **`-y`** *NIS_domain*
Convert a client to a master server replica	Subdomain master server as superuser	**`nisserver -R -d`** *subdomain*`.`*domain*`.` **`-h`** *clientname*

Table 7-2 NIS+ Domains Set Up Command Lines Summary *(Continued)*

Purpose	On Which Machine	Command Line
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 new `-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. See the scripts' man pages for more information.

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`(1M) script with default settings. The root master server will use the following defaults:

- Security level 2 (DES)—the highest level of NIS+ security
- NIS (YP) compatible set to OFF (instructions for setting NIS (`yp`) 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`(1M) 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 Chapter 5, "Understanding the Name Service Switch" for information on the Name Service Switch.

Prerequisites to Running `nisserver` *(1M)*

While this is usually true, 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.

▼ 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 with either of the following commands. The first example shows the C shell command; the second example shows the Bourne shell command.

```
rootmaster# setenv PATH $PATH:/usr/lib/nis

rootmaster# PATH=$PATH:/usr/lib/nis; export PATH
```

2. **Type the following 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 89 for instructions on how to change this name.

"NIS (YP) 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 89.

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 *Name Services Administration Guide* 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" below 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) y

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
groups_dir.wiz.com. created
passwd.org_dir.wiz.com. created
group.org_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
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
protocols.org_dir.wiz.com. created
rpc.org_dir.wiz.com. created
services.org_dir.wiz.com. created
timezone.org_dir.wiz.com. created

adding credential for rootmaster.wiz.com...
Enter login password:
```

The nissetup(1M) command creates the directories for each NIS+ table.

5. **Type your machine's root password at the prompt, then 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 91.

▼ How to Change Incorrect Information

If you typed n because some or all of the information returned to you was wrong in step 2 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 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 (YP) 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`(1M) daemon before it will work. See "How to Configure a Client as an NIS+ Server" 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." 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 *Name Services Administration Guide* for more information on changing security levels.

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`(1M) 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`*(1M)*

Before you can run the script `nispopulate`(1M):

- The information in the files must be formatted appropriately for the table into which it will be loaded. Chapter 4, "Understanding NIS+ Tables and Information" describes the format required for a text file to be transferred into its corresponding NIS+ table. Local `/etc` files are usually formatted properly. NIS maps from running NIS domains are presumed to be correctly formatted.
- It is recommended that you 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.) You also may want edit four of the copied files, `passwd`, `shadow`, `aliases` and `hosts` for security reasons. (See *User Accounts, Printers, and Mail*

Administration for more information on these files.) 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 as the script's output may otherwise scroll off the screen.

Note – The `nispopulate`(1M) 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 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` (1M) script adds credentials for all members of the NIS+ admin group.

b. Type the following 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` (1M) 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 89 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 table that was currently being populated may be 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 the following:

```
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.

populating ethers table from file /nis+files/ethers...

ethers table done.

populating group table from file /nis+files/group...

group table done.

populating hosts table from file /nis+files/hosts...

hosts table done.

Populating the NIS+ credential table for domain wiz.com.
from hosts table.

dumping hosts table...
loading credential table...

The credential table for domain wiz.com. has been populated.
The password used will be nisplus.

populating networks table from file /nis+files/networks...
networks table done.

populating passwd table from file /nis+files/passwd...
passwd table done.
Populating the NIS+ credential table for domain wiz.com.
from passwd table.

dumping passwd table...
loading credential table...
```

The script is using hosts and passwd information to create the credentials for hosts and users.

```
The credential table for domain wiz.com. has been populated.
The passwd used will be nisplus.

populating protocols table from file /nis+files/protocols...
protocols table done.

populating services table from file /nis+files/services...
services table done.

populating rpc table from file /nis+files/rpc...
rpc table done.

populating netmasks table from file /nis+files/netmasks...
netmasks table done.

populating bootparams table from file /nis+files/bootparams...
bootparams table done.

populating netgroup table from file /nis+files/netgroup...
netgroup table done

populating mail_aliases table from file /nis+files/aliases...
mail_aliases table done.

populating passwd table from file /nis+files/shadow...
passwd 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!
```

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 the following:

```
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.

populating ethers table from corporate.wiz.com NIS(YP) domain...
ethers table done.

populating group table from corporate.wiz.com NIS(YP) domain...
group table done.

populating hosts table from corporate.wiz.com NIS(YP) domain...
hosts table done.

Populating the NIS+ credential table for domain wiz.com.
from hosts table. The passwd used will be nisplus.

dumping hosts table...
loading credential table...

The credential table for domain wiz.com. has been populated.
The passwd used will be nisplus.

populating networks table from corporate.wiz.com NIS(YP)
domain...
networks table done.

populating passwd table from corporate.wiz.com NIS(YP) domain...
passwd table done.

Populating the NIS+ credential table for domain wiz.com.
from passwd table.

dumping passwd table...
loading credential table...
The credential table for domain wiz.com. has been populated.
```

The script is using hosts and passwd information to create the credentials for hosts and users.

```
The passwd used will be nisplus.
populating protocols table from corporate.wiz.com NIS(YP)
domain...
protocols table done.

populating services table from corporate.wiz.com NIS(YP)
domain...
services table done.

populating rpc table from corporate.wiz.com NIS(YP) domain...
rpc table done.

populating netmasks table from corporate.wiz.com NIS(YP)
domain...
parse error: no mask (key #)
netmasks table done.

populating bootparams table from corporate.wiz.com NIS(YP)
domain...
parse error: no value (key)
bootparams table done.

populating netgroup table from corporate.wiz.com NIS(YP)
domain...
netgroup table done.

populating mail_aliases table from corporate.wiz.com NIS(YP)
domain...
mail_aliases 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!
```

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 step) Type the following command to add yourself and other administrators to the root domain's admin group.

Note – This step is only necessary 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 password 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" for more information on initializing users. The group cache also has to expire before the new members will become active.

Use the `nisgrpadm`(1M) 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 `secondadmin`, to the `admin.wiz.Com.` group:

```
rootmaster# nisgrpadm -a admin.wiz.Com.  topadmin.wiz.Com. \
    secondadmin.wiz.Com.
Added "topadmin.wiz.Com." to group "admin.wiz.Com.".
Added "secondadmin.wiz.Com." to group "admin.wiz.Com.".
```

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`(1M) 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.

7

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 using the `nisclient`(1M) 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 network password created by the `nispopulate`(1M) script in the previous example (`nisplus`, the default password)

Note – The `-i` option used in "How to Initialize a New Client Machine" 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 file. See Chapter 5, "Understanding the Name Service Switch" for more information on resolving host names through DNS.

Prerequisites to Running `nisclient`(1M)

Before you can use the `nisclient`(1M) 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 network 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 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 hostname.

```
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, then press Return. This example uses the address `126.141.246.63`.**

```
Type server rootmaster's IP address: 126.141.246.63

setting up the domain information...

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

4. **Type the network password (also known as the Secure-RPC password) only if the network 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 network 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 preceding 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 network password to that user's login password. What actually happens is that the user's password and the network password are bound together. This procedure uses the `nisclient`(1M) script.

Prerequisites to Running `nisclient` *(1M)*

Before you can use the `nisclient`(1M) 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 network password—`nisplus` in this example
- The login password of the user that will become the NIS+ client

▼ How to Initialize an NIS+ User

1. **Type the following while logged in as the user to become an NIS+ client.**

```
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. **Type the network password (Secure-RPC password), which is `nisplus` in this case, and then press Return.**
   The password does not echo on the screen.

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

3. **Type the user's login password and then press Return.**
   The password does not echo on the screen.

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

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

Setting Up NIS+ Servers

Now that the client machines have been initialized, you can change any of them to NIS+ servers but not into root NIS+ servers. Root NIS+ servers are a special type of NIS+ server. See "Setting Up NIS+ Root Servers" 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+ command `rpc.nisd`(1M) instead of an NIS+ script.

Prerequisites to Running `rpc.nisd` *(1M)*

Before you can run `rpc.nisd`(1M):

- 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.

▼ How to Configure a Client as an NIS+ Server

♦ **Perform any of the following to configure a client as a server.**
This step creates a directory with the same name as the server and creates the server's initialization files. They 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.

- To configure a NIS+ server without NIS compatibility:

```
wizclient1# rpc.nisd
```

- To configure a NIS+ server *with* NIS compatibility:

  i. **Edit the** `/etc/init.d/rpc` file on the server.
  Uncomment the whole line containing the string `EMULYP="-Y"`.

  ii. **Type the following as superuser:**

```
wizclient1# rpc.nisd -Y
```

- To configure a NIS+ server *with* NIS compatibility *and* DNS forwarding (needed for SunOS 4.x NIS clients):

  i. **Edit the** `/etc/init.d/rpc` file on the server.
  Uncomment the whole line (remove the # character from the beginning of the line) containing the string `EMULYP="-Y"`.

  ii. **Add** `-B` **to the above line inside the quotes. The line should read:**
  `EMULYP="-Y -B"`

  iii. **Type the following 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 are then going to 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 since 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 upon the size of your tables.

This example shows the machine `wizclient1` being configured as a root replica. This procedure uses the NIS+ script `nisserver`(1M).

Prerequisites to Running `nisserver` *(1M)*

Before you can run `nisserver`(1M) 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`(1M) 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. **Type the following as superuser (root) on the NIS+ domain's root master server to create a root replica.**
   The `-R` option indicates that a replica should be set up. The `-d` option specifies the NIS+ domain name, wiz.com. in this instance. 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" 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`(1M) 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 (YP) 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 "How to Configure a Client as an NIS+ Server" 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. It is not recommended, however, that you have more than a few replicas per domain for overall performance reasons. 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`(1M).

Prerequisites to Running `nisserver` *(1M)*

Before you can run `nisserver`(1M) to create a master server for a new non-root 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`(1M) 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, for example, *newdomain.rootdomain*.

- The client machine name; `wizclient2` in this example
- The superuser password for the parent master server

In the following example, the new non-root 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 Non-Root Domain

1. **Type the following as superuser (root) on the NIS+ domain's root master server to create a new non-root domain master server.**
   The `-M` option indicates that a master server for a new non-root domain should be created. The `-d` option specifies the *new* domain name, 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 non-root domains are created with the same set of default values as root servers. See "How to Create a Root Master Server" for more information on NIS+ group, NIS (YP) 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" 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`(1M) 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
passwd.org_dir.subwiz.wiz.com. created
group.org_dir.subwiz.wiz.com. created
auto_master.org_dir.subwiz.wiz.com. created
auto_home.org_dir.subwiz.wiz.com. created
bootparams.org_dir.subwiz.wiz.com. created
cred.org_dir.subwiz.wiz.com. created
ethers.org_dir.subwiz.wiz.com. created
hosts.org_dir.subwiz.wiz.com. created
mail_aliases.org_dir.subwiz.wiz.com. created
sendmailvars.org_dir.subwiz.wiz.com. created
netmasks.org_dir.subwiz.wiz.com. created
netgroup.org_dir.subwiz.wiz.com. created
networks.org_dir.subwiz.wiz.com. created
protocols.org_dir.subwiz.wiz.com. created
rpc.org_dir.subwiz.wiz.com. created
services.org_dir.subwiz.wiz.com. created
timezone.org_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 7-1 on page 83.

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 2, “Understanding Name Services” 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`(1M) 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`*(1M)*

Before you can run the script `nispopulate`(1M) 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. Chapter 4, "Understanding NIS+ Tables and Information" describes the format required for a text file to be transferred into its corresponding NIS+ table. Local `/etc` files are usually formatted properly. NIS maps from running NIS domains are presumed to be correctly formatted.
- It is recommended that you 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.) You also may want edit four of the copied 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`(1M) 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.

▼ How to Populate Master Server Tables

Since this procedure is essentially the same as the procedure shown in "How to Populate the Root Master Server Tables," 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.

♦ **Perform either step *a* or step *b* to populate the master server tables on the new master server.**
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 as the script's output may otherwise scroll off the screen.

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

```
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)
```

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

```
wizclient2# nispopulate -Y -d subwiz.wiz.com. -h businessmachine
-a 130.48.58.242 -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" for the rest of this script's output.

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 upon 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 is going to remain a

client of the domain above the one it will be serving as a replica. This example only shows you 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."

Prerequisites to Running `nisserver` *(1M)*

Before you can run `nisserver`(1M) 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`(1M) 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

♦ **Type the following as superuser (root) on the NIS+ domain's master server (`wizclient2`) to create a replica.**
The `-R` option indicates that a replica should be set up. The `-d` option specifies the NIS+ domain name, subwiz.wiz.com. in this instance. 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.

```
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)
```

See “How to Create a Root Replica” 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`(1M) 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” does not set up an NIS+ client to resolve hostnames requiring DNS. You need to explicitly include DNS for clients in their name service switch file. See Chapter 5, “Understanding the Name Service Switch” 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 only shows you 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.”

Prerequisites to Running `nisclient`*(1M)*

Before you can use the `nisclient`(1M) 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 network 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 as superuser 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 hostname.

```
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)
```

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

Initializing Subdomain NIS+ Client Users

You use the same procedure (`nisclient`(1M)) 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 only shows you 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."

Prerequisites to Running `nisclient` *(1M)*

Before you can use the `nisclient`(1M) 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 network password—`nisplus` in this example
- The login password of the user that will become the NIS+ client

▼ How to Initialize an NIS+ Subdomain User

♦ **Type the following while logged in as the user to become an NIS+ client.**

```
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" for the rest of this script's output.

Summary of Commands for the Sample NIS+ Namespace

Table 7-3 on page 123 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 7-1 on page 83 for a diagram of the sample namespace.

Table 7-3 Sample NIS+ Namespace Command Lines Summary

Command Line	Purpose
#**setenv PATH $PATH:/usr/lib/nis** or #**PATH=$PATH:/usr/lib/nis; export PATH**	Set environment path to include /usr/lib/nis—C shell or Bourne shell
rootmaster# **nisserver -r -d wiz.com.**	Create root master server for wiz.com. domain
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**	Populate the root master server's NIS+ tables—from files or from NIS maps
rootmaster# **nisgrpadm -a admin.wiz.Com. topadmin.wiz.Com. \ secondadmin.wiz.Com.**	Add additional members to the admin group (2)
rootmaster# **nisping -C wiz.Com.**	Make a checkpoint of the NIS+ database
wizclient1# **nisclient -i -d wiz.com. -h rootmaster**	Initialize a NIS+ client machine in the wiz.com. domain
wizclient1user1prompt% **nisclient -u**	Initialize user as a NIS+ client
wizclient1#**rpc.nisd** or wizclient1#**rpc.nisd -Y** or wizclient1#**rpc.nisd -Y -B**	Convert NIS+ client to NIS+ server, without or with NIS compatibility or with NIS and DNS.
rootmaster# **nisserver -R -d wiz.com. -h wizclient1**	Create a root replica
rootmaster# **nisserver -M -d subwiz.wiz.com. -h wizclient2**	Convert a server to a non-root master server of the subwiz.wiz.com. domain
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**	Populate the new master server's NIS+ tables—from files or from NIS maps
wizclient2# **nisserver -R -d subwiz.wiz.com. -h wizclient3**	Create a master server replica
subclient1# **nisclient -i -d subwiz.wiz.com. -h wizclient2**	Initialize a NIS+ client in the subwiz.wiz.com. domain
subclient1user2prompt% **nisclient -u**	Initialize user as a NIS+ client

7

Part 3 — DNS Concepts

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

DNS Structure 8

Domain Name Service (DNS) is an application layer protocol that is part of the standard TCP/IP protocol suite. Specifically, DNS is a *naming* service; it obtains and provides information about hosts on a network.

Domain Name Service performs naming between hosts *within* your local administrative domain and *across* domain boundaries. It is distributed among a set of servers, commonly known as "name servers," each of which implements DNS by running a daemon called `in.named`.

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

On the client's side, DNS is implemented through the "resolver." The resolver is neither a daemon nor a particular program; rather, it is a library compiled into 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.

DNS Clients

A name server running `in.named` can also run the resolver; therefore, there can be two kinds of clients:

- client-only
- client/server

A client-only client does not run the `in.named` daemon; instead, it consults the resolver, which provides a list of possible name serving machines to which queries should be directed.

A client/server is a machine that uses the domain name service provided by `in.named` to resolve a user's queries. This type of machine may answer its own queries, but not necessarily.

DNS Servers

You implement DNS for a zone on one or more servers. A zone can include two master servers, and may or may not include other servers.

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. It is recommended that the data corresponding to any given zone 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.

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 up `in.named`. The primary server may also delegate authority to other servers in its zone, as well as to servers outside of it.

The "secondary" master server is a name server that 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 data.

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

A server at the root level of the network is called a "root domain name server." On the Internet, root domain name servers are maintained by the NIC. If a network is not connected to the Internet, primary and secondary name servers must be set up and administered 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 (`ttl`) 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 who have the authority for the information needed. But the caching-only server does not maintain any authoritative data itself.

8

Setting Up DNS Clients 9

Setting up DNS on a client involves two tasks:

- creating the `resolv.conf` file
- modifying the `/etc/nsswitch.conf` file

If you are setting up DNS on a name server, you need to complete these steps, in addition to setting up boot and data files. The server tasks are described in Chapter 10, “Setting Up DNS Servers”.

Creating `resolv.conf`

The domain name server uses several files to load its database. At the resolver level, it needs a file (called `/etc/resolv.conf`) listing the addresses of the servers where it can obtain the information needed. Whenever the resolver has to find the address of a host (or the name corresponding to an address) it builds a query package and sends it to the name servers it knows of (from `/etc/resolv.conf`). The servers either answer the query locally or use the services of other servers and return the answer to the resolver.

The `resolv.conf` file is read by the resolver to find out 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 names servers for information. Every DNS client system on your network must have a `resolv.conf` file in its `/etc` directory.

The first line of the file lists the domain name in the form:

```
domain domain_name
```

where *domain_name* is the name registered with the NIC (Network Information Center, an agency of the National Science Foundation). Succeeding lines list the IP addresses that the resolver should consult to resolve queries. IP address entries have the form:

```
nameserver IP_address
```

Below is a sample `resolv.conf` file.

```
; Sample resolv.conf file
domain Podunk.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 `/etc/nsswitch.conf`

To use DNS as the source of hostname information, follow the directions for enabling an NIS+ client to use DNS in *Name Services Administration Guide.*

Setting Up DNS Servers 10

Because every name server is a client of other name servers, you must complete the steps involved in setting up DNS on a client when you set up a machine to be a name server. These steps are:

- Creating the file `resolv.conf`.
- Modifying the `/etc/nsswitch.conf` file.

Instructions for completing these steps appear in Chapter 9, “Setting Up DNS Clients”. Once you complete these steps, you also need to do the following to set up DNS on a machine that is to be a name server:

- Creating the boot and data files that the name server, `in.named`, needs.

Instructions for completing these procedures 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.

If your local network is not on 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 at the end of this chapter.

Creating Boot and Data Files

In addition to the daemon `in.named`, DNS on a name server consists of a boot file and local data files. The default location of the boot file is `/etc/named.boot`. Common names for the local data files are `named.ca`,

named.local, hosts, and hosts.rev. In the descriptions of these files that follow, these names are used. However, you can name these files whatever you wish. In fact, it is suggested that you use names other than the ones used in this guide, to avoid confusion with files with similar names.

named.boot

The boot file named.boot establishes the server as a primary, a secondary, or a 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.

named.ca

named.ca establishes the names of root servers and lists their addresses. If you are connected to the Internet, named.ca lists the Internet name servers; otherwise, it lists the root domain name servers for your local network.

in.named 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.

To avoid confusion with the caching process, which has very little to do with this, you may want to call your file something like named.root or boot.root.

hosts

The hosts file contains all the data about the machines in the local zone. The name of this file is specified in the boot file. To avoid confusion with /etc/hosts, name the file something other than hosts. In Code Example 10-1 on page 135, the file is called puhosts.

hosts.rev

The hosts.rev file specifies a zone in the IN-ADDR.ARPA domain (the special domain that allows inverse mapping). The name of this file is specified in the boot file. In the illustration, the file is called puhosts.rev.

named.local

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. As with all other files, it can be called something other than the name used in this guide.

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 and secondary master servers 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.

The following is a sample boot file for a primary server:

Code Example 10-1 Sample Master Boot File

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

The entries in the file are explained below.

directory

The following 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. It 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 it must be full path names.

cache

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`).

primary

To set up a primary server, you need to create a file that contains all the authoritative data for the zone. Then you 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      Podunk.Edu.        puhosts
```

The first field designates the server as primary for the zone stated in the second field. The third field is the name of the file from which authoritative data is read.

The lines:

```
primary        32.128.in-addr.arpa     puhosts.rev
primary        0.0.127.in-addr.arp     named.local
```

indicate that the server is also a primary server for `32.128.in-addr.arpa` (that is, the reverse address domain for Podunk.Edu) and `0.0.127.in-addr.arpa` (that is, the local host loopback), and data for them is to be found, respectively, in the files `puhosts.rev` and `named.local`.

The following is a sample boot file for a secondary server in the same domain as the above primary server:

```
;
; Sample named.boot file for Secondary Master Name Server
;
; type         domain                                    source file or host
;
directory /var/named
cache          .   named.ca
secondary      Podunk.Edu.  128.32.0.4 128.32.0.10   128.32.136.22         puhosts.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      Podunk.Edu.128.32.0.4   128.32.0.10   128.32.136.22   puhosts.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, and that it is to get its data from the listed servers (usually the primary server followed by one or more secondary ones); attempts 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 server is started, data are loaded from the backup file, if it exists, and then one of the servers is consulted to check whether the data is still up to date.

This ability to specify multiple secondary 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; it is the mixture of entries in the boot file that determines it.

The interpretation of the other “secondary line” is similar to the above. Note also that although this is a secondary server for the domain Podunk.Edu and 32.128.in-addr.arpa, this is a primary server for 0.0.127.in-addr.arpa. (the local host).

The following is a sample boot file for a 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 above, 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. 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. Following this section is a discussion of the Standard Resource Record Format, including an explanation of each Resource Record relevant to the DNS data files.

The `hosts` file contains all the data about all the machines in your zone, including server names, 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, HINFO, CNAME, WKS, MX, MB, MR, MG. The file also include the SOA record, which indicates the start of a zone and includes the name of the host on which the `hosts` data file resides. The example on the following page shows a sample `hosts` file.

```
;                       sample          hosts       file
@                       IN              SOA         ourfox.Sample.Edu. kjd.monet.Sample.Edu.
(
                                                    1.1 ; Serial
                                                    10800 ; Refresh
                                                    1800 ; Retry
                                                    3600000 ; Expire
                                                     86400 ) ; Minimum
                        IN              NS          ourarpa.Sample.Edu.
                        IN              NS          ourfox.Sample.Edu.
ourarpa                 IN              A           128.32.0.4
                        IN              A           10.0.0.78
                        IN              HINFO       3B2 UNIX
arpa                    IN              CNAME       ourarpa
ernie                   IN              A           t128.32.0.6
                        IN              HINFO       3B2 UNIX
ourernie                IN              CNAME       ernie
monet                   IN              A           128.32.7
                        IN              A           128.32.130.6
                        IN              HINFO       Sun-4/110 UNIX
ourmonet                IN              CNAME       monet
ourfox                  IN              A           10.2.0.78
                        IN              A           128.32.0.10
                        IN              HINFO       Sun-4/110 UNIX
                        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 )
fox                     IN              CNAME       ourfox
toybox                  IN              A           128.32.131.119
                        IN              HINFO       3B2 UNIX
toybox                  IN              MX          0 monet.Sample.Edu
```

The named.local file sets up the local loopback interface for your name server. It needs to 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 by the PTR record. The file also needs to 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. Here is a sample named.local file:

```
; sample named.local file
@   IN      SOA     ourhost.Podunk.Edu. kjd.monet.Podunk.Edu.
           (
            1                   ; Serial
            3600                ; Refresh
            300                 ; Retry
            3600000             ; Expire
            3600 )              ; Minimum
    IN      NS                  ourhost.Podunk.Edu.
1   IN      PTR                 localhost.
```

hosts.rev is the file that sets up inverse mapping. It needs to 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 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.

Here is a sample `hosts.rev` file:

```
; sample hosts.rev file
@       IN  SOA ourhost.Podunk.Edu. kjd.monet.Podunk.Edu.
              (
                1.1    ; Serial
                3600           ; Refresh
                300            ; Retry
                3600000 ; Expire
                3600 )         ; Minimum
            IN  NS             ourarpa.Podunk.Edu.
            IN  NS             ourhost.Podunk.Edu.
4.0         IN  PTR            ourarpa.Podunk.Edu.
6.0         IN  PTR            ernie.Podunk.Edu.
7.0         IN  PTR            monet.Podunk.Edu.
10.0        IN  PTR            ourhost.Podunk.Edu.
6.130       IN  PTR            monet.Podunk.Edu.
```

The `named.ca` file contains the names and addresses of the root servers. Server names are indicated in the record NS and addresses in the record A. You need to add an NS record and an A record for each root server you want to include in the file.

The following is a 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
```

Standard Resource Record Format

In the Standard Resource Record Format, each line of a data file is a record called a Resource Record (RR), containing 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 braces), and the contents of the last vary according to the `Record Type` field.

name

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 one, in which case the current domain is appended to it.

ttl

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 resource record discussed below.

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, 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, and 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, and 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.

class

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

type

The fourth field states the type of the resource record. There are many types of RR's; the most commonly used types are discussed in "Resource Record Types" on page 146.

RR data

The contents of the 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, and you are advised to be consistent in your use of lower and upper case.

Special Characters

The following characters have special meanings:

.

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.

()

Use parentheses to group data that crosses a line. In effect, line terminations are not recognized within parentheses.

;

Semicolon starts a comment; the remainder of the line is ignored.

****

An asterisk signifies wildcarding.

Most resource records have the current origin appended to names if they are not terminated by a "." This is useful for appending the current domain name to the data, such as machine names, but may cause problems where you do not want this to happen. A good rule of thumb is to 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

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, for example:

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

The line is interpreted as a request to load the file `/etc/named/data/mailboxes` 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

Some of the most commonly used types of RR's are listed in Table 10-1.

Table 10-1 Commonly Used Resource Record Types

Type	Description
SOA	Start of Authority
NS	Name Server
A	Internet Address
CNAME	Canonical Name (nickname)
HINFO	Host Information
WKS	Well Known Services
PTR	Pointer
MX	Mail Exchanger

Code Example 10-2 on page 147 shows an example of a `hosts` file. It is presented here for illustration purposes only. Explanations of each field follow the code example.

Code Example 10-2 Sample hosts File

```
; sample hosts file
@                       IN      SOA         ourfox.Sample.Edu. kjd.monet.Sample.Edu. (
                                            1.1 ; Serial
                                            10800 ; Refresh
                                            1800 ; Retry
                                            3600000 ; Expire
                                            86400 ) ; Minimum
                        IN      NS          ourarpa.Sample.Edu.
                        IN      NS          ourfox.Sample.Edu.
                        IN      A           128.32.0.4
ourarpa                 IN      A           10.0.0.78
                        IN      HINFO       3B2 UNIX
arpa                    IN      CNAME       ourarpa
ernie                   IN      A           128.32.0.6
                        IN      HINFO       3B2 UNIX
ourernie                IN      CNAME       ernie
monet                   IN      A           128.32.7
                        IN      A           128.32.130.6
                        IN      HINFO       Sun-4/110 UNIX
ourmonet                IN      CNAME       monet
ourfox                  IN      A           10.2.0.78
                        IN      A           128.32.0.10
                        IN      HINFO       Sun-4/110 UNIX
                        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 nntplink chargen ftp
                                            auth time whois mtp
                                            pop rje finger smtp
                                            supdup hostnames
                                            domain
                                            nameserver )
fox                     IN      CNAME       ourfox
toybox                  IN      A           128.32.131.119
                        IN      HINFO       3B2 UNIX
toybox                  IN      MX          0 monet.Sample.Edu
```

SOA - Start of Authority

The following is the format of a Start of Authority resource record:

```
name   {ttl}      {class}    SOA    origin    person in charge
                                 (
                                    serial
                                    refresh
                                    retry
                                    expire
                                    minimum )
```

The Start of Authority (SOA) record designates the start of a zone. The zone ends at the next SOA record.

name

This field indicates the name of the zone. In the Code Example 10-2 on page 147, `@` indicates the current zone or origin.

IN

This field is the address 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.

Person_in_charge

This field is the mailing address for the person responsible for the name server.

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.

Note that the name server cannot handle numbers over 9999 after the decimal point.

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.

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. The following is a sample SOA resource record:

```
;name      {ttl}      class      SOA           origin     person in charge
 @                     IN         SOA        ourfox.Sample.Edu.kjd.monet.Sample.Edu.(
                                  1.1        ;Serial
                                  10800      ;Refresh
                                  3600       ;Retry
                                  432000     ;Expire
                                  86400)     ;Minimum
```

NS - Name Server

The following is the format of an NS resource record:

```
{name}     {ttl}      class      NS         Name-server name
```

The Name Server record (NS) 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. The following is a sample NS resource record:

```
;{name}    {ttl}      class      NS         Name-server name
                      IN         NS         ourarpa.Sample.Edu.
```

A - Address

The following is the format of an A resource record:

```
{name}     {ttl}      class      A          address
```

The Address record (A) lists the address for a given machine. The name field is the machine name, and the address is the IP address. One A record should exist for each address of the machine (in other words, gateways should be listed twice, once for each address).

```
;{name}    {ttl}      class      A          address
ourarpa                IN          A         128.32.0.4
                       IN         A          10.0.0.78
```

HINFO - Host Information

The following is the format of a HINFO resource record:

```
{name}     {ttl}      class      HINFO      Hardware       OS
```

The Host Information resource record (HINFO) 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. The following is a sample HINFO resource record:

```
;{name}    {ttl}      class      HINFO      Hardware       OS
                      IN         HINFO      Sun-3/280      UNIX
```

WKS - Well Known Services

The following is the format of a WKS resource record:

```
{name}     {ttl}      class      WKS        address        protocollist of services
```

The Well Known Services record (WKS) 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. The following is an example of a WKS resource record:

```
;{name}    {ttl}      class      WKS        address        protocollist 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)
```

CNAME - Canonical Name

The format of a CNAME resource record is:

```
 nickname      {ttl}      class      CNAME      Canonical_name
```

The Canonical Name resource record (CNAME) 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. The following is a sample CNAME resource record:

```
;nickname     {ttl}      class      CNAME      Canonical_name
ourmonet                 IN         CNAME      monet
```

PTR - Domain Name Pointer

The following is the format for a PTR resource record:

```
special name          {ttl}      class      PTR_real_name
```

A Pointer record (PTR) allows special names to point to some other location in the domain. 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 below set up reverse pointers for the special IN-ADDR.ARPA domain.

```
;special name                              {ttl}      class  PTR real name
  7.0                                                 IN     PTR monet.Podunk.Edu.
  2.2.18.128.in-addr.arpa                             IN     PTR blah.junk.COM.
```

MX - Mail Exchanger

The following is the format for an MX resource record:

```
name                    {ttl}          class      MX   preference_ value  mailer_exchanger
```

The Mail Exchanger (MX) 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 10-3 on

page 153, 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, they 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 10-3, 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 10-3 MX 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.
```

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 up, it 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 up 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 10-2 Domain Configuration of Imaginary Network—Class C

name	**number**
junk	223.100.100
widget	223.100.101
zap	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 the Domain Name Service 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 10-3 Domain Configuration of Imaginary Network—`junk` Zone

hostname	function	address
junk	primary	223.100.100.1
widget	secondary	223.100.101.1
zap	secondary	223.100.102.1
	hosts	223.100.100.2-80

Table 10-4 Domain Configuration of Imaginary Network—`widget` Zone

hostname	function	address
widget	primary	223.100.101.1
junk	secondary	223.100.100.1
zap	secondary	223.100.102.1
	hosts	223.100.101.2-110

Table 10-5 Domain Configuration of Imaginary Network—`zap` Zone

hostname	function	address
zap	primary	223.100.102.1
junk	secondary	223.100.100.1
widget	secondary	223.100.101.1
	hosts	223.100.102.2-156

The following are the boot files for the three servers in the network:

```
;
;   Boot file for server junk
directory      /var/named
cache          .                              named.root
primary        junk.COM                       junk.zone
primary        100.100.223.in-addr.arpa       junk.revzone
primary        0.0.127.in-addr.arpa           named.local
secondary      widget.junk.COM                223.100.101.1 223.100.102.1 widget.zone
secondary      zap.junk.COM                   223.100.101.1 223.100.102.1 zap.zone
secondary      101.100.223.in-addr.arpa       223.100.101.1 widget.rev
secondary      102.100.223.in-addr.arpa       223.100.102.1 zap.rev
```

```
;
;   Boot file for server widget
directory      /var/named
cache          .                              named.root
primary        widget.junk.COM                widget.zone
primary        101.100.223.in-addr.arpa       widget.revzone
primary        0.0.127.in-addr.arpa           named.local
secondary      junk.COM                       223.100.100.1 223.100.102.1 junk.zone
secondary      zap.junk.COM                   223.100.100.1 223.100.102.1 zap.zone
secondary      100.100.223.in-addr.arpa       223.100.100.1 junk.rev
secondary      102.100.223.in-addr.arpa       223.100.102.1 zap.rev
```

```
;
;   Boot file for server zap
directory      /var/named
cache          .                              named.root
primary        zap.junk.COM                   zap.zone
primary        102.100.223.in-addr.arpa       zap.revzone
primary        0.0.127.in-addr.arpa           named.local
secondary      junk.COM                       223.100.100.1 223.100.102.1 junk.zone
secondary      widget.junk.COM                223.100.100.1 223.100.101.1 widget.zone
secondary          100.100.223.in-addr.arpa   223.100.100.1 junk.rev
secondary      101.100.223.in-addr.arpa       223.100.101.1 widget.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 nameserver.

```
;
; resolv.conf file for server junk
;
domain         junk.COM
nameserver     127.0.0.1
nameserver     223.100.101.1
nameserver     223.100.102.1
```

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

```
;
; resolv.conf file for a host in zone widget.junk not running named
;
domain         widget.junk.COM
nameserver     223.100.100.1
nameserver     223.100.101.1
nameserver     223.100.102.1
```

```
;
; resolv.conf file for a host in zone zap.junk not running named
;
domain         zap.junk.COM
nameserver     223.100.100.1
nameserver     223.100.101.1
nameserver     223.100.102.1
```

The following are sample `named.local` files:

```
;
; named.local for server junk
;
@          IN  SOA     junk.COM.      ralph.sysad.zap.junk.COM.
(
               1.1            ;Serial
               10800          ;Refresh
               3600           ;Retry
               432000         ;Expire
               86400)         ;Minimum
           IN  NS             junk.COM.
1          IN  PTR            localhost.
```

```
;
; named.local for server widget
;
@          IN  SOA     widget.junk.COM.
ralph.sysad.zap.junk.COM.(
               1.1            ;Serial
               10800          ;Refresh
               3600           ;Retry
               432000         ;Expire
               86400)         ;Minimum
           IN  NS             widget.junk.COM.
1          IN  PTR            localhost.
```

```
;
; named.local for server zap
;
@          IN  SOA     zap.junk.COM.  ralph.sysad.zap.junk.COM.
(
               1.1            ;Serial
               10800          ;Refresh
               3600           ;Retry
               432000         ;Expire
               86400)         ;Minimum
           IN  NS             zap.junk.COM.
1          IN  PTR            localhost.
```

The `hosts` file for server `junk`, followed by its `$INCLUDE` file, appears in the following code sample:

Code Example 10-4 The `hosts` File for Sample Server `junk`

```
;
; junk zone hosts file for server junk
;
@                           IN          SOA             junk.COM.ralph.sysad.zap.junk.COM.
(
                                        1.1                 ;Serial
                                        10800               ;Refresh
                                        3600                ;Retry
                                        432000              ;Expire
                                        86400)              ;Minimum
                            IN          NS                  junk.COM.
                            IN          NS                  widget.junk.COM.
                            IN          NS                  zap.junk.COM.
widget.junk.COM.            IN          NS                  widget.junk.COM.
                            IN          NS                  junk.COM.
                            IN          NS                  zap.junk.COM.
zap.junk.COM.               IN          NS                  zap.junk.COM.
                            IN          NS                  junk.COM.
                            IN          NS                  widget.junk.COM.
junk.COM.                   IN          MX                  10 junk.COM.
*.junk.COM.                 IN          MX                  10 junk.COM.
; junk.COM hosts
$INCLUDE /var/named/hosts/junk
; hosts in junk zone as listed in /var/named/hosts/junk
junk                        A           223.100.100.1
                            A           10.1.0.56
                            MX          10                  junk.COM.
widget                      A           223.100.101.1
                            HINFO       "Sun 3/180"         Unix
                            MX          10 junk.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)
```

Code Example 10-5 The `hosts` File for Sample Server `junk` *(Continued)*

```
zap                             A          223.100.102.1
                                HINFO      Sun-4/110Unix
                                MX         10 junk.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)
lazy                            A          223.100.100.2
                                HINFO      "Sun 3/50"Unix
                                MX         10 junk.COM.
crazy                           A          223.100.100.3
                                HINFO      3B2                Unix
                                MX         10 junk.COM.
hazy                            A          223.100.100.4
                                HINFO      3B2                Unix
                                MX         10 junk.COM.
; all other hosts follow, up to 223.100.100.80
```

The following is the `hosts` file for server `widget`, followed by its `$INCLUDE` file:

```
;
; widget zone hosts file for server widget
;
@                               IN         SOA      widget.junk.COM.
ralph.sysad.zap.junk.COM. (
                                           1.1                ;Serial
                                           10800              ;Refresh
                                           3600               ;Retry
                                           432000             ;Expire
                                           86400)             ;Minimum
                                IN         NS                 widget.junk.COM.
                                IN         NS                 junk.COM.
                                IN         NS                 zap.junk.COM.
; junk.COM hosts
$INCLUDE /var/named/hosts/widget
```

```
 ; hosts in widget zone as listed in /var/named/hosts/widget
widget                  A               223.100.101.1
                        HINFO               "Sun 3/180"         Unix
                        MX                  10 junk.COM.
whatsit                 CNAME               widget.junk.COM
smelly                  A                   223.100.101.2
                        HINFO               3B2                 Unix
                        MX                  10 junk.COM.
stinky                  A                   223.100.101.3
                        HINFO               3B2                 Unix
                        MX                  10 junk.COM.
dinky                   A                   223.100.101.4
                        HINFO               "Sun 3/160"         Unix
                        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 junk.COM.
; all other hosts follow, up to 223.100.101.110
```

The following is the sample file of reverse addresses for hosts in the zone `junk`. 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:

```
; reverse address file for server junk, in /var/named/junk.revzone
100.100.223.in-addr.arpa.IN                 SOA         junk.COM.ralph.sysad.zap.junk.COM.
(
                                            1.1         ;Serial
                                            10800       ;Refresh
                                            3600        ;Retry
                                            432000      ;Expire
                                            86400)      ;Minimum
                      IN                      NS            junk.COM.
1                                           PTR         junk.COM.
2                                           PTR         lazy.junk.COM.
3                                           PTR         crazy.junk.COM.
4                                           PTR         hazy.junk.COM.
; all other hosts follow, up to 223.100.100.80
```

The reverse address files for servers `widget` and `zap` 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 – It is strongly recommended that the root domain server name be the primary name server for all top-level domains in the network.

Pre-Setup Worksheets A

Use the worksheets on the following pages to record planning information prior to NIS+ setup. *Name Services Administration Guide* provides a sample copy in the chapters that describe how to set up an NIS+ namespace manually.

A

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?**
	Groups	**N**	**O**	**G**	**W**	**Description**

Domain: _____ (*Continued*)

Rights (*cont*)	**Types of Objects**	**Category & Rights**				
	Tables	**N**	**O**	**G**	**W**	**Notes**
	bootparams					
	hosts					
	passwd					
	cred					
	group					
	netgroup					
	aliases					
	timezone					
	networks					
	netmasks					
	ethers					
	services					
	protocols					
	rpc					
	auto_home					
	auto_,master					

A

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?**
	Groups	**N**	**O**	**G**	**W**	**Description**

Domain: (*Continued*)

Rights (*cont*)	**Types of Objects**	**Category & Rights**				
	Tables	**N**	**O**	**G**	**W**	**Notes**
	bootparams					
	hosts					
	passwd					
	cred					
	group					
	netgroup					
	aliases					
	timezone					
	networks					
	netmasks					
	ethers					
	services					
	protocols					
	rpc					
	auto_home					
	auto_,master					

A

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?**
	Groups	**N**	**O**	**G**	**W**	**Description**

Domain: _____ (*Continued*)

Rights (*cont*)	**Types of Objects**	**Category & Rights**				
	Tables	**N**	**O**	**G**	**W**	**Notes**
	bootparams					
	hosts					
	passwd					
	cred					
	group					
	netgroup					
	aliases					
	timezone					
	networks					
	netmasks					
	ethers					
	services					
	protocols					
	rpc					
	auto_home					
	auto_,master					

A

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?**
	Groups	**N**	**O**	**G**	**W**	**Description**

Domain: _____ (*Continued*)

Rights (*cont*)	**Types of Objects**	**Category & Rights**				
	Tables	**N**	**O**	**G**	**W**	**Notes**
	bootparams					
	hosts					
	passwd					
	cred					
	group					
	netgroup					
	aliases					
	timezone					
	networks					
	netmasks					
	ethers					
	services					
	protocols					
	rpc					
	auto_home					
	auto_,master					

A

Glossary

access rights

The permissions assigned to classes of NIS+ principals that determine what operations they can perform on NIS+ objects: Read, Modify, Create or Destroy.

authentication

The determination of whether an NIS+ server can identify the sender of a request for access to the NIS+ namespace. Authenticated requests are divided into the authorization categories of Owner, Group, and World. Unauthenticated requests—the sender is unidentified, are placed in the Nobody category. Whether or not such a request is granted depends upon the access rights given to a particular category.

authorization

The determination of the access rights of a particular category of authenticated user. The categories are Owner, Group, World and Nobody. The possible rights a category could have are to read, modify, create and destroy an NIS+ object.

cache manager

The program that manages the local caches of NIS+ clients (NIS_DIR_CACHE), which are used to store location information about the NIS+ servers that support the directories most frequently used by those clients, including transport addresses, authentication information, and a time-to-live value.

client

(1) In the client-server model for file systems, the client is a machine that remotely accesses resources of a compute server, such as compute power and large memory capacity. (2) In the client-server model for window systems, the client is an *application* that accesses windowing services from a "server process." In this model, the client and the server can run on the same machine or on separate machines.

client-server model

A common way to describe network services and the model user processes (programs) of those services. Examples include the name-server / name-resolver paradigm of the *domain name system (DNS)* and file-server / file-client relationships such as *NFS* and diskless hosts. See also *client*.

coldstart file

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

credentials

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

data encrypting key

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

data encryption standard (DES)

A commonly used, highly sophisticated algorithm developed by the U.S. National Bureau of Standards for encrypting and decrypting data. See also *SUN-DES-1*.

decimal dotted notation

The syntactic representation for a 32-bit integer that consists of four 8-bit numbers written in base 10 with periods (dots) separating them. Used to represent IP addresses in the Internet as in: 192.67.67.20.

DES

See *data encryption standard (DES)*.

directory cache

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

DNS

See domain name system.

DNS-forwarding

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

DNS zones

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

DNS zone files

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

domain

(1) In the Internet, a part of a naming hierarchy. Syntactically, an Internet domain name consists of a sequence of names (labels) separated by periods (dots). For example, “tundra.mpk.ca.us.” (2) In International Organization for Standardization’s open systems interconnection (OSI), “domain” is generally used as an administrative partition of a complex distributed system, as in MHS private management domain (PRMD), and directory management domain (DMD).

domain name

The name assigned to a group of systems on a local network that share administrative files. The domain name is required for the network information service database to work properly. Se also *domain*.

domain name system (DNS)

The network information service used in the Internet.

encryption key

See *data encrypting key.*

GID

See group ID.

group

A collection of users who are referred to by a common name. Determines a user’s access to files. There are two types of groups: default user group and standard user group.

group ID

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

indexed name

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

internet

A collection of networks interconnected by a set of routers that enable them to function as a single, large virtual network.

Internet

(Note the capital "I") The largest internet in the world consisting of large national backbone nets (such as *MILNET*, NSFNET, and *CREN*) and a myriad of regional and local campus networks all over the world. The Internet uses the Internet protocol suite. To be on the Internet the user must have IP connectivity, for example, be able to *Telnet* to—or *ping*—other systems. Networks with only e-mail connectivity are not actually classified as being on the Internet.

Internet address

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

IP

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

IP address

A unique number that identifies each host in a network.

key (column)

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

key, encrypting

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

key server

A Solaris process that stores private keys.

mail exchange records

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

mail hosts

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

master server

The server that maintains the master copy of the network information service database.

MIS

management information system

name resolution

The process of translating workstation/user names to addresses.

name server

Servers which run the DNS software and store the names and addresses of the workstations in the domain.

name service switch

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

namespace

(1) DNS namespace—A collection of networked workstations that use the DNS software. (2) NIS namespace—A collection of *non*-hierarchical network information used by the NIS software. (3) NIS+ namespace—A collection of hierarchical network information used by the NIS+ software.

network information service (NIS)

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

network information service plus (NIS+)

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

network mask

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

NIS-compatibility mode

A configuration of NIS+ that allows NIS clients to have access to the data stored in NIS+ tables. When in this mode, NIS+ servers can answer requests for information from both NIS and NIS+ clients.

NIS domain

A master set of *network information service (NIS)* maps maintained on the NIS master server and distributed to that server's NIS slaves.

NIS maps

A file used by NIS that holds information of a particular type, for example, the password entries of all users on a network or the names of all host machines on a network. Programs that are part of the NIS service query these maps. See also *network information service (NIS).*

NIS+ object

The tables which store the NIS+ data when queried by NIS+ clients.

NIS+ principal

A client user or a client workstation whose credentials have been stored in the namespace. Any user or machine that can generate a request to a NIS+ server.

NIS+ tables

Database-like entities that maintain information about NIS+ entry objects on a local area network. The format in which NIS+ data is stored. NIS+ provides sixteen predefined or system tables. Each table stores a different type of information.

NIS+ transaction log

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

principal

See NIS+ principal

private key

The private component of a pair of mathematically generated numbers, which, when combined with a private key, generates the DES key. The DES key in turn is used to encode and decode information. The public key of the sender is only available to the owner of the key. Every user or machine has their own public and private key pair.

public key

The public component of a pair of mathematically generated numbers, which, when combined with a private key, generates the DES key. The DES key in turn is used to encode and decode information. The public key is available to all users and machines. Every user or machine has their own public and private key pair.

populate tables

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

remote procedure call (RPC)

An easy and popular paradigm for implementing the client-server model of distributed computing. A request is sent to a remote system to execute a designated procedure, using arguments supplied, and the result is returned to the caller.

replica

NIS+ server that is a duplicate copy of the principal or master NIS+ server database. Replicas run NIS+ server software and maintain copies of NIS+ tables. A replica server increases the availability of NIS+ services.

reverse resolution

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

root domain

The topmost domain in a hierarchical namespace.

RPC

See *remote procedure call (RPC)*.

server

(1) In the *client-server model* for file systems, the server is a machine with compute resources (and is sometimes called the compute server), and large memory capacity. Client machines can remotely access and make use of these resources. In the client-server model for window systems, the server is a process that provides windowing services to an application, or “client process.” In this model, the client and the server can run on the same machine or on separate machines. (2) A *daemon* that actually handles the providing of files.

slave server

(1) A server system that maintains a copy of the *network information service (NIS)* database. It has a disk and a complete copy of the operating system. (2) Slave servers are called replicas in NIS+.

subnet

A working scheme that divides a single logical network into smaller physical networks to simplify routing.

table

A display of data in rows and columns.

TCP

See *transport control protocol (TCP).*

TCP/IP

Acronym for transport control protocol/interface program. The protocol suite originally developed for the Internet. It is also called the *Internet* protocol suite. SunOS networks run on TCP/IP by default.

transport control protocol (TCP)

The major transport protocol in the Internet suite of protocols providing reliable, connection- oriented, full-duplex streams. Uses IP for delivery. See *TCP/IP.*

Index

Symbols

A

C

D

E

F

G

H

O

P

R

S

T

U

V

Y