System Administration Guide: Naming and Directory Services (NIS+)

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–4558–10
January 2005

Adobe PostScript

041130@10536

Contents

Figures

Preface

Solaris Administration Guide: Naming and Directory Services (NIS+) describes the setup, configuration, and administration of the NIS+ network name service with the Solaris™ 10 Operating System (Solaris OS). This manual is part of the Solaris 10 System and Network Administrator collection.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release. For more information, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and visit `http://www.sun.com/directory/nisplus/transition.html`.

Note – This Solaris™ release supports systems that use the SPARC® and x86 families of processor architectures: UltraSPARC®, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems appear in the *Solaris 10 Hardware Compatibility List* at `http://www.sun.com/bigadmin/hcl`. This document cites any implementation differences between the platform types.

In this document the term "x86" refers to 64-bit and 32-bit systems manufactured using processors compatible with the AMD64 or Intel Xeon/Pentium product families. For supported systems, see the *Solaris 10 Hardware Compatibility List*.

Who Should Use This Book

This manual is written for experienced system and network administrators.

Although this book introduces networking concepts relevant to Solaris naming and directory services, it explains neither the networking fundamentals nor the administration tools in the Solaris OS.

How This Book Is Organized

This manual is divided into parts according to the respective naming and directory services.

About Naming and Directory Services: Part I

NIS+ Setup and Configuration: Part II

NIS+ Administration: Part III

NIS+ Error Messages: Appendix A

How the System Administration Volumes Are Organized

Here is a list of the topics that are covered by the volumes of the System Administration Guides.

Book Title	Topics
System Administration Guide: Basic Administration	User accounts and groups, server and client support, shutting down and booting a system, managing services, and managing software (packages and patches)
System Administration Guide: Advanced Administration	Printing services, terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems
System Administration Guide: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data

Book Title	Topics
System Administration Guide: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, Solaris IP filter, Mobile IP, IP network multipathing (IPMP), and IPQoS
System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)	DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP and transitioning from NIS+ to LDAP
System Administration Guide: Naming and Directory Services (NIS+)	NIS+ naming and directory services
System Administration Guide: Network Services	Web cache servers, time-related services, network file systems (NFS and Autofs), mail, SLP, and PPP
System Administration Guide: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Solaris cryptographic framework, privileges, RBAC, SASL, and Solaris Secure Shell
System Administration Guide: Solaris Containers-Resource Management and Solaris Zones	Resource management topics projects and tasks, extended accounting, resource controls, fair share scheduler (FSS), physical memory control using the resource capping daemon (rcapd), and dynamic resource pools; virtualization using Solaris Zones software partitioning technology

Related Books

- *DNS and Bind*, by Cricket Liu and Paul Albitz, (O' Reilly, 1992)

Accessing Sun Documentation Online

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

Ordering Sun Documentation

Sun Microsystems offers select product documentation in print. For a list of documents and how to order them, see “Buy printed documentation” at `http://docs.sun.com`.

Typographic Conventions

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

TABLE P–1 Typographic Conventions

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

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

AaBbCc123	Command-line placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. Do *not* save the file. (Emphasis sometimes appears in bold online.)

Shell Prompts in Command Examples

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

TABLE P–2 Shell Prompts

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

PART I About Naming and Directory Services

The following chapter describes the `nsswitch.conf` file, which you use to coordinate the use of the different services.

CHAPTER **1**

The Name Service Switch

This chapter describes the name service switch, what it does, and how clients use it to obtain naming information from one or more sources. You use the name service switch to coordinate usage of different naming services. For an overview of the Solaris naming and directory services DNS, NIS and LDAP, see "Naming and Directory Services (Overview)" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

About the Name Service Switch

The name service switch is a file named `nsswitch.conf`. It controls how a client machine or application obtains network information. It is used by client applications that call any of the `getXbyY()` interfaces such as:

- `gethostbyname()`
- `getpwuid()`
- `getpwnam()`
- `getipnodebyname()`

Each machine has a switch file in its `/etc` directory. Each line of that file identifies a particular type of network information, such as host, password, and group, followed by one or more sources where the client is to look for that information.

A client can obtain naming information from one or more of the switch's sources. For example, an NIS+ client could obtain its hosts information from an NIS+ table and its password information from a local `/etc` file. In addition, it could specify the conditions under which the switch must use each source (see Table 1–1).

The Solaris software automatically loads an `nsswitch.conf` file into every machine's `/etc` directory as part of the installation process. Four alternate (template) versions of the switch file are also loaded into `/etc` for LDAP, NIS, NIS+, or files. See "The `nsswitch.conf` Template Files" on page 34.

These four files are alternate default switch files. Each one is designed for a different primary naming service: `/etc` files, NIS, NIS+, or LDAP. When the Solaris software is first installed on a machine, the installer selects the machine's default naming service: NIS+, NIS, local files, or LDAP. During installation, the corresponding template file is copied to `nsswitch.conf`. For example, for a machine client using LDAP, the installation process copies `nsswitch.ldap` to `nsswitch.conf`. Unless you have an unusual namespace, the default template file as copied to `nsswitch.conf` should be sufficient for normal operation.

If you later change a machine's primary naming service, you copy the appropriate alternate switch file to `nsswitch.conf`. (See "The `nsswitch.conf` Template Files" on page 34.) You can also change the sources of particular types of network information used by the client by editing the appropriate lines of the `/etc/nsswitch.conf` file. The syntax for doing this is described below, and additional instructions are provided in "Modifying the name service switch" on page 38.

Format of the `nsswitch.conf` File

The `nsswitch.conf` file is essentially a list of 16 types of information and the sources that `getXXbyYY()` routines search for that information. The 16 types of information, not necessarily in this order, are:

- `aliases`
- `bootparams`
- `ethers`
- `group`
- `hosts`
- `ipnodes`
- `netgroup`
- `netmasks`
- `networks`
- `passwd` (includes shadow information)
- `protocols`
- `publickey`
- `rpc`
- `services`
- `automount`
- `sendmailvars`

The following table provides a description of the kind of sources that can be listed in the switch file for the information types above.

TABLE 1–1 Switch File Information Sources

Information Sources	Description
`files`	A file stored in the client's `/etc` directory. For example, `/etc/passwd`.
`nisplus`	An NIS+ table. For example, the `hosts` table.
`nis`	An NIS map. For example, the `hosts` map.
`compat`	Compat can be used for password and group information to support old-style + or - syntax in `/etc/passwd`, `/etc/shadow`, and `/etc/group` files.
`dns`	Can be used to specify that host information be obtained from DNS.
`ldap`	Can be used to specify entries be obtained from the LDAP directory.

Search Criteria

Single Source. If an information type has only one source, such as `nisplus` a routine using the switch searches for the information in that source *only*. If it finds the information, it returns a `success` status message. If it does not find the information, it stops searching and returns a different status message. What the routine does with the status message varies from routine to routine.

Multiple Sources. If a table has more than one source for a given information type, the switch directs the routine to start searching for the information in the first source that is listed. If it finds the information, it returns a `success` status message. If it does not find the information in the first source, it tries the next source. The routine will search through all of the sources until it has found the information it needs, or it is halted by encountering a `return` specification. If all of the listed sources are searched without finding the information, the routine stops searching and returns a `non-success` status message.

Switch Status Messages

If a routine finds the information, it returns a `success` status message. If it does not find the information for which it is looking, it returns one of three unsuccessful status messages, depending on the reason for not finding the information. Possible status messages are listed in the following table.

TABLE 1–2 Switch Search Status Messages

Status Message	Meaning of Message
`SUCCESS`	The requested entry was found in the specified source.
`UNAVAIL`	The source is not responding or is unavailable. That is, the NIS+ table, or NIS map, or `/etc` file could not be found or accessed.

TABLE 1–2 Switch Search Status Messages *(Continued)*

Status Message	Meaning of Message
`NOTFOUND`	The source responded with "No such entry." In other words, the table, map, or file was accessed but it did not contain the needed information.
`TRYAGAIN`	The source is busy; it might respond next time. In other words, the table, map, or file was found, but it could not respond to the query.

Switch Action Options

You can instruct the switch to respond to status messages with either of these two *actions* shown in the following table.

TABLE 1–3 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 combination of `nsswitch.conf` file status message and action option determines what the routine does at each step. This combination of status and action is called the 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`. Stop looking for the information and proceed using the information that has been found.
- `UNAVAIL=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status.
- `NOTFOUND=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status.
- `TRYAGAIN=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status.

Because these are the default search criteria, they are assumed. That is, you do not have to explicitly specify them in the switch file. You can change these default search criteria by explicitly specifying some other criteria using the *STATUS=action* syntax show above. For example, the default action for a `NOTFOUND` condition is to continue the search to the next source. To specify that for a particular type of information, such as `networks`, the search is to halt on a `NOTFOUND` condition, you would edit the `networks` line of the switch file to read:

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

The `networks: nis [NOTFOUND=return] files` line specifies a non-default criterion for the `NOTFOUND` status. Non-default criteria are delimited by square brackets.

In this example, the search routine behaves as follows:

- If the `networks` map is available and contains the needed information, the routine returns with a `SUCCESS` status message.
- If the `networks` map is not available, the routine returns with an `UNAVAIL` status message and by default continues on to search the appropriate `/etc` file.
- If the `networks` map is available and found, but it does not contain the needed information, the routine returns with a `NOTFOUND` message. But, instead of continuing on to search the appropriate `/etc` file, which would be the default behavior, the routine stops searching.
- If the `networks` map is busy, the routine returns with an `TRYAGAIN` status message and by default continues on to search the appropriate `/etc` file.

Note – Lookups in the `nsswitch.conf` file are done in the order in which items are listed. However, password updates are done in reverse order, unless otherwise specified by using the `passwd -r` *repository* command. See "The Switch File and Password Information" on page 40 for more information.

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 switch file's defaults.

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 default values.

Auto_home and Auto_master

The switch search criteria for the `auto_home` and `auto_master` tables and maps is combined into one category called `automount`.

Timezone and the Switch File

The `timezone` table does not use the switch, so it is not included in the switch file's list.

Comments in `nsswitch.conf` Files

Any `nsswitch.conf` file line beginning with a comment character (`#`) is interpreted as a comment line and is ignored by routines that search the file.

When a comment character (`#`) is included in the middle of the line, characters preceding the comment mark *are* interpreted by routines that search the `nsswitch.conf` file. Characters to the right of the comment mark are interpreted as comments and ignored.

TABLE 1–4 Switch File Comment Examples

Type of Line	Example
Comment line (not interpreted).	# hosts: nisplus [NOTFOUND=return] files
Fully interpreted line.	hosts: nisplus [NOTFOUND=return] file
Partially interpreted line (the `files` element not interpreted)	hosts: nisplus [NOTFOUND=return] # files

Keyserver and `publickey` Entry in the Switch File

Caution – You must restart the keyserver after you make a change to `nsswitch.conf`

The keyserver reads the `publickey` entry in the name service switch configuration file only when the keyserver is started. As a result, if you change the switch configuration file, the keyserver does not become aware of changes to the `publickey` entry until it is restarted.

The `nsswitch.conf` Template Files

Four `nsswitch.conf` template files are provided with the Solaris software to accommodate different naming services. Each of them provides a different default set of primary and subsequent information sources.

The four template files are:

- *NIS+ template file*. The `nsswitch.nisplus` configuration file specifies NIS+ as the primary source for all information except passwd, group, automount, and aliases. For those four 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.

- *NIS template 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. 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.
- *Files template file*. The `nsswitch.files` configuration file specifies local `/etc` files as the only source of information for the machine. There is no "files" source for `netgroup`, so the client will not use that entry in the switch file.

Copy the template file that most closely meets your requirements to the`nsswitch.conf` configuration file and then modify the file as needed.

For example, to use the LDAP template file, you would type the following command:

```
mymachine# cp nsswitch.ldap nsswitch.conf
```

The Default Switch Template Files

Here are the four switch files supplied with Solaris software.

EXAMPLE 1–1 NIS+ Switch File Template (`nsswitch.nisplus`)

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

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

EXAMPLE 1–1 NIS+ Switch File Template (`nsswitch.nisplus`) *(Continued)*

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

EXAMPLE 1–2 NIS Switch File Template

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

EXAMPLE 1–3 Files Switch File Template

```
#
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
```

EXAMPLE 1–3 Files Switch File Template *(Continued)*

```
passwd: files
group: files
hosts: files
networks: files
protocols: files
rpc: files
ethers: files
netmasks: files
bootparams: files
publickey: files
# At present there isn't a 'files' backend for netgroup;
# the system will figure it out pretty quickly, and will notuse
# netgroups at all.
netgroup: files
automount: files
aliases: files
services: files
sendmailvars: files
```

The `nsswitch.conf` File

The default `nsswitch.conf` file that is installed when you install the Solaris software for the first time is determined by which naming service you select during the Solaris software installation process. Each line of that file identifies a particular type of network information, such as host, password, and group, followed by one or more sources, such as NIS+ tables, NIS maps, the DNS hosts table, or local `/etc`, where the client is to look for that information. When you chose a naming service, the switch template file for that service is copied to create the new `nsswitch.conf` file. For example, if you choose NIS+, the `nsswitch.nisplus` file is copied to create a new `nsswitch.conf` file.

An `/etc/nsswitch.conf` file is automatically loaded into every machine's `/etc` directory by the Solaris 9release software, along with the following alternate (template) versions:

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

These alternate template files contain the default switch configurations used by the NIS+ and NIS services, local files, and LDAP. When the Solaris software is first installed on a machine, the installer selects the machine's default naming service: NIS+, NIS, local files, or LDAP. During installation, the corresponding template file is copied to `/etc/nsswitch.conf`. For example, for a machine client using NIS+, the installation process copies `nsswitch.nisplus` to `nsswitch.conf`.

Unless you have an unusual namespace, the default template file as copied to `nsswitch.conf`should be sufficient for normal operation.

Selecting a Different Configuration File

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

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

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

▼ Modifying the name service switch

To change to a switch file, follow these steps:

1. **Log in to the client as superuser.**
2. **Copy the alternate file appropriate for the machine's naming service over the `nsswitch.conf` file.**

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

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

   NIS Version

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

   Local `/etc` *Files Version*

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

3. **Reboot the machine.**

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

How to Enable an NIS+ Client to Use IPv6

1. **Log in as superuser.**
2. **Edit the `/etc/nsswitch.conf` file.**
3. **Add the new `ipnodes` source and specify the naming service (such as ldap).**

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

   `ipnodes` defaults to `files`. During the transition from IPv4 to IPv6, where all naming services are not aware of IPv6 addresses, you should accept the `files` default. Otherwise, unnecessary delays might result during the resolution of addresses.
4. **Save the file and reboot the machine.**

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

Ensuring Compatibility With +/- Syntax

If +/- is used in `/etc/passwd`, `/etc/shadow`, and `/etc/group` files, you will need to modify the `nsswitch.conf`file to insure compatibility.

- *NIS+*. To provide +/- semantics with NIS+, change the `passwd` and `groups` sources to `compat` and add a `passwd_compat: nisplus` entry to the `nsswitch.conf` file after the `passwd` or `group` entry as shown below:

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

  The above specifies that client routines obtain their network information from `/etc` files and NIS+ tables as indicated by the `+/-` entries in the files.
- *NIS*. To provide the same syntax as in the SunOS™ 4 release, change the `passwd` and `groups` sources to `compat`.

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

  This specifies that /etc files and NIS maps as indicated by the +/- entries in the files.

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

The Switch File and Password Information

It is possible to include and access password information in multiple repositories, such as `files` and `nisplus`. You can use the `nsswitch.conf` file to establish the lookup order for that information.

Caution – `files` must be the first source in the `nsswitch.conf` file for `passwd` information.

In an NIS+ environment, the `passwd` line of the `nsswitch.conf` file should list the repositories in the following order.

```
passwd: files nisplus
```

In an NIS environment, the `passwd` line of the `nsswitch.conf` file should list the repositories in the following order.

```
passwd: files nis
```

Tip – Listing `files` first allows `root` to log in, under most circumstances, even when the system encounters some network or naming services issues.

Maintaining multiple repositories *for the same user* is not recommended. By maintaining centralized password management in a single repository for each user, you reduce the possibilities of confusion and error. If you choose to maintain multiple repositories per user, update password information by using the `passwd -r` command.

```
passwd -r repository
```

If no repository is specified with the `-r` option, `passwd` updates the repositories listed in `nsswitch.conf` in reverse order.

PART II NIS+ Setup and Configuration

This part describes the setup and configuration of the NIS+ naming service in the Solaris OS.

CHAPTER 2

NIS+: An Introduction

This chapter provides an overview of the *Network Information Service Plus* (NIS+).

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

About NIS+

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

The NIS+ name 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.

NIS+ enables you to store information about machine addresses, security information, mail information, Ethernet interfaces, and network services in central locations where all machines 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. 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 requester is a valid user on the network. *Authorization* determines whether a particular user is allowed to have or modify the information requested.

Solaris clients use the name service switch (`/etc/nsswitch.conf` file) to determine from where a machine 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. A complete description of the switch software and its associated files is provided in Chapter 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

Within the security system described in "NIS+ Security" on page 48, 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 your 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

The *Network Information Service Plus* (NIS+) differs from the *Network Information Service* (NIS) in several ways. NIS+ has many new features, and the terminology it uses for concepts similar to NIS is different. Look in the Glossary if you see a term you don't recognize. The following table gives an overview of the major differences between NIS and NIS+.

TABLE 2–1 Differences Between NIS and NIS+

NIS	NIS+
Flat domains—no hierarchy	Hierarchical —data stored in different levels in the namespace
Data stored in two 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

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-1980s. 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 client-server networks 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 1990s require scalability and decentralized administration, the NIS+ namespace was designed with hierarchical domains, like those of DNS.

For example, Figure 2–1 shows a sample company with a parent domain named `doc`, and two subdomains named `sales` and `manf`.

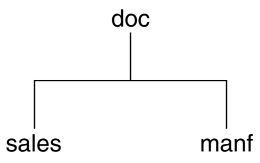


FIGURE 2–1 Example of Hierarchical Domains

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 splits 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 machines, 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 your umbrella domain. A domain can even be administered from within another domain.

The principal NIS+ 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.

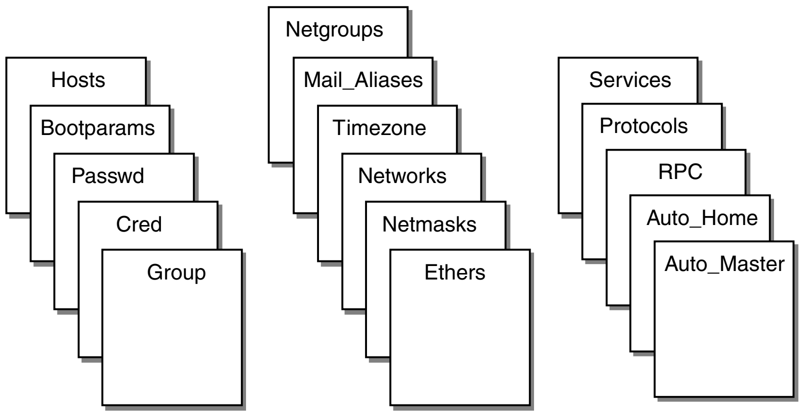
However, NIS+ uses an updating model that is completely different from the you used by NIS. Since at the time NIS was developed, the type of information it would store changed infrequently, NIS was developed with an update model that focused on stability. Its updates are handled manually and, in large 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. 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 "NIS+ Domains" on page 57, "NIS+ Servers" on page 59, and "NIS+ Clients and Principals" on page 62, respectively.

An NIS+ domain can be connected to the Internet through its NIS+ clients, using the name service switch (see Example 1–1). 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 machine addresses, while the `passwd` table stores information about users of the network.

NIS+ tables provide two major improvements over the maps used by NIS. First, you can search an NIS+ table 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, you can access and manipulate the information in NIS+ tables 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 10.

Yon can use NIS in conjunction with NIS+ under the following principles and conditions:

- Servers within a domain. While you can have both NIS and NIS+ servers operating in the same domain, doing so is not recommended for long periods. As a general rule, using both services in the same domain should be limited to a relatively short transition period from NIS to NIS+.
- Subdomains. If the master server of your root domain is running NIS+, you can set up subdomains whose servers are all running NIS. (If your root domain master server is running NIS, you cannot have subdomains.)
- Machines within a domain. If a domain's servers are running NIS+, individual machines within that domain can be set up to use either NIS+, NIS, or `/etc` files for their name service information. In order for an NIS+ server to supply the needs of an NIS client, the NIS+ server must be running in NIS-compatibility mode.

  If a domain's servers are running NIS, individual machines within that domain can be set up to use either NIS or `/etc` files for name services (they cannot use NIS+).

  The service a machine uses for various name services is controlled by the machine's `nsswitch.conf` file. This file is called the *switch* file. See Chapter 1 for further information.

NIS+ Security

NIS+ protects the structure of the namespace, and the information it stores, by the complementary processes of *authorization* and *authentication*.

- *Authorization*. Every component in the namespace specifies the type of operation it will accept and from whom. This is authorization.
- *Authentication*. NIS+ attempts to *authenticate* every request for access to the namespace. Requests come from NIS+ *principals*. An NIS+ principal can be a process, machine, root, or a user. Valid NIS+ principals possess an NIS+ *credential*. NIS+ authenticates the originator of the request (principal) by checking the principal's credential.

If the principal possesses an authentic (valid) credential, and if the principal's request is one that the principal is authorized to perform, NIS+ carries out the request. If either the credential is missing or invalid, or the request is not one the principal is authorized to perform, NIS+ denies the request for access. An introductory description of the entire NIS+ security system is provided in Chapter 11.

Solaris 1.x Releases and NIS-Compatibility Mode

NIS+ can be used by machines running NIS with Solaris 1x or 2x Release software. In other words, machines within an NIS+ domain can have their `nsswitch.conf` files set to `nis` rather than `nisplus`. To access NIS+ service on machines running NIS, you must run the NIS+ servers in *NIS-compatibility mode*.

NIS-compatibility mode enables an NIS+ server running Solaris software 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 that an 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 "Setting Up NIS+ Servers" in *System Administration Guide: Naming and Directory Services (NIS+)*.

Two more differences need to be pointed out. First, instructions for setting up a server in NIS-compatibility mode are slightly different than those used to set up a standard NIS+ server. Second, 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 2. However, to understand more about the authentication process and NIS-compatibility mode, see "NIS+ Security Overview" in *System Administration Guide: Naming and Directory Services (NIS+)*.

If you wish to set the NIS-compatibility mode option to persist across reboots, you must modify the `/lib/svc/method/nisplus` file, as shown in "NIS+ and the Service Management Facility" on page 85.

NIS+ Administration Commands

NIS+ provides a full set of commands for administering a namespace. The table below, summarizes them.

Note – Most of the command line administrative tasks associated with the NIS+ service are managed by the Service Management Facility (SMF). For details about using SMF with NIS+, see "NIS+ and the Service Management Facility" on page 85. For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

TABLE 2–2 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.
`nisauthconf`	Optionally configure Diffie-Hellman key length.
`nisbackup`	Backs up NIS directories.
`nis_cachemgr`	Starts the NIS+ cache manager on an NIS+ client.
`niscat`	Displays the contents of NIS+ tables.
`nis_checkpoint`	Forces service to checkpoint data that has been entered in the log but not checkpointed to disk.
`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.
`nisclient`	Initializes NIS+ principals.
`nisdefaults`	Lists an NIS+ object's default values: domain name, group name, machine 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+ tables.
`nislog`	Displays the contents of NIS+ transaction log.

TABLE 2–2 NIS+ Namespace Administration Commands *(Continued)*

Command	Description
`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. (Rather than using `nispasswd`, you should use `passwd` or `passwd -r nisplus`.)
`nis_ping`	Forces a replica to update its data from the master server.
`nispopulate`	Populates the NIS+ tables in a new NIS+ domain.
`nisprefadm`	Specifies the order in which clients are to seek NIS+ information from NIS+ servers.
`nisrestore`	Restores previously backed up NIS+ directories and can also be used to quickly bring online new NIS+ replica servers.
`nisrm`	Removes NIS+ objects (except directories) from the namespace.
`nisrmdir`	Removes NIS+ directories and replicas from the namespace.
`nisserver`	Shell script used to set up a new NIS+ server.
`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.
`nisstat`	Reports statistics and other information about an NIS+ server.
`nistbladm`	Creates or deletes NIS+ tables, and adds, modifies or deletes entries in an NIS+ table.
`nistest`	Reports the current state of the NIS+ namespace.
`nisupdkeys`	Updates the public keys stored in an NIS+ object.
`passwd`	Changes password information stored in the NIS+ Passwd table. Also administers password aging and other password-related parameters.

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()`)

Setup and Configuration Preparation

Before configuring your NIS+ namespace, you must:

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

    See "NIS+: An Introduction" in *System Administration Guide: Naming and Directory Services (NIS+)* for a detailed description of these and other planning issues, and recommended guidelines.
- Prepare your existing namespace (if any). See "Preparing the Existing Namespace" on page 75.
- Choose a root server machine.
- Make sure that you have at least one system already running at your site that can be used as your root master server. This machine must contain at least one user (root) in the system information files, such as `/etc/passwd`. Machines usually come with root in the system files, so this should not be a problem.

NIS and NIS+

Both NIS and NIS+ perform some of the same tasks. NIS+, however, allows for hierarchical domains, namespace security, and other features that NIS does not provide. For a more detailed comparison between NIS and NIS+, see "How NIS+ Differs From NIS" in *System Administration Guide: Naming and Directory Services (NIS+)*.

You can use NIS in conjunction with NIS+ under the following principles and conditions:

- Servers within a domain. While you can have both NIS and NIS+ servers operating in the same domain, doing so is not recommended for long periods. As a general rule, using both services in the same domain should be limited to a relatively short transition period from NIS to NIS+. If some clients need NIS service, you can run NIS+ in NIS-compatibility mode as explained in "Solaris 1.x Releases and NIS-Compatibility Mode" on page 49.
- Subdomains. If the master server of your root domain is running NIS+, you can set up subdomains whose servers are all running NIS. (If your root domain master server is running NIS, you cannot have subdomains.) This might be useful in situations where you are moving from NIS to NIS+. For example, suppose your enterprise had separate, multiple NIS domains, possibly at different sites. Now you need to link them all together into a single, hierarchical multi-domain namespace under NIS+. By first setting up the root domain under NIS+, you can then designate the legacy NIS domains as sub-domains that continue to run NIS until it is convenient to switch them over to NIS+.
- Machines within a domain.
  - If a domain's servers are running NIS+, individual machines within that domain can be set up to use either NIS+, NIS, or `/etc` files for their name service information. In order for an NIS+ server to supply the needs of an NIS client, the NIS+ server must be running in NIS-compatibility mode as described in "Solaris 1.x Releases and NIS-Compatibility Mode" on page 49.
  - If a domain's servers are running NIS, individual machines within that domain can be set up to use either NIS or `/etc` files for name services (they cannot use NIS+).

NIS+ Files and Directories

Table 2–3 lists the UNIX directories used to store NIS+ files.

TABLE 2–3 Where NIS+ Files are Stored

Directory	Where	Contains
`/usr/bin`	All machines	NIS+ user commands
`/usr/lib/nis`	All machines	NIS+ administrator commands
`/usr/sbin`	All machines	NIS+ daemons
`/usr/lib/`	All machines	NIS+ shared libraries
`/var/nis/data`	NIS+ server	Data files used by NIS+ server
`/var/nis`	NIS+ server	NIS+ working files
`/var/nis`	NIS+ client machines	Machine-specific data files used by NIS+

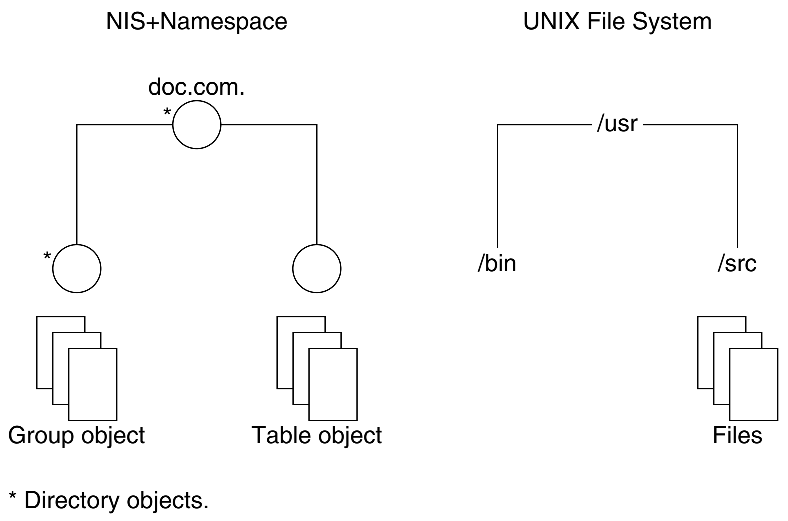
Caution – Do not rename the `/var/nis` or `/var/nis/data` directories or any of the files in these directories that were created by `nisinit` or any of the other NIS+ setup procedures. In Solaris Release 2.4 and earlier versions, the `/var/nis` directory contained two files named *hostname*`.dict` and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

Note – With the Solaris platform, the NIS+ data dictionary (`/var/nis/data.dict`) is now machine independent. This allows you to easily change the name of an NIS+ server. You can also now use the NIS+ backup and restore capabilities to transfer NIS+ data from one server to another. See Chapter 21.

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, machines, 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 five important differences:

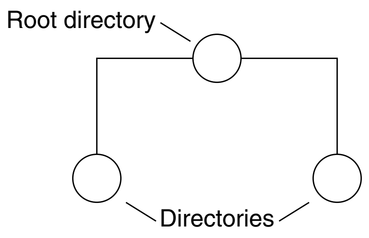
- Although both use directories, the other objects in an NIS+ namespace are tables and groups, not files.
- The NIS+ namespace is administered only through NIS+ administration commands or graphical user interfaces designed for that purpose, such as the Solaris Management Console™ tools; it cannot be administered with standard

UNIX file system commands or GUIs.

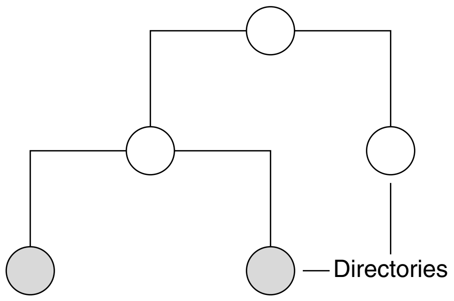
- The names of UNIX file system components are separated by slashes (`/usr/bin`), but the names of NIS+ namespace objects are separated by dots (`doc.com.`).
- The "root" of a UNIX file system is reached by stepping through directories from right to *left* (`/usr/src/file1`), while the root of the NIS+ namespace is reached by stepping from left to *right* (`sales.doc.com.`).
- Because NIS+ object names are structured from left to right, a fully qualified name always ends in a dot. Any NIS+ object ending in a dot is assumed to be a fully qualified name. NIS+ object names that do not end in a dot are assumed to be relative names.

NIS+ Namespace 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.

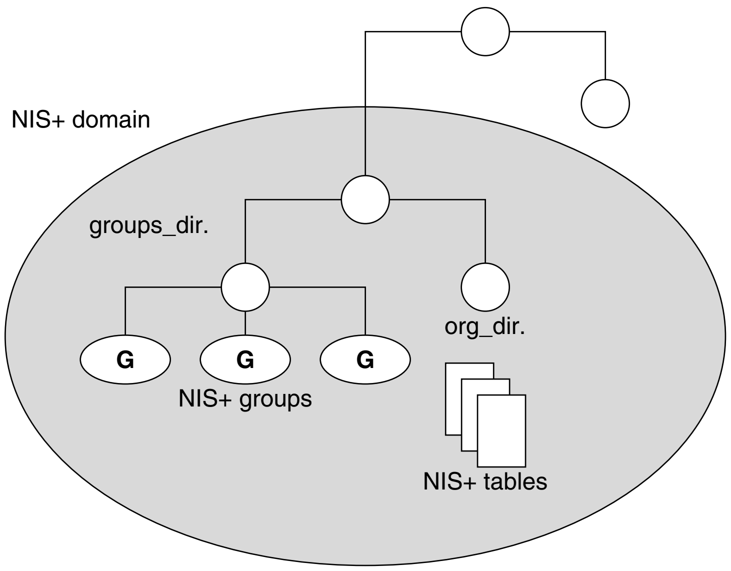
Whereas UNIX directories are designed to hold UNIX files, NIS+ directories are designed to hold NIS+ objects: other directories, tables and groups. Each NIS+ domain-level directory contains the following sub-directories:

- `groups_dir`. Stores NIS+ group information.
- `org_dir`. Stores NIS+ system tables.

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 Manufacturing Division.

NIS+ 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.

For example, suppose the DOC company has Sales and Manufacturing divisions. To support those divisions, its NIS+ namespace would most likely be arranged into three major directory groups, with a structure that looked like the following diagram.

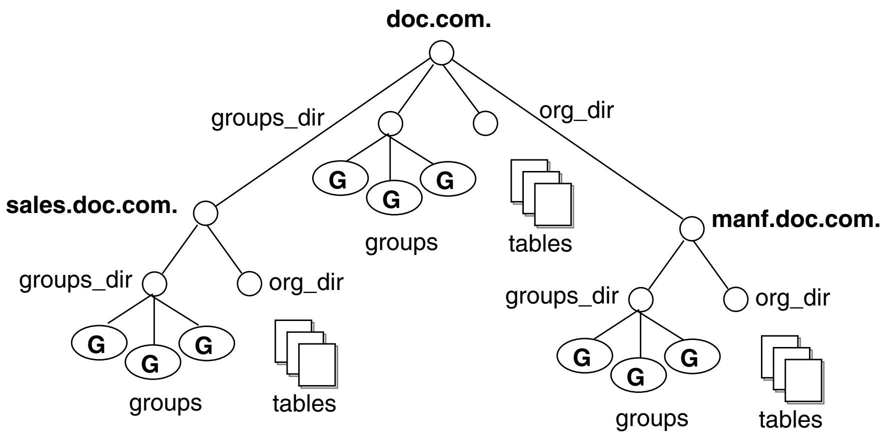


FIGURE 2–2 Example NIS+ Directory Structure

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

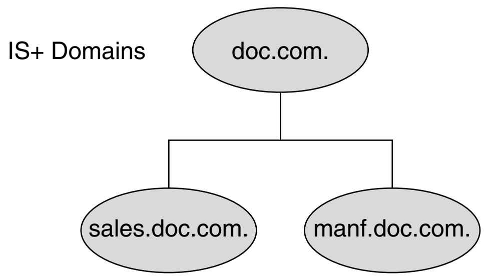
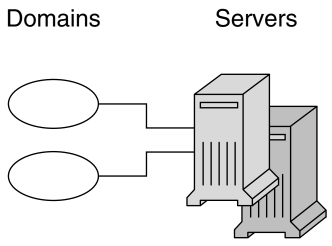


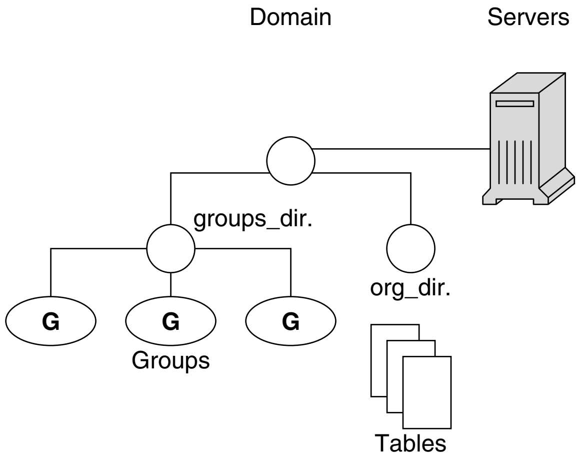
FIGURE 2–3 Example NIS+ Domains

NIS+ 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*:

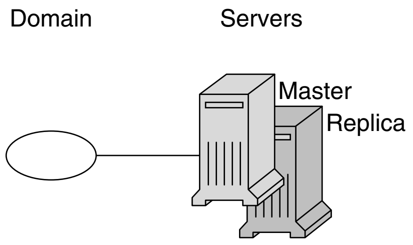


This connection between the server and the directory object is established during the process of setting up a domain. One thing is important to mention now: when that connection is established, the directory object stores the name and IP address of its server. This information is used by clients to send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

Any Solaris platform based machine can be an NIS+ server. The software for both NIS+ servers and clients is bundled together into the release. Therefore, any machine that has the Solaris Release 2 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 machine 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 machine 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 you root master server. The master servers of other domains are simply called master servers. Likewise, there are root replica servers and regular replica servers.

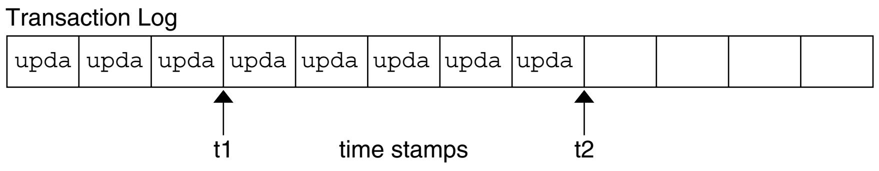
Both master and replica servers store NIS+ tables and answer client requests. The master, however, stores the master copy of a domain's tables. The replicas store only duplicates. The administrator loads information into the tables in the master server, and the master server propagates it to the replica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

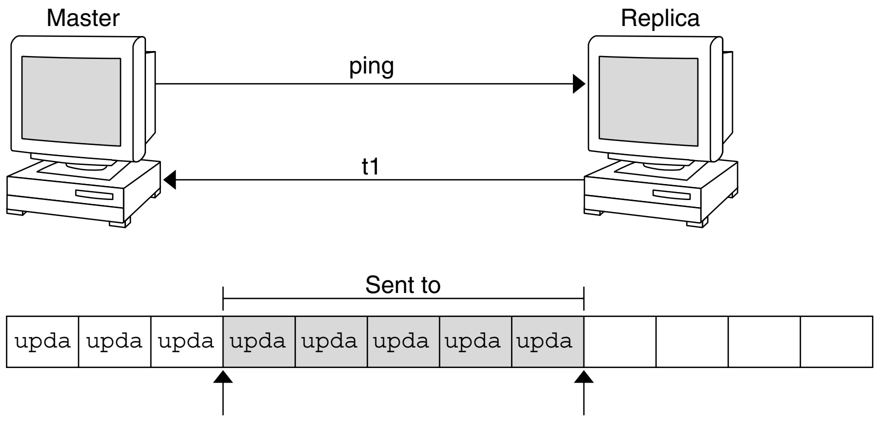
An NIS+ master server implements updates to its objects immediately; however, it tries to "batch" several updates together before it propagates them to its replicas. When a master server receives an update to an object, whether a directory, group, link, or table, it waits about two minutes for any other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



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 time stamp 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 time stamp 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 time stamp:



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 time stamp from a replica that is before its earliest time stamp 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 to requests with a `Server Busy - Try Again` or similar message.

NIS+ Clients and Principals

NIS+ principals are the entities (clients) that submit requests for NIS+ services.

NIS+ Principal

An NIS+ principal may be someone who is logged in to a client machine as a regular user or someone who is logged in as superuser (root). In the first instance, the request actually comes from the client user; in the second instance, the request comes from the client machine. Therefore, an NIS+ principal can be a client user or a client machine.

(An NIS+ principal can also be the entity that supplies an NIS+ service from an NIS+ server. Since all NIS+ servers are also NIS+ clients, much of this discussion also applies to servers.)

NIS+ Client

An NIS+ client is a machine 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.)

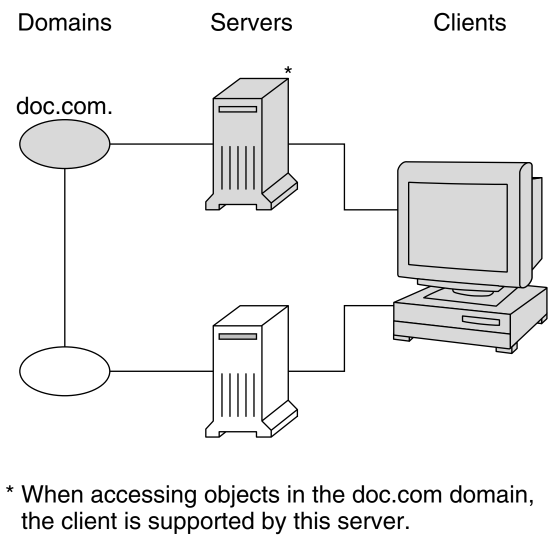
An NIS+ client can access any part of the namespace, subject to security constraints. In other words, if it has been authenticated and has been granted the proper permissions, it can access information or objects in any domain in the namespace.

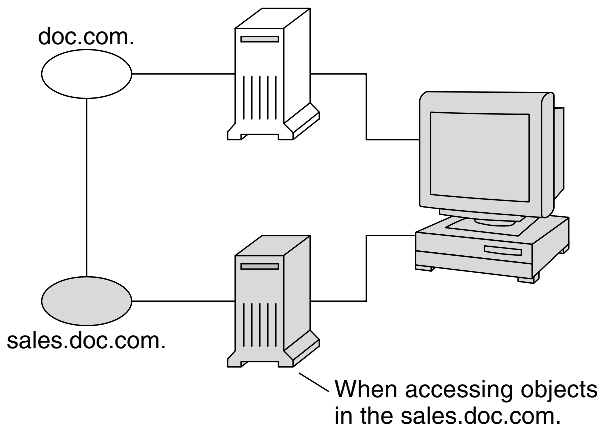
Although a client can access the entire namespace, a client *belongs* to only one domain, which is referred to as its *home* domain. A client's home domain is usually specified during installation, but it can be changed or specified later. All the information about a client, such as its IP address and 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 machine into an NIS+ table does not automatically make that machine an NIS+ client. It simply makes information about that machine available to all NIS+ clients. That machine cannot request NIS+ service unless it is actually set up as an NIS+ client.

Conversely, making a machine an NIS+ client does not enter information about that machine into an NIS+ table. It simply allows that machine to receive NIS+ service. If information about that machine 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:


Domains
Servers
Clients
*
doc.com.
* When accessing objects in the doc.com domain,
the client is supported by this server.

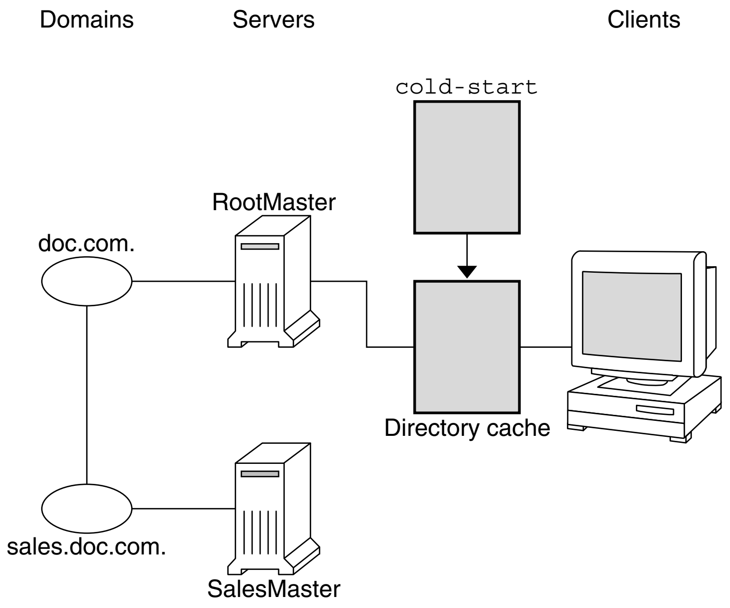


How does the client know which server that is? By trial and error. Beginning with its home server, the client tries first one server, then another, until it finds the right one. When a server cannot answer the client's request, it sends the client information to help locate the right server. Over time, 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 Cold-Start File and Directory Cache

When a client is initialized, it is given a *cold-start file*. The cold-start 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 cold-start file contains the directory object of the client's home domain.

A cold-start file is used only to initialize a client's local *directory cache*. The directory cache is managed by an NIS+ facility called the *cache manager*. The cache manager stores the directory objects that enable a client to send its requests to the proper servers. The information obtained from the client's cold-start file is downloaded into a file named `NIS_SHARED_DIRCACHE` in `/var/nis`.



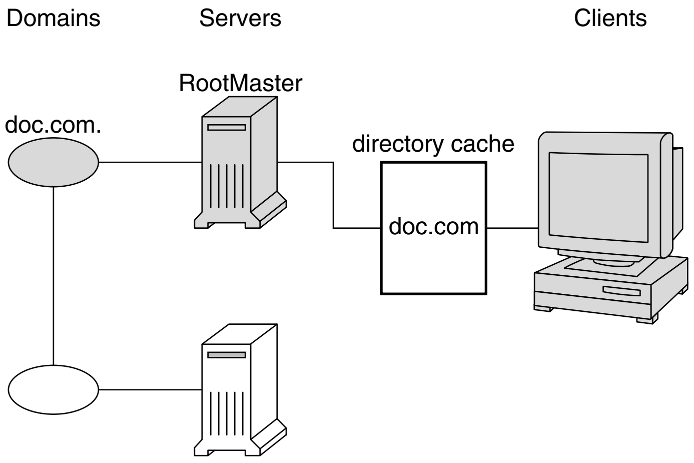
By storing a copy of the namespace's directory objects in its directory cache, a client can know which servers support which domains. (To view the contents of a client's cache, use the `nisshowcache` command, described in "The `nisshowcache` Command" on page 336.) Here is a simplified example:

Domain Name and Directory Name are the same	Supporting Server	IP Address
`doc.com.`	rootmaster	172.29.6.77
`sales.doc.com.`	salesmaster	172.29.6.66

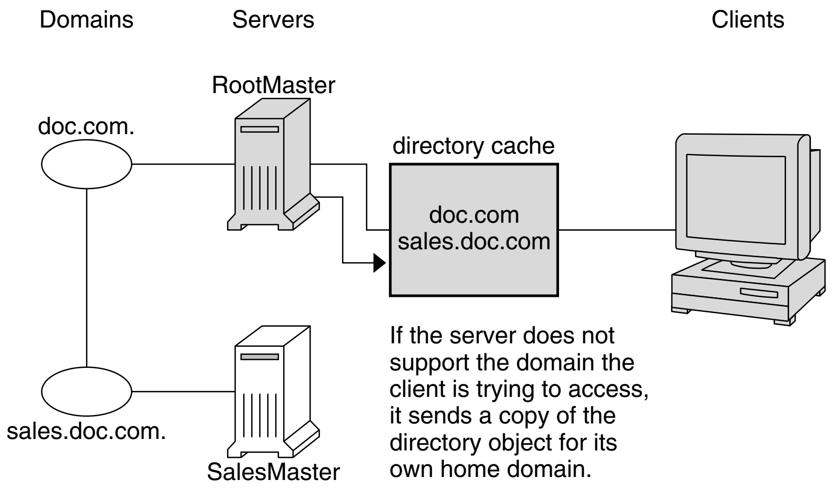
Domain Name and Directory Name are the same	Supporting Server	IP Address
`manf.doc.com.`	manfmaster	172.29.6.37
`int.sales.doc.com.`	Intlsalesmaster	10.22.3.7

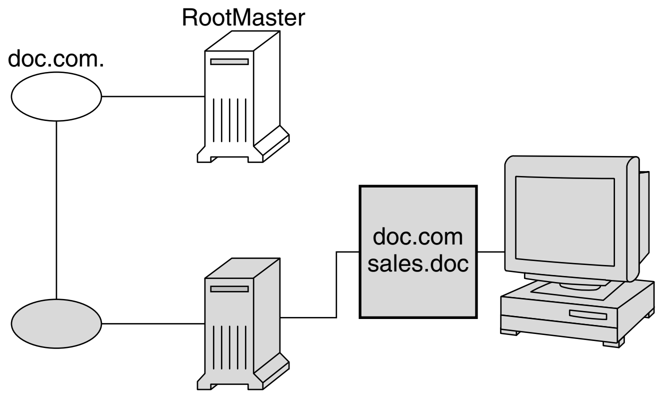
To keep these copies up-to-date, each directory object has a *time-to-live* (TTL) field. Its default value is 12 hours. If a client looks in its directory cache for a directory object and finds that it has not been updated in the last 12 hours, the cache manager obtains a new copy of the object. You can change a directory object's time-to-live value with the `nischttl` command, as described in "The `nischttl` Command" on page 341. 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 cold-start file provides the first entry in the cache. Therefore, when the client sends its first request, the request goes to the server specified by the cold-start 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.doc.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.


Domains
Servers
Clients
RootMaster
doc.com.
directory cache
doc.com
sales.doc.com
If the server does not support the domain the client is trying to access, it sends a copy of the directory object for its own home domain.
sales.doc.com.
SalesMaster



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.

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

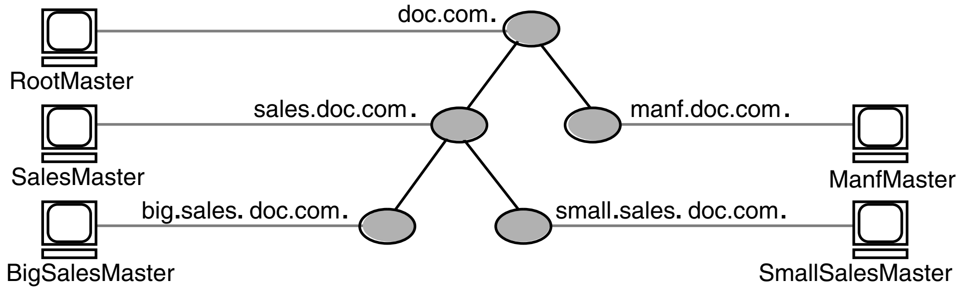
An NIS+ server is also an NIS+ client. In fact, before you can set up a machine as a server, you must initialize it as a client. The only exception is the root master server, which has its own unique setup process.

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:

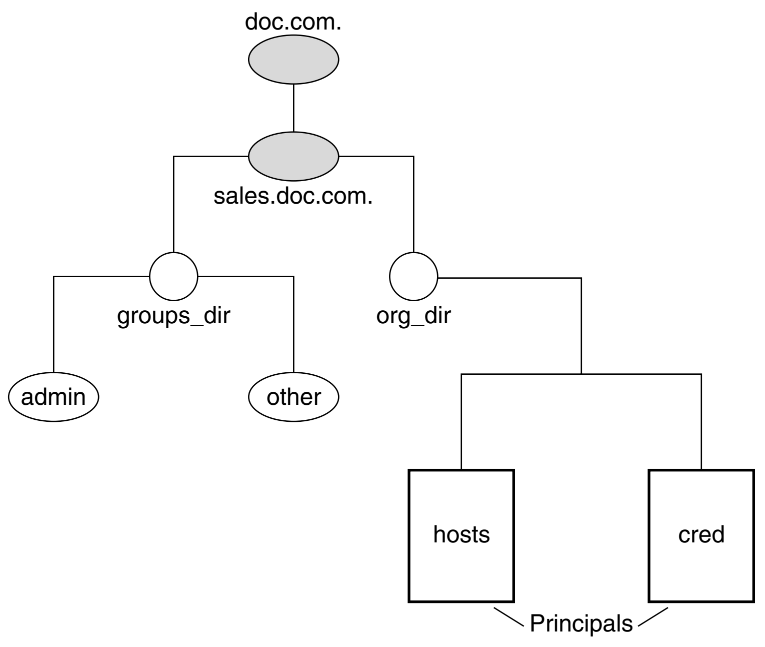
Server	Supports	Belongs to
RootMaster	doc.com.	doc.com.
SalesMaster	sales.doc.com.	doc.com.
IntlSalesMaster	intl.sales.doc.com.	sales.doc.com.
ManfMaster	manf.doc.com.	doc.com.

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 "Naming Conventions" on page 69.

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 below.

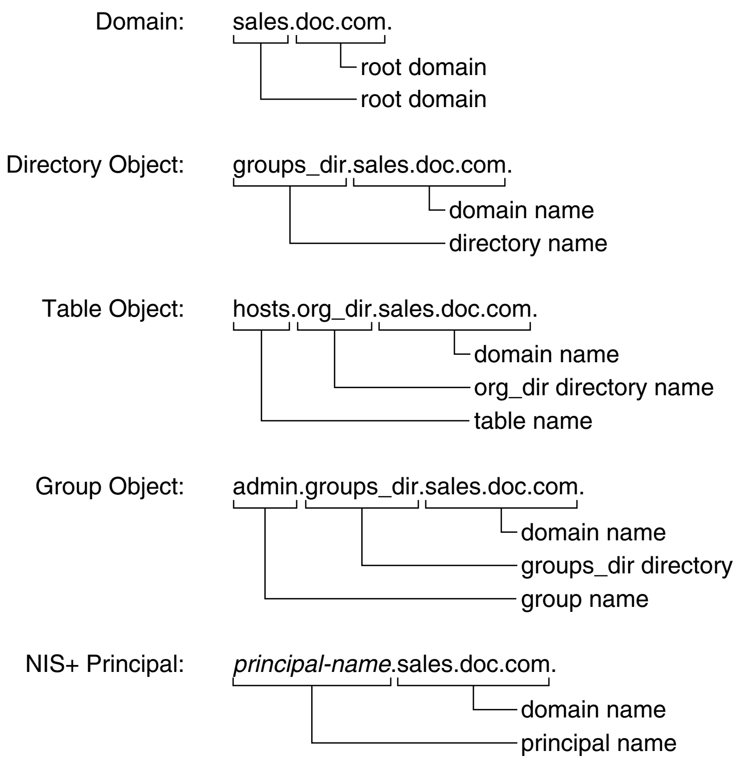


FIGURE 2–4 Fully qualified Names of Namespace Components

NIS+ Domain Names

A fully qualified NIS+ domain name is formed from left to right, starting with the local domain and ending with the root domain:

`doc.com.` (root domain)

`sales.doc.com.` (subdomain)

`intl.sales.doc.com.` (a third level subdomain)

The first line above shows the name of the root domain. The root domain must always have at least two elements (labels) and must end in a dot. The last (right most) label may be anything you want, but in order to maintain Internet compatibility, the last element must be either an Internet organizational name (as shown below), or a two or three character geographic identifier such as `.jp.` for Japan.

TABLE 2–4 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 second and third lines above show the names of lower-level domains.

Directory Object Names

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.manf.doc.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.mydomain.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_PATH` Environment Variable" on page 74).

Tables and Group Names

Fully qualified table and group names are formed by starting with the object name and appending the directory name, followed by the fully qualified 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.doc.com.
admin.groups_dir.doc.com.
admin.groups_dir.sales.doc.com.
admin.groups_dir.sales.doc.com.
hosts.org_dir.doc.com.
hosts.org_dir.doc.com.
hosts.org_dir.sales.doc.com.
hosts.org_dir.sales.doc.com.
```

Table Entry Names

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,...],tablename
```

Column is the name of the table column. *Value* is the actual value of that column. *Tablename* 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.doc.com.
[addr=129.44.2.2,name=elm],hosts.org_dir.sales.doc.com.
[addr=129.44.2.3,name=oak],hosts.org_dir.sales.doc.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 Part 2.

Host Names

Host names may contain up to 24 characters. Letters, numbers, the dash (-) and underscore (_) characters are allowed in host names. Host names are not case sensitive (that is, upper and lower case letters are treated as the same). The first character of a host name must be a letter of the alphabet. Blank spaces are not permitted in host names.

Note – Dots (`.`) are not permitted in host names. For example, a host name such as `doc.2` is not permitted. Dots are not allowed in host names even if they are enclosed in quotes. For example, `'doc.2'` is not permitted. Dots are only used as part of a fully qualified host name to identify the domain components. For example, `doc-2.sales.doc.com.` is a correct fully qualified host name.

Domains and hosts should not have the same name. For example, if you have a sales domain you should not have a machine named `sales`. Similarly, if you have a machine named `home`, you do not want to create a domain named `home`. This caution also applies to subdomains. For example, if you have a machine named `west` you don't want to create a `sales.west.doc.com` subdomain.

NIS+ Principal Names

NIS+ principal names are sometimes confused with Secure RPC netnames. 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. The following list provides examples.

NIS+ principal name	`olivia.sales.doc.com.`
Secure RPC netname	`unix.olivia@sales.doc.com`

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 Name Symbols

You can form namespace names from any printable character in the ISO Latin 1 set. However, the names cannot start with these characters: `@ < > + [ ] - / = . , : ;`

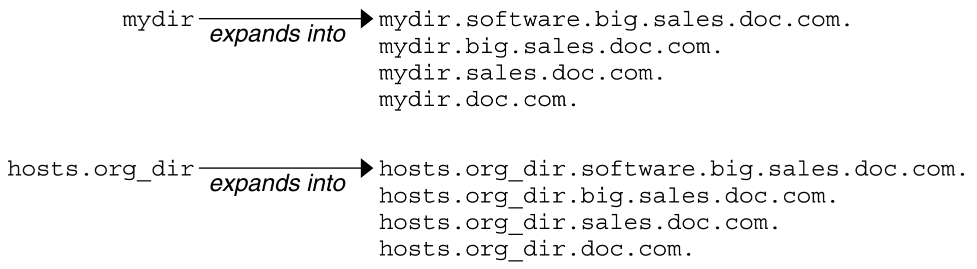
To use a string, enclose it in double quotes. To use a quote sign in the name, quote the sign too (for example, to use `o'henry`, type `o"'"henry`). To include white space (as in John Smith), use double quotes within single quotes, like this:

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

See "Host Names" on page 72 for restrictions that apply to host names.

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.doc.com.` domain).


mydir
expands into
mydir.software.big.sales.doc.com.
mydir.big.sales.doc.com.
mydir.sales.doc.com.
mydir.doc.com.
hosts.org_dir
expands into
hosts.org_dir.software.big.sales.doc.com.
hosts.org_dir.big.sales.doc.com.
hosts.org_dir.sales.doc.com.
hosts.org_dir.doc.com.

NIS_PATH Environment Variable

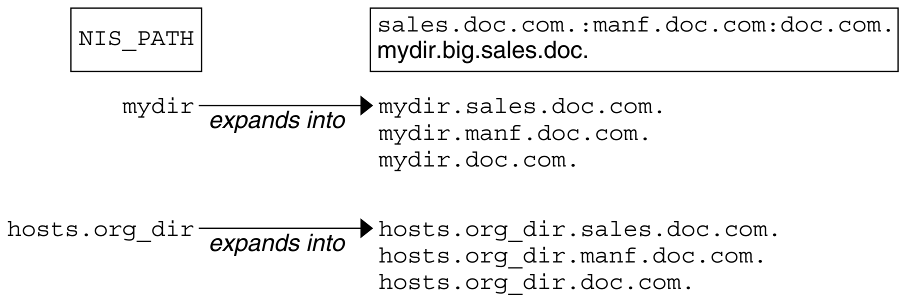
You can change or augment the list of directories NIS+ searches through by changing the value of the environment variable `NIS_PATH`. `NIS_PATH` accepts a list of directory names separated by colons:

```
setenv NIS_PATH directory1: directory2: directory3 ...
```

or

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

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



Like `$PATH` and `$MANPATH`, the `NIS_PATH` variable accepts the 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:

NIS_PATH	$:org_dir.$:groups_dir.$ mydir.big.sales.doc.
If default directory is:	NIS_PATH is effectively:
sales.doc.com.	sales.doc.com.:org_dir.sales.doc.com.:groups_dir.sales.doc.com.
manf.doc.com.	manf.doc.com.:org_dir.manf.doc.com.:groups_dir.manf.doc.com.
doc.com.	doc.com.:org_dir.doc.com.:groups_dir.doc.com.

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

Note – Keep in mind that additions and changes to your `NIS_PATH` may increase the number of lookups that NIS+ has to perform and thus slow down performance.

Preparing the Existing Namespace

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

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

For your reference, key preparations are summarized below:

- *Domain and host names*. Domains and hosts must not have the same name. For example, if you have a `sales` domain you cannot have a machine named `sales`. Similarly, if you have a machine named `home`, do not create a domain named `home`. This caution also applies to subdomains; for example, if you have a machine named `west,` you do not want to create a `sales.west.doc.com` subdirectory.

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

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

Two Configuration Methods

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

- *With the setup (configuration) scripts*. This chapter and Chapter 4 describe how to configure NIS+ using the three NIS+ scripts: `nisserver`, `nispopulate`, and `nisclient`. This is the easiest, as well as recommended, method.
- *With the NIS+ command set*. Chapter 6, Chapter 7, and Chapter 8 describe how to configure NIS+ using the NIS+ command set. While this method gives you more flexibility than the scripts method, it is more difficult. This method should be used only by experienced NIS+ administrators who need to configure a namespace with characteristics significantly different than those provided by the configuration scripts.

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

See Chapter 22 for information on how to remove an NIS+ directory or domain, an NIS+ server, or the NIS+ namespace.

CHAPTER 3

NIS+ Setup Scripts

This chapter describes the NIS+ scripts and their functionalities.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

About the NIS+ Scripts

Caution – Before running the NIS+ setup scripts, make sure you have performed the steps described in "NIS+ Configuration Overview" on page 83.

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

TABLE 3–1 NIS+ Scripts

NIS+ Script	What It Does
`nisserver`	Sets up the root master, non-root master and replica servers with level 2 security (DES)

TABLE 3–1 NIS+ Scripts *(Continued)*

NIS+ Script	What It Does
`nispopulate`	Populates NIS+ tables in a specified domain from their corresponding system files or NIS maps
`nisclient`	Creates NIS+ credentials for hosts and users; initializes NIS+ hosts and users

What the NIS+ Scripts Will Do

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

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

What the NIS+ Scripts Will Not Do

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

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

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

  See Chapter 4 for how to use the `nisgrpadm` command instead of one of the NIS+ scripts to add extra NIS+ principals to the NIS+ admin group.
- Create private tables
- Run an NIS+ server at any security level other than level 2
- Start the `rpc.nisd` daemon on remote servers, which is required to complete server installation

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

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

CHAPTER **4**

Configuring NIS+ With Scripts

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

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Configuration Overview

Using the configuration scripts is the recommended method of setting up and configuring an NIS+ namespace. Using these scripts is easier than to trying to set up an NIS+ namespace with the NIS+ command set, as described in Chapter 6, Chapter 7, and Chapter 8.

(See the `nisserver`, `nispopulate`, and `nisclient` man pages for complete descriptions of the scripts. See the Glossary for definitions of terms and acronyms you do not recognize.)

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

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

TABLE 4–1 Recommended NIS+ Configuration Procedure Overview

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

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

Configure root domain clients	Configure the client machines. (Some of them will subsequently be converted into servers.) Initialize users as NIS+ clients.	`nisclient`
Enable servers	Enable some clients of the root domain to become servers. Some servers will later become root replicas; others will support lower-level domains.	`svcadm enable`

Configure root replicas	Designate one or more of the servers you just configured as replicas of the root domain.	`nisserver`
		 `svcadm`

Configure non-root domains	Create a new domain. Designate a previously enabled server as its master. Create its admin group and admin credentials.	`nisserver`

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

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

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

NIS+ and the Service Management Facility

Most of the command line administrative tasks associated with the NIS+ service are managed by the Service Management Facility (SMF). For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

- Administrative actions on the NIS+ service, such as enabling, disabling, or restarting, can be performed using the `svcadm` command. When the service is started or stopped, any dependent processes are also started or stopped.

  Tip – Temporarily disabling a service by using the `-t` option provides some protection for the service configuration. If the service is disabled with the `-t` option, the original settings would be restored for the service after a reboot. If the service is disabled without `-t`, the service will remain disabled after reboot.

- SMF automatically starts `nis_cachemgr` when it enables the NIS+ service, if it detects the `/var/nis/NIS_COLD_START` file.
- The NIS+ Fault Managed Resource Identifier (FMRI) is `svc:/network/rpc/nisplus:`*<instance>*.
- The FMRI for `keyserv` is `svc:/network/rpc/keyserv:`*<instance>*.
- You can query the status of NIS+ by using the `svcs` command.
  - Example of `svcs` command and output.

    ```
    # svcs \*nisplus\*
    STATE          STIME    FMRI
    disabled       Sep_01   svc:/network/rpc/nisplus:default
    ```

  - Example of `svcs -l` command and output.

    ```
    # svcs -l network/rpc/nisplus
    fmri         svc:/network/rpc/nisplus:default
    ```

```
enabled      false
state        disabled
next_state   none
restarter    svc:/system/svc/restarter:default
dependency   require_all/none svc:/network/rpc/keyserv (online)
```

You can also use the `svccfg` utility to get more detailed information about a service. See the `svccfg`(1M) man page.

- You can check a daemon's presence by using the `ps` command.

  ```
  # ps -e | grep rpc.nisd
  ```

Note – Do not use the `-f` option with `ps` because this option attempts to translate user IDs to names, which causes more naming service lookups that might not succeed.

Using `svcadm` With `rpc.nisd -x`

In general, the `/usr/sbin/rpc.nisd` daemon is administered using the `svcadm` command. However, when `rpc.nisd` is invoked with `-x nisplusLDAPinitialUpdateOnly=yes`, `rpc.nisd` performs the specified action, then exits. That is, `rpc.nisd` does not daemonize. SMF should not be used in conjunction with `-x nisplusLDAPinitialUpdateOnly=yes`. SMF can be used any other time you want to start, stop, or restart the `rpc.nisd` daemon.

The following example shows `rpc.nisd` used with `-x nisplusLDAPinitialUpdateOnly=yes`.

```
# /usr/sbin/rpc.nisd -m mappingfile \
-x nisplusLDAPinitialUpdateAction=from_ldap \
-x nisplusLDAPinitialUpdateOnly=yes
```

Modifying the `/lib/svc/method/nisplus` File

If you want to include specific options when you invoke the `rpc.nisd` daemon with SMF, add the options to the `/lib/svc/method/nisplus` file. The following list provides some commonly used options.

`-S 0`	Sets the server's security level to 0, which is required at this point for bootstrapping. Because no `cred` table exists yet, no NIS+ principals can have credentials; if you used a higher security level, you would be locked out of the server.
`-B`	Supports DNS forwarding
`-Y`	Starts the NIS+ daemon in NIS-compatibility mode

▼ *How to Modify the `/lib/svc/method/nisplus` File*

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Using Role-Based Access Control (Tasks)" in *System Administration Guide: Security Services*.

2. **Stop the NIS+ service.**

   ```
   # svcadm disable network/rpc/nisplus:default
   ```

3. **Open the `/lib/svc/method/nisplus` file.**

   Use your preferred text editor.

4. **Edit the file to add the desired options.**

   Example—

   Change:

   ```
   /usr/sbin/rpc.nisd $nisd_flags || exit $?
   ```

   To:

   ```
   /usr/sbin/rpc.nisd $nisd_flags -Y -B || exit $?
   ```

   In this example, the `-Y` and `-B` options are added to `rpc.nisd`, so the options are automatically implemented at startup.

5. **Save and quit.**

6. **Start the NIS+ service.**

   ```
   # svcadm enable network/rpc/nisplus:default
   ```

Creating a Sample NIS+ Namespace

The procedures in this chapter show you how to create a sample NIS+ namespace. The sample NIS+ namespace will be created from `/etc` files and NIS maps. This sample shows you how to use the scripts both when your site is not running NIS and when NIS is running at your site. You can set your servers to NIS-compatibility mode if they will be serving NIS clients. See "Solaris 1.x Releases and NIS-Compatibility Mode" on page 49 for more information on NIS-compatibility mode.

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

The sample namespace contains the following components:

- A root master server named `master` for the `doc.com.` domain
- Four clients of the root domain, `doc.com.`:
  - The first client, `client1`, will become a root replica (for the `doc.com.` domain).
  - The second client, `client2`, will become a master server for a new subdomain (for the `sub.doc.com.` domain).
  - The third client, `client3`, will become a non-root replica server of the new subdomain (for the `sub.doc.com.`) domain.
  - The fourth client, `client4`, will remain solely a client of the root domain (`doc.com.`).
- Two clients, `subclient1` and `subclient2`, of the subdomain (`sub.doc.com.`).

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

Summary of NIS+ Scripts Command Lines

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

TABLE 4–2 NIS+ Domains Configuration Command Lines Summary

Action	Machine	Command

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

Optionally, if using DES authentication, select the Diffie-Hellman key length	Server and client machines as superuser	`nisauthconf -dh`*key-length-alg-type* `des`
Create a root master server without or with NIS (YP) compatibility.	Root master server as superuser	`nisserver -r-d`*newdomain.* or `nisserver -Y-r-d` *newdomain.*
Populate the root master server tables from files or from NIS maps.	Root master server as superuser	`nispopulate -F-p` */files* `-d` *newdomain.* or `nispopulate -Y-d` *newdomain.* `-h` *NISservername*`\ -a` *NIS_server_ipaddress* `-y` *NIS_domain*
Add additional users to the NIS+ admin group.	Root master server as superuser	`nisgrpadm-aadmin.`*domain.name.domain.*
Make a checkpoint of the NIS+ database.	Root master server as superuser	`nisping -C` *domain.*
Initialize a new client machine.	Client machine as superuser	`nisclient -i-d` *domain.* `-h` *master1*
Initialize user as an NIS+ client.	Client machine as user	`nisclient -u`
Start the NIS+ service (`rpc.nisd` daemon)—required to convert a client to a server without or with NIS compatibility (and DNS forwarding) .	Client machine as superuser	Modify the `/lib/svc/method/nisplus` file to add the `-Y` or `-B` options as needed, then enable the NIS+ service: `svcadm enable /network/rpc/nisplus`
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*

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

Action	Machine	Command
Populate the new master server tables from files or from NIS maps.	New subdomain master server as superuser	`nispopulate -F-p/`*subdomaindirectory* `-d \` *newsubdomain.domain* . or `nispopulate-Y-d`*newsubdomain .domain*`.-h` *NISservername* `-a`*NIS_server_ipaddress* `-y` *NIS_domain*
Convert a client to a master server replica.	Subdomain master server as superuser	`nisserver-R-d`*subdomain .domain*. `-h` *clientname*
Initialize a new client of the subdomain. Clients can be converted to subdomain replicas or to another server.	New subdomain client machine as superuser	`nisclient -i -d` *newsubdomain.domain*. `-h \` *subdomainmaster*
Initialize user as an NIS+ client.	Client machine as user	`nisclient -u`

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

Setting Up NIS+ Root Servers

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

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

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

Prerequisites to Running `nisserver` to Set Up a Root Server

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

You need the following information before running `nisserver`.

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

TABLE 4–3 Internet Organizational Domains

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

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

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

▼ How to Create a Root Master Server

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

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

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

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

   ```
   nisauthconf dh640-0 des
   ```

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

   ```
   nisauthconf dh640-0
   ```

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

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

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

   "NIS+ group" refers to the group of users who are authorized to modify the information in the `doc.com.` domain. (Domain names always end with a period.) Modification includes deletion. `admin.`*domainname* is the default name of the group. See "How to Change Incorrect Information" on page 94 for instructions on how to change this name.

   "NIS compatibility" refers to whether an NIS+ server accepts information requests from NIS clients. When set to `OFF`, the default setting, the NIS+ server does not fulfill requests from NIS clients. When set to `ON`, an NIS+ server fulfills such requests. You can change the NIS-compatibility setting with this script. See "How to Change Incorrect Information" on page 94.

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

---

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

   Typing `n` causes the script to prompt you for the correct information. (See "How to Change Incorrect Information" on page 94 for what you need to do if you type `n`.)

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

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

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

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

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

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

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

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

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

Tables" on page 96.

▼ How to Change Incorrect Information

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

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

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

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

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

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

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

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

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

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

   Note – If you choose to make this server NIS compatible, you also need to edit a file and restart the rpc.nisd daemon before it will work. See "Configuring a Client as an NIS+ Server" on page 107 for more information.

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

   When the information is correct, continue with Step 3 in "How to Create a Root Master Server" on page 92. You can keep choosing -n until the information is correct.

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

The procedure for setting up a multihomed NIS+ server is the same as setting up a single interface server. The only difference is that there are more interfaces that need to be defined in the hosts database (`/etc/hosts` and `/etc/inet/ipnodes` files, and NIS+ `hosts` and `ipnodes` tables). Once the host information is defined, use the `nisclient` and `nisserver` scripts to set up the multihomed NIS+ server. For information about setting up a multihomed replica server, see "How to Set Up Multihomed NIS+ Replica Servers" on page 112.

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

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

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

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

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

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

   ```
   127.0.0.1 localhost loghost
   192.168.10.x hostA hostA-10 hostA-eri0
   192.168.11.y hostA hostA-11 hostA-eri1
   192.168.12.z hostA hostA-12
   ```

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

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

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

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

Populating NIS+ Tables

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

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

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

Prerequisites to Running `nispopulate` to Populate Root Server Tables

Before you run the `nispopulate` script, be sure the following prerequisites have been met.

- View each local `/etc` file or NIS map from which you will load data. Make sure there are no spurious or incorrect entries. Make sure that the right data is in the correct place and format. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. You can always add individual entries after configuration is completed. That is easier than trying to load incomplete or damaged entries.
- The information in the files must be formatted appropriately for the table into which it will be loaded. Chapter 9 describes the format required for a text file to be transferred into its corresponding NIS+ table.
- Make sure that domain and host names are different. Domains and hosts cannot have the same name. For example, if you have a `sales` domain you cannot have a machine named `sales`. Similarly, if you have a machine named `home`, do not create a domain named `home`. This caution also applies to subdomains. For example, if you have a machine named `west`, do not create a `sales.west.doc.com` subdomain.
- Remove all dots and underscores in host names. NIS+ uses dots (periods) to delimit between machine names and domains and between parent and subdomains, so you cannot have a machine name containing a dot. You also cannot

use underscores in hostnames, since DNS does not allow it. Before running the `nispopulate` script, you must eliminate any dots in your host names. You can convert host name dots to hyphens. For example, you cannot have a machine named `sales.alpha`. You can convert that name to `sales-alpha`.

- If you are setting up a network for the first time, you may not have much network information stored anywhere. In that case, you first need to collect the information, then type it into the *input file*—which is essentially the same as an `/etc` file.
- For safety's sake, you should make copies of the `/etc` files and use the copies to populate the tables instead of the actual ones. (This example uses files in a directory called `/nisplusfiles`, for instance.)
- Edit four of the copied NIS table files, `passwd`, `shadow`, `aliases`, and `hosts`, for security problems, particularly items that you do not want distributed across the namespace. For example, you might want to remove the following lines from the copy of your local `passwd` file so that they are not made available across the namespace:

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

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

Information You Need

If populating from files, you need the following information.

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

If populating from NIS maps, you need:

- The new NIS+ domain name
- The NIS domain name

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

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

▼ How to Populate the Root Master Server Tables

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

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

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

   ---

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

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

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

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

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

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

   The `-Y` option indicates that the tables take their data from NIS maps. The `-d` option specifies the NIS+ domain name. The `-h` option specifies the NIS server's machine name. (In this case, the NIS server's name is `salesmaster`. You have to insert the name of a real NIS server at your site to create the sample domain.) The `-a` option specifies the NIS server's IP address. (In this case, the address is `130.48.58.111`. You have to insert the IP address of a real NIS server at your site to create the sample domain.) The `-y` option specifies the NIS domain name. (In this case, the domain's name is `sales.doc.com.`; you have to insert the NIS domain name of the real NIS domain at your site to create the sample domain.)

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

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

   Typing `n` causes the script to prompt you for the correct information. (See "How to Change Incorrect Information" on page 94 for what you need to do if the information is incorrect.)

   - If you performed substep *a* of Step a, you will see the following:

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

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

   - If you performed substep *b* of Step b, you will see the following:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

You do not have to wait for the other administrators to change their default passwords to perform this step; however, they must already be listed in the `passwd` table before you can add them to the admin group. Members of the admin group will be unable to act as NIS+ principals until they add themselves to the domain. See "How to Initialize an NIS+ User" on page 105 for more information on initializing users. The group cache also has to expire before the new members become active.

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

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

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

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

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

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

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

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

Setting Up NIS+ Client Machines

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

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

Note – The `-i` option used in "How to Initialize a New Client Machine" on page 103 does not configure an NIS+ client to resolve host names requiring DNS. You need to explicitly include DNS for clients in their name service switch files.

Prerequisites to Running `nisclient` to Set Up Client Machines

Before you use the `nisclient` script, be sure the following prerequisites have been met.

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

Information You Need

You need the following information to run `nisclient`.

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

▼ How to Initialize a New Client Machine

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

   On the master server, type

   ```
   nisauthconf
   ```

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

   ```
   dh640dh-0 des
   ```

   type the following command on the client machine

   ```
   nisauthconf dh640dh-0 des
   ```

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

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

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

3. **Type `y`.**

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

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

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

   This example uses the hypothetical address `123.123.123.123`.

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

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

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

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

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

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

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

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

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

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

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

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

Creating Additional Client Machines

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

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

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

Initializing NIS+ Client Users

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

Prerequisites to Running `nisclient` to Initialize a Client User

Before you use the `nisclient` script to initialize a user, be sure the following prerequisites have been met.

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

Information You Need

You need the following information to run `nisclient`.

- A user's login name (`user1` in this example)
- The default Secure RPC password (`nisplus` in this example)
- The login password of the user who will become the NIS+ client

▼ How to Initialize an NIS+ User

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

```
user1prompt% nisclient -u
At the prompt below, type the network password (also known as the
Secure-RPC password) that you obtained either from your administrator
```

```
or from running the nispopulate script.
Please enter the Secure-RPC password for user1:
```

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

   The password does not echo on the screen.

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

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

   The password does not echo on the screen.

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

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

Setting Up NIS+ Servers

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

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

---

Note – You can have only one NIS+ master root server. Root NIS+ servers are a special type of NIS+ server. This section does not describe how to configure a root master server; see "Setting Up NIS+ Root Servers" on page 90 for more information.

You can configure servers any of these different ways:

- Without NIS compatibility
- With NIS compatibility
- With NIS compatibility and DNS forwarding—you only need to set DNS forwarding if you are going to have SunOS 4.x clients in your NIS+ namespace

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.

Prerequisites to Setting Up a NIS+ Server

Before you start the NIS+ service from the command line by using `svcadm`, be sure the following prerequisites have been met.

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

Information You Need

You need the superuser password of the client that you will convert into a server before you can start the NIS+ service by using the `svcadm` command.

Configuring a Client as an NIS+ Server

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

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

▼ How to Configure a Server Without NIS Compatibility

You need to be superuser or establish an equivalent role to perform this procedure.

1. **View the `/lib/svc/method/nisplus` file to verify that the `-Y` option does not appear.**

   See "NIS+ and the Service Management Facility" on page 85 for more information.

2. **Start the NIS+ service.**

   You need to be superuser or establish an equivalent role to perform this step.

   ```
   client1# svcadm enable /network/rpc/nisplus:default
   ```

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

▼ How to Configure a Server With NIS Compatibility

You need to be superuser or establish an equivalent role to perform this procedure.

1. **Edit the `/lib/svc/method/nisplus` file on the server to add the `-Y` option.**

   See "NIS+ and the Service Management Facility" on page 85 for more information.

2. **Start the NIS+ service.**

   You need to be superuser or establish an equivalent role to perform this step.

   ```
   client1# svcadm enable /network/rpc/nisplus
   ```

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

▼ How to Configure a Server With DNS Forwarding and NIS Compatibility

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

You need to be superuser or establish an equivalent role to perform this procedure.

1. **Edit the `/lib/svc/method/nisplus` file on the server to add the `-Y` and `-B` options.**

   See "NIS+ and the Service Management Facility" on page 85 for more information.

2. **Start the NIS+ service.**

   You need to be superuse,r or establish an equivalent role, to perform this step.

   ```
   client1# svcadm enable /network/rpc/nisplus:default
   ```

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

Creating Additional Servers

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

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

Creating a Root Replica Server

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

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

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

See "Creating a Root Replica Server" on page 109 for additional information on how to determine the optimum number of replicas.

"How to Create a Root Replica" on page 110 shows the machine `client1` being configured as a root replica for the `doc.com.` domain. This procedure uses the NIS+ `nisserver` script. (You can also use the NIS+ command set to configure a replica server as described in "Using NIS+ Commands to Configure a Replica Server" on page 161.)

Prerequisites to Running `nisserver` to Create a Root Replica

Before you run `nisserver` to create a replica, be sure the following prerequisites have been met.

- The domain must already have been configured and its master server must be running.
- The tables of the master server must be populated. (At a minimum, the hosts table must have an entry for the new client machine.)
- You must have initialized the new server as a client machine in the domain, as described in "Setting Up NIS+ Client Machines" on page 102.

- You must have started the NIS+ service, `rpc.nisd`, on the new replica server, as described in "Setting Up NIS+ Servers" on page 106.
- You must be logged in as root on the root master server. In this example, the root master machine is named `master1`.

Information You Need

You need the following information to run `nisserver`.

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

▼ How to Create a Root Replica

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

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

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

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

   Typing `n` causes the script to prompt you for the correct information. (See "How to Change Incorrect Information" on page 94 for what you need to do if you type `n`.)

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

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

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

```
Is this information correct? (type 'y' to continue, 'n' to exit this script)
y
The system client1 is now configured as a replica server for domain doc.com..
The NIS+ server daemon, rpc.nisd, must be running on client1 with the proper
options to serve this domain. ...
```

Note – If you want to run this replica in NIS (YP) compatibility mode, modify the `/lib/svc/method/nisplus` file to add the `-Y` option. The file needs modification only if you want the root replica to fulfill NIS client requests and it was not already configured as an NIS-compatible server. See "Configuring a Client as an NIS+ Server" on page 107 for more information about creating NIS-compatible servers and "NIS+ and the Service Management Facility" on page 85 for more information about using Service Management Facility commands with NIS+.

---

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

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

   a. **Stop the NIS+ service.**

   ```
   # svcadm disable /network/rpc/nisplus:<instance>
   ```

   b. **Edit the server's `/lib/svc/method/nisplus` file to add the `-Y` option.**

   c. **Restart the NIS+ service.**

   ```
   # svcadm enable /network/rpc/nisplus:<instance>
   ```

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

   You can do this in two ways:

   - The preferred method of loading data on to a new replica server is to use the NIS+ backup and restore capabilities to back up the master server, then "restore" that data on to the new replica server. This step is described in detail in "How to Load Namespace Data—`nisrestore` Method" on page 164.
   - Run `nisping`. Running `nisping` initiates a full resynch of all NIS+ data from the master server to this new replica. If your namespace is large, this can take a long time, during which your master server is very busy and slow to respond and your new replica is unable to answer NIS+ requests. This step is described in detail in "How to Load Namespace Data—`nisping` Method" on page 166.

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

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

▼ How to Set Up Multihomed NIS+ Replica Servers

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

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

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

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

This procedure shows how to set up any NIS+ non-root master servers. The following example creates a replica for the root domain. For information about setting up a multihomed root server, see "How to Set Up a Multihomed NIS+ Root Master Server" on page 95.

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

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

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

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

   For example:

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

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

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

   For example:

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

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

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

For example:

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

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

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

   For example:

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

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

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

   For example:

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

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

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

Creating a Subdomain

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

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

Prerequisites to Running `nisserver` to Create a Subdomain

Before you run `nisserver` to create a master server for a new non-root domain, be sure the following prerequisites have been met.

- The parent domain must already have been configured and its master server must be running.

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

Information You Need

You need the following information to run `nisserver` to create a master server for a non-root domain.

- The client machine name (`client2,` in this example)
- The superuser password for the parent master server
- A name for the new non-root domain—the name of the new domain includes the name of the parent domain with this syntax: *newdomain.rootdomain*

  In "How to Create a New Non-Root Domain" on page 114, the new non-root domain is called `sub.doc.com.`

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

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

The non-root server's credentials must still be in the domain above it in the hierarchy. Configure the non-root servers as described in "How to Create a New Non-Root Domain" on page 114. Only after the servers are properly configured, can you change the domainname to that of the domain it serves. See the `-k` option of `nisinit` and the `-d` option of `nisserver`.

▼ How to Create a New Non-Root Domain

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

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

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

Master servers of new non-root domains are created with the same set of default values as root servers. See "How to Create a Root Master Server" on page 92 for more information on NIS+ group, NIS compatibility, and security level.

2. **Type y to continue.**

   Typing n causes the script to prompt you for the correct information. (See "How to Change Incorrect Information" on page 94 for what you need to do if you type n.)

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

3. **Type y to continue.**

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

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

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

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

Creating Additional Domains

Repeat the preceding procedure for changing servers to master servers of new non-root domains on as many server machines as you like. Every new master server is a new domain. Plan your domain structure before you start creating an NIS+ namespace. See "Structure of the NIS+ Namespace" on page 55 for more information on planning an NIS+ hierarchy.

Populating the New Subdomain's Tables

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

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

Prerequisites to Running `nispopulate` to Populate a Subdomain's Tables

Before you run the `nispopulate` script to populate the new master server's tables, be sure the following prerequisites have been met.

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

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

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

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

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

Information You Need

The information you need to gather depends upon whether you are populating from files or from NIS maps.

If populating from files, you need the following information.

- 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 following information.

- The new NIS+ domain name
- The NIS domain name

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

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

Populating the Master Server Tables

Since this procedure is essentially the same as the procedure shown in "How to Populate the Root Master Server Tables" on page 98, this example shows you only what you would type to populate the tables of the new `sales.doc.com.` domain. For more information about this procedure, see "How to Populate the Root Master Server Tables" on page 98.

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

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

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

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

How to Populate the Tables From Files

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

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

How to Populate the Tables From NIS Maps

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

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

See "How to Populate the Root Master Server Tables" on page 98 for additional information.

Creating Subdomain Replicas

The same principles that apply to root domain replicas apply to subdomain replicas (see "Creating a Root Replica Server" on page 109).

You use the same procedure to create a subdomain replica as you do to create a root replica. The major difference between creating the root replica and a subdomain replica is that the machine you are going to convert to a subdomain replica remains a client of the domain above the one it serves as a replica. This example shows you only what you type to create a replica for the new domain. For the rest of the script's output, see "How to Create a Root Replica" on page 110.

Prerequisites to Running `nisserver` to Create a Subdomain Replica

Before you run `nisserver` to create a replica, be sure the following prerequisites have been met.

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

Information You Need

You need the following information to run `nisserver`.

- The domain name
- The client machine name (`client3,` in this example)
- The superuser password for the root master server

▼ How to Create a Replica

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

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

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

See "How to Create a Root Replica" on page 110 for the rest of this script's output.

Initializing Subdomain NIS+ Client Machines

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

Note – The `-i` option used in "How to Initialize a New Subdomain Client Machine" on page 121 does not configure an NIS+ client to resolve host names requiring DNS. You need to explicitly include DNS for clients in their name service switch files.

You use the same procedure to initialize a client in the new domain as you do to initialize a client in the root domain. This example shows you only what you would type to initialize a client for the new domain. For the rest of the script's output, see "How to Initialize a New Client Machine" on page 103.

Prerequisites to Running `nisclient` to Initialize a Client Machine

Before you use the `nisclient` script to initialize a client machine, be sure the following prerequisites have been met.

- The domain must have already been configured and its master server must be running.

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

Information You Need

You need the following information to run `nisclient`.

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

▼ How to Initialize a New Subdomain Client Machine

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

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

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

See "How to Initialize a New Client Machine" on page 103 for the rest of this script's output.

Initializing Subdomain NIS+ Client Users

You use the same procedure (`nisclient`) to initialize a user in the new domain as you do to initialize a user in the root domain. All users must make themselves NIS+ clients. This example shows you only what you would type to initialize a user for the new domain. For the rest of the script's output, see "How to Initialize an NIS+ User" on page 105.

Prerequisites to Running `nisclient` to Initialize a Subdomain Client User

Before you use the `nisclient` script to initialize a user, be sure the following prerequisites have been met.

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

Information You Need

You need the following information to run `nisclient`.

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

▼ How to Initialize an NIS+ Subdomain User

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

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

See "How to Initialize an NIS+ User" on page 105 for the rest of this script's output.

Summary of Commands for the Sample NIS+ Namespace

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

TABLE 4–4 Creating the Sample Namespace: Command Summary

Tasks	Commands

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

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

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

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

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

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

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

Populate the new master server's NIS+ tables—from files or from NIS maps.	`client2# nispopulate -F -p /nis+files -d sales.doc.com.`
	or
	`client2# nispopulate -Y -d sales.doc.com. -h \`
	`businessmachine -a 172.31.58.242 -y business.doc.com.`

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

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

CHAPTER 5

Setting Up the Root Domain

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

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release. (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Introduction to Setting Up the Root Domain

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

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

Setting up the root domain involves three major tasks:

- Preparing the root master server
- Creating the root domain
- Creating credentials for the root domain

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

Standard Versus NIS-Compatible Configuration Procedures

The steps in this chapter apply to both a standard NIS+ root domain and an NIS-compatible root domain. There are, however, some important differences. The NIS+ daemon for an NIS-compatible domain must be started with the `-Y` option, which allows the root master server to answer requests from NIS clients. See "NIS+ and the Service Management Facility" on page 85 for more information.

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

Note – The NIS+ service is managed by the Service Management Facility (SMF). Administrative actions on this service, such as enabling, disabling, or restarting, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+. For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

Establishing the Root Domain

The procedure describes each step in detail and provides related information. For those who do not need detailed instructions, a summary listing of the necessary commands is provided on "Root Domain Configuration Summary" on page 139.

Summary of Steps

Here is a summary of the entire configuration process:

1. Log in as superuser to the root master server.
2. Check the root master server's domain name.
3. Check the root master server's switch-configuration file.
4. Optionally, configure the Diffie-Hellman key length.
5. Restart the name service cache (`nscd`), if you made any changes to the `nsswitch.conf` file.
6. Delete `/etc/.rootkey` and restart `keyserv`.
7. Stop the NIS+ service.
8. Clean out leftover NIS+ material and processes.
9. Name the root domain's admin group.
10. Create the root directory and initialize the root master server.
11. [Optional] Edit the `/lib/svc/method/nisplus` file to add desired options.
12. Start the NIS+ daemon.
13. Verify that the root objects were created.
14. Create the root domain's subdirectories and tables.
15. Create DES credentials for the root master server.
16. Create the root domain's admin group.
17. Add the root master to the root domain's admin group.
18. Update the root domain's public keys.
19. Start the NIS+ service (this starts the cache manager and the NIS+ daemon, with security level 2).
20. Add your LOCAL credentials to the root domain.
21. Add your DES credentials to the root domain.
22. Add credentials for other administrators.
23. Add yourself and other administrators to the root domain's admin group.
24. Allocate sufficient swap space.

Establishing the Root Domain—Task Map

TABLE 5–1 Establishing the Root Domain

Task	Description	For Instructions, Go To
Establishing the Root Domain	Use the `domainname` command to establish the root domain. Optionally extend the Diffie-Hellman key length. Stop and start the `ncsd` daemon. Disable and restart `keyserv`. Clean out leftover NIS+ information.	"How to Configure a Root Domain" on page 129

Security Considerations

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

Note – The NIS+ security system is complex. If you are not familiar with NIS+ security, you might want to review Chapter 11 before starting to configure your NIS+ environment.

Prerequisites to Establishing a Root Domain

Before proceeding, make sure that the following prerequisites have been met.

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

Information You Need

In order to complete this task you need to know:

- The superuser password of the machine that will become the root master server
- The name of the root domain

- The name of the root domain's admin group
- Your UID and password
- The UID of any administrator whose credentials you will add to the root domain

▼ How to Configure a Root Domain

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

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

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

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

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

   If the name is not correct, change it.

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

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

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

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

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

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

This command displays the current `nsswitch.conf` file. The primary name service referenced by this file should be `nisplus`. If the root master server's configuration file does not use `nisplus` as the primary name service, exchange it for one that does, as explained in "Selecting a Different Configuration File" on page 38.

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

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

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

5. **If you made any changes at all to the `nsswitch.conf` file, restart the name service cache (`nscd`) daemon.**

   ```
   rootmaster# svcs \*name\*
   online     Jan_12   svc:/system/name-service-cache:default
   rootmaster# svcadm restart system/name-service-cache
   ```

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

   Complete instructions are provided in Chapter 1.

6. **Delete the `/etc/.rootkey` file and restart `keyserv`.**

   ```
   rootmaster# cp /etc/nsswitch.nisplus /etc/nsswitch.conf
   rootmaster# svcs \*keyserv\*
   online     Jan_12   svc:/network/rpc/keyserv:default
   rootmaster# svcadm disable network/rpc/keyserv
   rootmaster# rm -f /etc/.rootkey
   rootmaster# svcadm enable network/rpc/keyserv
   ```

7. **Stop the NIS+ service.**

   ```
   rootmaster# svcs \*nisplus\*
   online     Jan_12   svc:/network/rpc/nisplus:default
   rootmaster# svcadm disable network/rpc/nisplus:default
   rootmaster# svcs \*nisplus\*
   disabled   Jan_12   svc:/network/rpc/nisplus:default
   ```

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

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

   ```
   rootmaster# ls /var/nis
   NIS_COLD_START NIS_SHARED_CACHE
   rootmaster# rm -rf /var/nis/*
   ```

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

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

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

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

   For C Shell

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

   For Bourne or Korn Shell

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

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

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

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

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

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

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

    This is not the root directory object; it is a file that NIS+ uses to describe the root of the namespace for internal purposes. The NIS+ root directory object is created later.

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

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

---

11. **(Optional) Edit the `/lib/svc/method/nisplus` file to add the options you want.**

    You must include the `-S 0` option. Use your preferred text editor.

    See "NIS+ and the Service Management Facility" on page 85 for more information about editing the `/lib/svc/method/nisplus` file.

`-S` 0	Sets the server's security level to 0, which is required at this point for bootstrapping. Because no `cred` table exists yet, no NIS+ principals can have credentials; if you used a higher security level, you would be locked out of the server.
`-B`	Supports DNS forwarding
`-Y`	Starts the NIS+ daemon in NIS-compatibility mode

12. **Start the NIS+ service daemon.**

```
rootmaster# svcadm enable network/rpc/nisplus:default
```

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

    Your namespace should now have:

    - A root directory object (`root.dir`)
    - A root master server (`rootmaster`) running the NIS+ service
    - A cold start file for the master server (`NIS_COLD_START`)
    - A transaction log file (`trans.log`)
    - A table dictionary file (`data.dict`)

The root directory object is stored in the directory created in Step 10. Verify that it is there.

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

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

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

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

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

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

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

```
rootmaster# ps -e | grep rpc.nisd
root   1081  1  61  16:43.33  ?  0:01  rpc.nisd  -S 0
root   1087  1004  11  16:44:09  pts/1  0:00  grep  rpc.nisd
```

The root domain's `NIS_COLD_START` file, which contains the Internet address (and, eventually, public keys) of the root master server, is placed in `/var/nis`. Although there is no NIS+ command that you can use to examine its contents, its contents are loaded into the server's directory cache (`NIS_SHARED_DIRCACHE`). You can examine those contents with the `/usr/lib/nis/nisshowcache` command.

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

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

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

    For standard NIS+ only

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

    NIS-compatible only

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

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

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

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

The root directory now contains two subdirectories.

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

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

15. **Create DES credentials for the root master server.**

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

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

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

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

If you persist and retype the same password, NIS+ will still create the credential. The new password will be stored in `/etc/.rootkey` and be used by the keyserver when it starts up. To give the keyserver the new password right away, run `keylogin -r`, as described in Chapter 12.

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

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

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

16. **Create the root domain's admin group.**

    This step creates the admin group named in Step 9. Use the `nisgrpadm` command with the `-c` option. The example below creates the `admin.doc.com.` group.

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

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

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

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

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

```
rootmaster# nisgrpadm -a admin.doc.com. rootmaster.doc.com.
```

```
Added rootmaster.doc.com. to group admin.doc.com.
```

To verify that the root master is indeed a member of the group, use the `nisgrpadm` command with the `-l` option (see Chapter 17).

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

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

18. **Update the root domain's public keys.**

    Normally, directory objects are created by an NIS+ principal that already has DES credentials. In this case, however, the root master server could not acquire DES credentials until *after* it created the `cred` table (since there was no parent domain in which to store its credentials). As a result, three directory objects—`root`, `org_dir`, and `groups_dir`—do not have a copy of the root master server's public key. (You can verify this by using the `niscat -o` command with any of the directory objects. Look for the public key field. Instructions are provided in Chapter 18.)

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

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

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

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

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

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

19. **Restart the NIS+ service.**

Now that the root master server has DES credentials and the root directory object has a copy of the root master's public key, when you restart the root master, it automatically starts with security level 2.

The Service Management Facility automatically starts `nis_cachemgr` when it enables the NIS+ service, if it detects the `/var/nis/NIS_COLD_START` file.

For an NIS-compatible root domain, be sure to edit the `/lib/svc/method/nisplus` file to add the `-Y` flag. See "NIS+ and the Service Management Facility" on page 85 for more information.

```
rootmaster# svcs \*nisplus\*
online     Jan_12   svc:/network/rpc/nisplus:default
rootmaster# svcadm restart network/rpc/nisplus:default
```

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

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

---

20. **Add your LOCAL credentials to the root domain.**

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

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

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

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

    For more information about the `nisaddcred` command, see Chapter 12.

21. **Add your DES credentials to the root domain.**

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

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

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

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

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

22. **Add credentials for the other administrators.**

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

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

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

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

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

In this case, the first instance of `nisaddent` populates the `passwd` table—except for the password column. The second instance populates the shadow column. Each administrator can later change his or her network password using the `chkey` command. Chapter 12 describes how to do this.

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

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

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

```
Added miyoko.doc.com. to group admin.doc.com.
```

24. **Allocate sufficient swap space to accommodate NIS+ tables.**

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

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

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

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

Root Domain Configuration Summary

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

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

Tasks	Actions

Log in as superuser to `rootmaster`.	`rootmaster% su`
	`Password:`

Check domain name.	`# domainname`
Check Switch file.	`# more /etc/nsswitch.conf`
[Optional] Configure Diffie-Hellman key length.	`# nisauthconf dh640-0 des`
Restart `nscd` if the switch file was modified.	`# svcadm restart /system/name-service-cache`

TABLE 5–2 Setting Up a Root Domain: Action Summary *(Continued)*

Tasks	Actions

Remove `/etc/.rootkey` and restart `keyserv`.	`# svcadm disable /network/rpc/keyserv`
	`# rm -f /etc/.rootkey`
	`# svcadm enable /network/rpc/keyserv`

Stop NIS+ services.	`# svcadm disable /network/rpc/nisplus`
Remove leftover NIS+ material.	`# rm -rf /var/nis*`
Name the admin group.	`# NIS_GROUP=admin.doc.com.; export NIS_GROUP`
Initialize the root master.	`# nisinit -r`
[NIS-compat only] Start the daemon with the `-S 0` and `-Y` options.	Edit the `/lib/svc/method/nisplus` file to add the `-S 0` and `-Y` options, then restart the service, as follows. `# svcadm restart network/rpc/nisplus`
[NIS+ Only] Start daemon with `-S 0`.	Edit the `/lib/svc/method/nisplus` file to add the `-S 0` option, then enable the service, as follows. `# svcadm enable network/rpc/nisplus`

Verify creation of root objects.	`# ls -l /var/nis/data`
	`# niscat -o doc.com.`

Create `org_dir` and `groups_dir` tables.	`# /usr/lib/nis/nissetup [-Y]`

Create DES credentials for root master.	`# nisaddcred des`
	`Enter login password:`

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

Update root directory's keys. Update `org_dir`'s keys. Update `groups_dir`'s keys.	`# /usr/lib/nis/nisupdkeys doc.com.`
	`# /usr/lib/nis/nisupdkeys org_dir.doc.com.`
	`# /usr/lib/nis/nisupdkeys groups_dir.doc.com.`

Restart the NIS+ service.	`# svcadm restart network/rpc/nisplus`
Add your LOCAL credentials.	`# nisaddcred -p 11177 -P topadmin.doc.com. local`

Add your DES credentials.	`# nisaddcred -p unix.11177@doc.com -P topadmin.doc.com. des`
	`Enter login password:`

Add credentials for other admins.	`# nisaddcred ...`

TABLE 5–2 Setting Up a Root Domain: Action Summary *(Continued)*

Tasks	Actions
Add other admins to `admin` group.	`# nisgrpadm -a admin.doc.com` *members*
Allocate swap space.	`# /usr/lib/nis/nisstat`

CHAPTER 6

Configuring NIS+ Clients

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

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Introduction to Setting Up NIS+ Clients

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

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

At Step 11 in the client configuration instructions you must choose which of three methods to use: broadcast, host name, or cold-start file. Because each method is implemented differently, each has its own task description. After initializing a client by one of these methods, you can continue setting up the client by returning to Step 12.

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

Configuring the Client

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

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

Setting up an NIS+ client involves the following tasks:

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

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

1. Logging in to the domain's master server
2. Creating DES credentials for the new client machine
3. Ascertaining the Diffie-Hellman key length used on the master server
4. Logging in as superuser to the client
5. Assigning the client its new domain name
6. Stopping and restarting `nscd`
7. Checking the client's `nsswitch.conf` file
8. Setting the client's Diffie-Hellman key
9. Cleaning out leftover NIS+ material and processes
10. Initializing the client
11. Removing the `/etc/.rootkey` file and restarting the `keyserv` daemon
12. Running `keylogin`

13. Rebooting the client

Security Considerations When Configuring the Client

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

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

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

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

You can change the directory object's access rights, provided you have modify rights to it yourself, by using the `nischmod` command, described in Chapter 15.

Prerequisites to Configuring the Client's Credentials

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

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

The client must have:

- Read rights to the directory object of its home domain.
- The client's home domain must already be configured and running NIS+.
- An entry in either the master server's `/etc/hosts` or `/etc/inet/ipnodes` file or in its domain's hosts or ipnodes table.

- A unique machine name that does duplicate any user ID.
- A machine name that does not contain any dots. (For example, a machine named `sales.alpha` is not allowed; a machine named `sales-alpha` is allowed.)

Information You Need

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

Configuring the Client—Task Map

Note – The NIS+ service is managed by the Service Management Facility (SMF). Administrative actions on this service, such as enabling, disabling, or restarting, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+. For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

TABLE 6–1 Configuring the Client

Task	Description	For Instructions, Go To
Configuring the Client	Create credentials fpr the client. Prepare the client machine and initialize it as an NIS+ client.	"How to Configure an NIS+ Client" on page 146

▼ How to Configure an NIS+ Client

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

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

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

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

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

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

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

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

For more information about the `nisaddcred` command, see Chapter 12.

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

   For example:

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

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

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

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

   See "Changing a Machine's Domain Name" on page 149 for information on how to assign (or change) a client's domain name, then return to Step 6.

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

   Make sure the client is using an NIS+ version of the `nsswitch.conf` file. This ensures that the primary source of information for the client will be NIS+ tables. See Example 1–1 for a description of an NIS+ switch file.

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

   ```
   client1# svcadm restart /system/name-service-cache
   ```

   (You do not need to stop and restart the keyserver at this point, as you will do so in Step 12.)

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

   For example:

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

9. **Stop the NIS+ service.**

   ```
   client1# svcadm disable network/rpc/nisplus:default
   client1# svcs \*nisplus\*
   disabled   Jan_12   svc:/network/rpc/nisplus:default
   ```

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

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

```
client1# ls /var/nis
NIS_COLD_START NIS_SHARED_CACHE
client1# rm -rf /var/nis/*
```

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

11. **Initialize the client.**

    You can initialize a client in three different ways: by host name, by cold-start file, or by broadcast. Choose and perform one of those methods. After initializing the client, proceed with Step 12.

12. **Delete the `/etc/.rootkey` file and restart the `keyserv` daemon.**

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

```
client1# cp /etc/nsswitch.nisplus /etc/nsswitch.conf
client1# svcs \*keyserv\*
online     Jan_12   svc:/network/rpc/keyserv:default
client1# svcadm disable network/rpc/keyserv
client1# rm -f /etc/.rootkey
client1# svcadm enable network/rpc/keyserv
```

Setting Up DNS Forwarding

▼ To enable DNS forwarding capabilities on an NIS+ client:

1. **Login as superuser.**
2. **Properly configure the `hosts` line in the `/etc/resolve.conf` file to read: `hosts:nisplus dns files`.**

In this implementation of NIS, if a `/etc/resolve.conf` file exists on the server, `ypstart` *automatically* starts the `ypserv` daemon with the `-d` option to forward requests to DNS.

Changing a Machine’s Domain Name

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

Security Considerations When Changing a Machine’s Domain Name

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

Information You Need

- The machine’s superuser password
- The new domain name

Changing a machine’s Domain—Task Map

TABLE 6–2 Configuring the Client

Task	Description	For Instructions, Go To
Changing a machine’s Domain	Use the `domainname` command to change the client machine domain	“How to Change a Client’s Domain Name” on page 149

▼ How to Change a Client’s Domain Name

1. **Log in to the machine and become superuser.**

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

   ```
   client1% su
   Password:
   ```

2. **Change the machine’s domain name.**

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

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

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

3. **Verify the result.**

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

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

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

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

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

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

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

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

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

Initializing an NIS+ Client

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

- Broadcast method (see "Broadcast Initialization" on page 150)
- Host-name method (see "Initializing a Client—Host Name Method" on page 151)
- Cold-start file method (see "Initializing a Client—Cold-Start Method" on page 153)

Broadcast Initialization

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

This is the simplest way to configure a client but is also the least secure. The NIS+ server that responds to the broadcast sends the client all the information that the client needs in its cold-start file, including the server's public key. Presumably, only an NIS+

server will respond to the broadcast. However, the client has no way of knowing whether the machine that responded to the broadcast is indeed a trusted server. As a result, this method is only recommended for sites with small, secure networks.

Security Considerations When Initializing a Client

You must perform this task as superuser on the client.

Prerequisites to Initializing a Client

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

Information You Need

You need the superuser password to the client.

Initializing an NIS+ Client—Task Map

TABLE 6–3 Initializing an NIS+ Client

Task	Description	For Instructions, Go To
Initializing an NIS+ Client	Use `nisclient` command to initialize an NIS+ client	"How to Configure an NIS+ Client" on page 146

How to Initialize a Client—Broadcast Method

● **Initialize the client.**

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

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

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

Initializing a Client—Host Name Method

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

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

Security Considerations When Initializing a Client—Host Name Method

You must perform this operation as superuser on the client.

Prerequisites to Initializing a Client—Host Name Method

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

Information You Need

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

Initializing an NIS+ Client (Host Name Method)—Task Map

TABLE 6–4 Initializing an NIS+ Client—Host Name Method

Task	Description	For Instructions, Go To
Initializing a Client by Host Name	Use `nisinit` command to initialize an NIS+ client by host name.	"How to Initialize a Client—Host Name Method" on page 152

▼ How to Initialize a Client—Host Name Method

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

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

2. **Initialize the client.**

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

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

```
All done.
```

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

Initializing a Client—Cold-Start Method

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

Security Considerations When Initializing a Client—Cold-Start Method

You must perform this task as superuser on the client.

Prerequisites to Initializing a Client—Cold-Start Method

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

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

Information You Need

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

Initializing an NIS+ Client (Cold-Start Method)—Task Map

TABLE 6–5 Initializing an NIS+ Client—Cold-Start Method

Task	Description	For Instructions, Go To
InitializingClient via Cold-Start File	Use `nisinit` command to initialize an NIS+ client via a cold-start file	"How to Initialize a Client—Cold-Start Method" on page 153

▼ How to Initialize a Client—Cold-Start Method

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

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

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

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

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

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

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

NIS+ Client Configuration Summary

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

TABLE 6–6 Setting Up a Client: Command Summary

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

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

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

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

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

Remove the `/etc/.rootkey file` and restart the keyserver.	`client1# svcadm disable network/rpc/keyserv`
	`client1# rm -f /etc/.rootkey`
	`client1# svcadm enable network/rpc/keyserv`

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

CHAPTER 7

Configuring NIS+ Servers

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

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Setting Up an NIS+ Server

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

Note – The NIS+ service is managed by the Service Management Facility (SMF). Administrative actions on this service, such as enabling, disabling, or restarting, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+. For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

Standard Versus NIS-Compatible Configuration (NIS+ Server)

The differences between setting up an NIS-compatible and a standard NIS+ server are the same as the differences between setting up standard and NIS-compatible root master servers (see "Standard Versus NIS-Compatible Configuration Procedures" on page 126). The server must have a properly configured `/etc/resolv.conf` file. In addition, the NIS+ daemon for an NIS-compatible server must be started with the `-Y` option (and the `-B` option for DNS forwarding), which allows the server to answer requests from NIS clients. For information about implementing the `-Y` and `-B` options, refer to "NIS+ and the Service Management Facility" on page 85.

Note – Whenever `rpc.nisd` is started with either the `-Y` or `-B` option, a secondary daemon named `rpc.nisd_resolv` is spawned to provide name resolution.

Here is a summary of the entire configuration process:

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

Security Considerations When Configuring NIS+ Servers

Note – The NIS+ security system is complex. If you are not familiar with NIS+ security, you might want to review Chapter 11 before starting to configure your NIS+ environment.

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

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

Prerequisites to Configuring NIS+ Servers

- The root domain must already be configured (see Chapter 5).
- The server must have already been initialized as an NIS+ client (see Chapter 6).
- To configure a server you must be logged in as superuser on that machine.
- For the server to run in NIS-compatibility mode and support DNS forwarding, it must have a properly configured `/etc/resolv.conf` file.

Information You Need

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

▼ How to Configure an NIS+ Server

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

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

   The following steps assume that you rebooted the machine after you set it up as an NIS+ client, as instructed in "Configuring the Client" on page 144. Rebooting starts the cache manager, which is a recommended prerequisite to the following step. If you did not reboot the machine, restart the NIS+ service now by using `svcadm`.

2. **(Optional) Edit the `/lib/svc/method/nisplus` file to add the options you want.**

   Use your preferred text editor.

   See "NIS+ and the Service Management Facility" on page 85 for more information about editing the `/lib/svc/method/nisplus` file.

`-B`	Supports DNS forwarding
`-Y`	Starts the NIS+ daemon in NIS-compatibility mode

3. **Start the NIS+ daemon.**

   ```
   server# svcadm enable network/rpc/nisplus:default
   ```

   To verify that the NIS+ service is running, use the `svcs` command.

   ```
   server# svcs \*nisplus\*
   STATE        STIME    FMRI
   online       Jan_12   svc:/network/rpc/nisplus:default
   ```

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

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

The trans.log file is a transaction log. You can examine the contents of the transaction log by using the nislog command, described in "The nislog Command" on page 339.

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

Now this server is ready to be designated a master or replica of a domain, as described in Chapter 8. This step completes this task. A task summary is provided on "Server Configuration Summary" on page 167.

Adding a Replica to an Existing Domain

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

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

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

See "Creating a Root Replica Server" in *System Administration Guide: Naming and Directory Services (NIS+)* for more information on replica distribution and on how to determine the optimum number of replicas. To add a replica to an existing domain you must first configure the new replica, then load the NIS+ data set for your namespace.

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

- *Scripts*. You can use the `nisserver` script, as described in "Creating a Root Replica Server" on page 109. This method automatically performs a full re-sync to load the NIS+ data set on to the new replica server. This is the preferred method because it is easiest, but it might be slower than using the NIS+ command set and backup/restore.
- *NIS+ command set*. You can use the NIS+ command set to configure a replica, as described in "Using NIS+ Commands to Configure a Replica Server" on page 161. This requires more knowledge of NIS+ than using `nisserver`. One advantage of this method is that it gives you the maximum amount of control and monitoring. Another advantage is that you can bring up a replica by manually creating the domain directories, then loading the NIS+ data set using `nisbackup` and `nisrestore`. Using the NIS+ backup and restore capability loads data faster than that used by `nisserver`.

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

  - `nisping`. When you configure a new replica server with either the `nisserver` script or the NIS+ command set, the master server automatically begins to load the namespace data set on to the new replica over the network using `nisping`. If your namespace is large, this could take a long time, during which requests for naming information could be delayed. See "Using `nisping` to Load Data Onto a Replica Server" on page 165 for details.
  - *Backup and restore*. You can interrupt the transfer of data via `nisping`, and use the NIS+ backup and restore capabilities to load your namespace data on to a newly configured replica server, as described in "Using `nisrestore` to Load Data Onto a Replica Server" on page 163. This is the preferred method because the replica's data set is downloaded on to the replica, which is much faster than having the master transfer the data set to the replica over the network.

Using NIS+ Commands to Configure a Replica Server

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

Security Considerations When Configuring a Replica Server

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

Prerequisites to Configuring a Replica Server

- The domain must have already been configured and have a master server up and running.
- The new replica server must already be configured as an NIS+ server, as described in "Setting Up an NIS+ Server" on page 157.

Information You Need

- Name of the server
- Name of the domain

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

TABLE 7–1 Using NIS+ Commands to Configure a Replica Server

Task	Description	For Instructions, Go To
Setting Up an NIS+ Server	Use NIS+ commands to set up an NIS+ server	"How to Configure a Replica Server With NIS+ Commands" on page 162

▼ How to Configure a Replica Server With NIS+ Commands

In this example, the master server is named `master1`, and the new replica is named `replica2`.

1. **Log in to the domain's master server.**
2. **Make sure the NIS+ serive is running.**

```
master1# svcs -l network/rpc/nisplus:default
```

3. **Add the replica to the domain.**

   Run the `nismkdir` command with the `-s` option. The example below adds the replica machine named `replica2` to the `doc.com.` domain.

```
master1# nismkdir -s replica2 doc.com.
master1# nismkdir -s replica2 org_dir.doc.com.
```

```
master1# nismkdir -s replica2 groups_dir.doc.com.
```

When you run the `nismkdir` command on a directory object that already exists, it does not recreate the directory but modifies it, according to the flags you provide. In this case, the `-s` flag assigns the domain an additional replica server. You can verify that the replica was added by examining the directory object's definition, using the `niscat -o` command.

Caution – Never run `nismkdir` on the replica machine. Running `nismkdir` on a replica creates communications problems between the master and the replicas.

Your new replica is now configured. You can now load your NIS+ data set on to the replica. You can do this in two ways:

- `nisping`. If you do nothing, your master server will use the `nisping` command to load your namespace data on to your newly configured replica server. If your namespace is large, this process can take hours. During this process, requests for naming information can be delayed. See "Using `nisping` to Load Data Onto a Replica Server" on page 165 for details.
- *Backup and restore*. You can interrupt the transfer of data via `nisping` and use the NIS+ backup and restore capabilities to load your namespace data on to a newly configured replica server, as described in "Using `nisrestore` to Load Data Onto a Replica Server" on page 163. Because it is so much faster and more efficient, this is the preferred method.

Using `nisrestore` to Load Data Onto a Replica Server

This section describes how to use the NIS+ backup and restore utilities to load namespace data onto a newly configured replica. This is the preferred method of loading data on to a replica.

Security Considerations When Loading Data on a Replica Server By Using `nisrestore`

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

Prerequisites to Loading Data By Using `nisrestore`

- The domain must have already been configured and have a master server up and running.

- The new replica server must already be configured as an NIS+ server, as described in "Setting Up an NIS+ Server" on page 157.
- The new replica server must be configured as a replica, as described in "Using NIS+ Commands to Configure a Replica Server" on page 161.

Using `nisrestore` to Load Data on to a Replica Server — Task Map

TABLE 7–2 Using `nisrestore` to Load Data on to a Replica Server

Task	Description	For Instructions, Go To
Using `nisrestore` to Load Data on to a Replica Server	Use the `nisrestore` command to load data on to a replica server	"How to Load Namespace Data—`nisrestore` Method" on page 164

▼ How to Load Namespace Data—`nisrestore` Method

In this example, the master server is named `master1`, and the new replica is named `replica2`.

1. **Stop the NIS+ service on the new replica server.**

   This interrupts the automatic transfer of namespace data from the master to the replica with the `nisping` command.

   ```
   replica2# svcadm disable /network/rpc/nisplus:default
   ```

2. **Perform an NIS+ backup on the master server.**

   This step is described in more detail in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*. The example below shows how to use the `nisbackup` command to backup up the `master1` server to the `/var/master1_bakup` directory.

   ```
   master1# nisbackup -a /var/master1_bakup
   ```

   The most convenient method of using `nisrestore` to configure a new replica is to back up the master's data to an NFS mounted directory that the new replica can access. This example assumes that both the master and the new replica server have access to the `/var/master1_bakup` directory.

   Another method is to use the `tar` command to copy the data from the `/var/master1_bakup` directory to some transferable storage media, such as a tape cartridge, then copy the data from storage media into a directory mounted on the new replica, then use that directory as the source for the `nisrestore` command, as described in Step 3.

3. **Download the NIS+ data set onto the new replica using the `nisrestore` command.**

   This step is described in more detail in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*. The example below shows how to use the `nisrestore` command to down load NIS+ data on to the `client2` replica from the `/var/master1_bakup` directory.

   ```
   replica2# nisrestore -a /var/master1_bakup
   ```

   If the replica you are creating is for the root domain, or if you get an error message that `nisrestore` cannot verify or look up needed data, then use the `nisrestore -f` option. For example:

   ```
   replica2# nisrestore -f -a /var/master1_bakup
   ```

4. **Start the NIS+ service on the new replica.**

   See "How to Configure an NIS+ Server" on page 159 for details.

Using `nisping` to Load Data Onto a Replica Server

This section describes how to use the `nisping` command to load namespace data onto a newly configured replica. In most cases, it is not necessary to actually run the `nisping` command because the process should begin automatically.

The problem with the `nisping` method is that it requires a full resync of data from the master to the replica over the network using NIS+ protocols. If your namespace is large, this process can take hours, during which requests for naming information can be delayed.

Security Considerations When Loading Data By Using `nisping`

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

Prerequisites to Loading Data By Using `nisping`

- The domain must have already been configured and have a master server up and running.
- The new replica server must already be configured as an NIS+ server, as described in "Setting Up an NIS+ Server" on page 157.
- The new replica server must by configured as a replica, as described in "Using NIS+ Commands to Configure a Replica Server" on page 161.

Using `nisping` to Load Data on to a Replica Server — Task Map

TABLE 7–3 Using `nisping` to Load Data on to a Replica Server

Task	Description	For Instructions, Go To
Using `nisping` to Load Data on to a Replica Server	Use `nisping` to load data on to a replica server	"How to Load Namespace Data—`nisping` Method" on page 166

▼ How to Load Namespace Data—`nisping` Method

Normally, the loading for namespace data is automatically initiated by the master server. If that does not occur, run the `nisping` command as described below.

- **Run `nisping` on the directories**

This step sends a message (a "ping") to the new replica, telling it to ask the master server for an update. If the replica does not belong to the root domain, be sure to specify its domain name. (The example below includes the domain name only for completeness. Since the example used throughout this task adds a replica to the root domain, the `doc.com.` domain name in the example below is not necessary.)

```
master1# nisping doc.com.
master1# nisping org_dir.doc.com.
master1# nisping groups_dir.doc.com.
```

You should see results similar to these:

```
master1# nisping doc.com.
Pinging replicas serving directory doc.com. :
Master server is master1.doc.com.
 No last update time
Replica server is replica1.doc.com.
 Last update seen was Wed Nov 18 11:24:32 1992
 Pinging ... replica2.doc.com.
```

If your namespace is large, this process can take a significant amount of time. For more information about `nisping`, see Chapter 18.

Server Configuration Summary

Table 7–5 and Table 7–4 provide a summary of the tasks described in this chapter. They assume the simplest cases, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. This summary does not show the server's responses to each command.

TABLE 7–4 Adding a Replica Named `replica2` to `doc.com.`: Command Summary

Tasks	Commands
Log in as superuser to domain master.	`master1% su`

Designate the new replica.	`# nismkdir -s replica2 doc.com.`
	`# nismkdir -s replica2 org_dir.doc.com.`
	`# nismkdir -s replica2 groups_dir.doc.com.`

Ping the replica.	`# /usr/lib/nis/nisping doc.com.`
	`# /usr/lib/nis/nisping org_dir.doc.com.`
	`# /usr/lib/nis/nisping groups_dir.doc.com.`

Note – Rather than using `nisping` to transfer data to the new replica, as shown in the example above, an easier method is to use the NIS+ backup and restore capability, as described in "Using `nisrestore` to Load Data Onto a Replica Server" on page 163.

TABLE 7–5 Starting Up a Non-root Master Server: Command Summary

Tasks	Commands
Log in to the server as root.	`server% su`
[NIS-compat only] Start the daemon with `-Y`; include `-B` if you want DNS forwarding.	Edit the `/lib/svc/method/nisplus` file to add the desired options, then restart the service, as follows. `# svcadm restart network/rpc/nisplus`
[NIS+ Only] Start the daemon.	`# svcadm enable network/rpc/nisplus`

CHAPTER **8**

Configuring a Non-Root Domain

This chapter provides step-by-step instructions for using the NIS+ command set to configure a subdomain domain (also known as a non-root domain) including designating its master and replica servers.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Setting Up a Non-Root Domain

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

You should not configure a non-root domain until *after* you have configured its servers.

Setting up a non-root domain involves the following tasks:

- Establishing security for the domain
- Creating the domain's directories
- Creating the domain's tables

- Designating the domain's servers

As with setting up the root domain, these tasks cannot be performed sequentially. To make the configuration process easier to execute, they have been broken down into individual steps and the steps have been arranged into the most efficient order.

Standard Versus NIS-Compatible Configuration (Non-Root Domain)

The differences between NIS-compatible and standard NIS+ servers in subdomains are the same as they are for servers in the root domain (see "Standard Versus NIS-Compatible Configuration Procedures" on page 126).

The NIS+ daemon for each server in an NIS-compatible domain should have been started with the `-Y` option, as instructed in Chapter 7. An NIS-compatible domain also requires its tables to provide read rights for the nobody class, which allows NIS clients to access the information stored in them. This is accomplished with the `-Y` option to the `nissetup` command, shown in Step 4. (The standard NIS+ domain version uses the same command but without the `-Y` option, so it is described in the same step.)

Here is a summary of the entire configuration process:

1. Log in to the domain's master server.
2. Name the domain's administrative group.
3. Create the domain's directory and designate its servers.
4. Create the domain's subdirectories and tables.
5. Create the domain's admin group.
6. Assign full group access rights to the directory object.
7. Add the servers to the domain's admin group.
8. Add credentials for other administrators.
9. Add the administrators to the domain's admin group.

Security Considerations When Setting Up a Non-Root Domain

Note – The NIS+ security system is complex. If you are not familiar with NIS+ security, you might want to review Chapter 17 before starting to configure your NIS+ environment.

At most sites, to preserve the security of the parent domain, only the parent domain's master server or an administrator who belongs to the parent domain's admin group is allowed to create a domain beneath it. Although this is a policy decision and not a requirement of NIS+, the instructions in this chapter assume that you are following that policy. Of course, the parent domain's admin group must have create rights to the parent directory object. To verify this, use the `niscat -o` command.

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

If you are more concerned about convenience than security, you can make the new domain's master server a member of its parent domain's admin group, then perform the entire procedure from the server. Use the `nisgrpadm` command, described in Chapter 17.

Prerequisites to Setting Up a Non-Root Domain

- The parent domain must be configured and running.
- The server that will be designated as this domain's master must be initialized and running NIS+.
- If you will designate a replica server, the master server must be able to obtain the replica's IP address through its `/etc/hosts` or `/etc/inet/ipnodes` file or from its NIS+ hosts table.

Information You Need

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

Setting Up a Non-Root Domain—Task Map

TABLE 8–1 Setting Up a Non-root Domain

Task	Description	For Instructions, Go To
Setting Up a Non-root Domain	Use NIS+ commands to set up a non-root domain	"How to Configure a Non-Root Domain" on page 172

▼ How to Configure a Non-Root Domain

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

   Log in to the server that you will designate as the new domain's master. The steps in this task use the server named `smaster`, which belongs to the `doc.com.` domain, and will become the master server of the `sales.doc.com.` subdomain. The administrator performing this task is `nisboss.doc.com.`, a member of the `admin.doc.com.` group. That group has full access rights to the `doc.com.` directory object.

2. **Name the domain's administrative group.**

   Although you won't actually create the admin group until Step 5, you need to identify it now. This enables the `nismkdir` command used in the following step to create the directory object with the proper access rights for the group. It does the same for the `nissetup` utility used in Step 4.

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

   For C Shell

   ```
   smaster# setenv NIS_GROUP admin.sales.doc.com.
   ```

   For Bourne or Korn Shell

   ```
   smaster# NIS_GROUP=admin.sales.doc.com.
   smaster# export NIS_GROUP
   ```

3. **Create the domain's directory and designate its servers.**

   The `nismkdir` command, in one step, creates the new domain's directory and designates its supporting servers. It has the following syntax:

   ```
   nismkdir -m master -s replica domain
   ```

   The `-m` flag designates its master server, and the `-s` flag designates its replica.

   ```
   smaster# nismkdir -m smaster -s salesreplica sales.doc.com.
   ```

Caution – Always run nismkdir on the master server. Never run nismkdir on the replica machine. Running nismkdir on a replica creates communications problems between the master and the replica.

The directory is loaded into /var/nis. Use the niscat -o command to view it (do not use cat or more).

```
smaster# niscat -o sales.doc.com.
Object Name : Sales
Owner : nisboss.doc.com.
Group : admin.sales.doc.com.
Domain : doc.com.
Access Rights : ----rmcdr---r---
.
```

Unlike the root directory, this directory object *does* have the proper group assignment. As a result, you won't have to use nischgrp.

4. **Create the domain's subdirectories and tables.**

   This step adds the org_dir and groups_dir directories and the NIS+ tables beneath the new directory object. Use the nissetup utility, but be sure to add the new domain name. And, for an NIS-compatible domain, include the -Y flag.

   NIS compatible

   ```
   smaster# /usr/lib/nis/nissetup -Y sales.doc.com.
   ```

   NIS+

   ```
   smaster# /usr/lib/nis/nissetup sales.doc.com.
   ```

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

   ```
   smaster# /usr/lib/nis/nissetup
   org_dir.sales.doc.com. created
   groups_dir.sales.doc.com. created
   auto_master.org_dir.sales.doc.com. created
   auto_home.org.dir.sales.doc.com. created
   bootparams.org_dir.sales.doc.com. created
   cred.org_dir.sales.doc.com. created
   ethers.org_dir.sales.doc.com. created
   group.org_dir.sales.doc.com. created
   hosts.org_dir.sales.doc.com. created
   mail_aliases.org_dir.sales.doc.com. created
   sendmailvars.org_dir.sales.doc.com. created
   netmasks.org_dir.sales.doc.com. created
   netgroup.org_dir.sales.doc.com. created
   networks.org_dir.sales.doc.com. created
   passwd.org_dir.sales.doc.com. created
   protocols.org_dir.sales.doc.com. created
   rpc.org_dir.sales.doc.com. created
   services.org_dir.sales.doc.com. created
   timezone.org_dir.sales.doc.com. created
   ```

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

You can verify the existence of the `org_dir` and `groups_dir` directories by looking in your master server's `/var/nis/data` directory. They are listed along with the root object and other NIS+ tables. The tables are listed under the `org_dir` directory. You can examine the contents of any table by using the `niscat` command, described in Chapter 9 (although at this point the tables are empty).

5. **Create the domain's admin group.**

   This step creates the admin group named in Step 2. Use the `nisgrpadm` command with the `-c` option. This example creates the `admin.sales.doc.com.` group

   ```
   smaster# nisgrpadm -c admin.sales.doc.com.
    Group admin.sales.doc.com. created.
   ```

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

6. **Assign full group access rights to the directory object.**

   By default, the directory object grants only its group read access, which makes the group no more useful than the world class. To make the configuration of clients and subdomains easier, change the access rights that the directory object grants its group from read to read, modify, create, and destroy. Use the `nischmod` command.

   ```
   smaster# nischmod g+rmcd sales.doc.com.
   ```

   Complete instructions for using the `nischmod` command are provided in Chapter 15.

7. **Add the servers to the domain's admin group.**

   At this point, the domain's group has no members. Add the master and replica servers, using the `nisgrpadm` command with the `-a` option. The first argument is the group name; the others are the names of the new members. This example adds `smaster.doc.com.` and `salesreplica.doc.com.` to the `admin.sales.doc.com.` group:

   ```
   smaster# nisgrpadm -a admin.sales.doc.com. smaster.doc.com.
   salesreplica.doc.com.
   Added smaster.doc.com. to group admin.sales.doc.com.
   Added salesreplica.doc.com. to group admin.sales.doc.com.
   ```

   To verify that the servers are indeed members of the group, use the `nisgrpadm` command with the `-l` option (see Chapter 17).

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

```
No implicit nonmembers
No recursive nonmembers
```

8. **Add credentials for other administrators.**

   Add the credentials of the other administrators who will work in the domain.

   For administrators who already have DES credentials in another domain, add LOCAL credentials. Use the `nisaddcred` command with both the `-p` and the `-P` flags.

   ```
   smaster# nisaddcred -p 33355 -P nisboss.doc.com. local
   ```

   For administrators who do not yet have credentials, you can proceed in two different ways.

   - One way is to ask them to add their own credentials. However, they will have to do this as superuser. Here is an example in which an administrator with a UID of 22244 and a principal name of `juan.sales.doc.com.` adds his own credentials to the `sales.doc.com.` domain.

     ```
     smaster# nisaddcred -p 22244 -P juan.sales.doc.com. local
     smaster# nisaddcred -p unix.22244@sales.doc.com -P \
     juan.sales.doc.com. des
     Adding key pair for unix.22244@sales.doc.com.
     Enter login password:
     ```

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

     ```
     smaster# nisaddcred -p 22244 -P juan.sales.doc.com. local
     smaster# nisaddcred -p unix.22244@sales.doc.com -P \
     juan.sales.doc.com. des
     Adding key pair for unix.22244@sales.doc.com.
     Enter juan's login password:
     nisaddcred: WARNING: password differs from login passwd.
     Retype password:
     ```

   Each administrator can later change his or her network password by using the `chkey` command. Chapter 12 and Chapter 13 describe how to do this.

   Note – In the two examples shown in Step 8, the domain name following the lower case `-p` flag must *never* end in a trailing dot, while the domain name following the upper case `-P` flag must *always* end in a trailing dot.

   ---

9. **Add the administrators to the domain's admin group.**

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

```
smaster# nisgrpadm -a admin.sales.doc.com. juan.sales.doc.com.
Added juan.sales.doc.com. to group admin.sales.doc.com.
```

10. **Allocate sufficient swap space to accommodate NIS+ tables.**

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

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

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

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

Subdomain Configuration Summary

Table 8–2 is a summary of the steps required to configure a non-root domain. It assumes the simplest case, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. This summary does not show the server's responses to each command.

TABLE 8–2 Setting Up a Subdomain Command Summary

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

Name the domain's admin group.	`# NIS_GROUP=admin.sales.doc.com.`
	`# export NIS_GROUP`

Create the domain's directory and designate its servers.	`# nismkdir -m smaster -s salesreplica sales.doc.com.`

TABLE 8–2 Setting Up a Subdomain Command Summary *(Continued)*

Tasks	Commands
Create `org_dir`, `groups_dir`, and tables. (For NIS-compatibility, use `-Y`.)	`# /usr/lib/nis/nissetup sales.doc.com.`
Create the admin group.	`# nisgrpadm -c admin.sales.doc.com.`
Assign full group rights to the domain's directory.	`# nischmod g+rmcd sales.doc.com.`
Add servers to admin group.	`# nisgrpadm -a admin.sales.doc.com. smaster.doc.com. sreplica.doc.com.`

Add credentials for other admins.	`# nisaddcred -p 22244 -P juan.sales.doc.com. local`
	
	`# nisaddcred -p unix.22244@sales.doc.com. juan.sales.doc.com. DES`

Add admins to domain's admin group.	`# nisgrpadm -a admin.sales.doc.com. juan.sales.doc.com.`

CHAPTER 9

Setting Up NIS+ Tables

This chapter provides step-by-step instructions for using the NIS+ command set to populate NIS+ tables on a master server from `/etc` files or NIS maps, how to transfer information back from NIS+ tables to NIS maps, how to limit access to the passwd column of the `passwd` table.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Setting Up Tables

Note – It is much easier to perform this task with the NIS+ installation scripts, as described in Part 1, than with the NIS+ command set, as described here. The methods described in this chapter should be used only by those administrators who are very familiar with NIS+ and who require some nonstandard features or configurations not provided by the installation scripts. Also, if you have them available, the Solaris Management Console tools provide easier methods of working with NIS+ tables.

You can populate NIS+ tables in four ways:

- From files, as described in "Populating NIS+ Tables From Files" on page 181
- From NIS maps, as described in "Populating NIS+ Tables From NIS Maps" on page 186
- With the `nispopulate` script, as described in "Populating NIS+ Tables" on page 96 and "Populating the New Subdomain's Tables" on page 116

- With Solaris Management Console tools, if you have them available

When populating tables from maps or files, the tables should have already been created in the process of setting up a root or subdomain, as explained in Chapter 5 and Chapter 8. Although you can populate a domain's tables at any time after they are created, it is recommended that you do so immediately after setting up the domain. This enables you to add clients more easily, since the required information about the clients should already be available in the domain's tables.

Populating Tables—Options

When you populate a table—whether from a file or an NIS map—you can use any of these options:

- *Replace* - With the replace option, NIS+ first deletes all existing entries in the table and then adds the entries from the source. In a large table, this adds a large set of entries into the master server's `/var/nis/trans.log` file (one set for removing the existing entries, another for adding the new ones), taking up space in `/var/nis`. Thus, propagation to replicas will take longer.
- *Append* - The append option adds the source entries to the NIS+ table. Existing entries are not touched.
- *Merge* - The merge option produces the same results as the replace option but uses a different process. The Merge option updates existing entries rather than deleting and then replacing them. With the merge option, NIS+ handles three types of entries differently:
  - Entries that exist only in the source are added to the table.
  - Entries that exist in both the source and the table are updated in the table.
  - Entries that exist only in the NIS+ table are deleted from the table.

  When updating a large table with a file or map whose contents are not vastly different from those of the table, the merge option can spare the server a great many operations. Because it deletes only the entries that are not duplicated in the source (the replace option deletes *all* entries, indiscriminately), it saves one delete and one add operation for every duplicate entry. Therefore, this is the preferred option.

Populating NIS+ Tables From Files

This task transfers the contents of an ASCII file, such as `/etc/hosts`, into an NIS+ table.

Here is an outline of the procedure:

1. Check the content of each file that you will be transferring data from.
2. Make a copy of each file. Use this copy for the actual transfer. In this guide, copies of files to be transferred are given names ending in `xfr` (for example, `hosts.xfr`).
3. Log in to an NIS+ client. (You must have credentials and permissions allowing you to update the tables. See "Security Considerations When Populating Tables From Files" on page 181, below.)
4. Add `/usr/lib/nis` to the search path for this shell (if not already done).
5. Use `nisaddent` to transfer any of these files one at a time: `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`, `shadow`, and `ipnodes`.

   Note – The new `/etc/inet/ipnodes` file contains IPv4 and IPv6 addresses. Use `nisaddent` to transfer the `/etc/inet/ipnodes` file into the `ipnodes.org_dir` table.

   ---

6. Transfer the `publickey` file.
7. Transfer the automounter information.
8. Ping any replicas.
9. Checkpoint the tables.

Security Considerations When Populating Tables From Files

You can populate NIS+ tables from any NIS+ client or from the root master server. You do not have to be logged in as superuser (root) to populate NIS+ tables, but you do have to have the proper credentials and access rights. If you are going to replace or merge the entries in the table with the entries from the text file, you must have create and destroy rights to the table. If you are going to append the new entries, you only need create rights.

Note – The NIS+ security system is complex. If you are not familiar with NIS+ security, you may want to review Chapter 11 before starting to set up your NIS+ environment.

After you complete this operation, the table entries will be owned by the NIS+ principal that performed the operation and the group specified by the `NIS_GROUP` environment variable.

Prerequisites to Populating Tables From Files

- The domain must have already been set up and its master server must be running.
- The domain's servers must have enough swap space to accommodate the new table information.
- The information in the file must be formatted appropriately for the table into which it will be loaded. See "Prerequisites to Running `nispopulate` to Populate Root Server Tables" on page 96 for information concerning what format a text file must have to be transferred into its corresponding NIS+ table. Local `/etc` files are usually formatted properly, but may have several comments that you need to remove.
- Machine and user names cannot be duplicated. All users and all machines must have unique names. You cannot have a machine with the same name as a user.
- Machine names cannot contain dots (periods) or underscores. For example, a machine named `sales.alpha` is not allowed. Hyphens, however, are allowed. For example, a machine name such as `sales-alpha` is allowed.

Information You Need

You need the name and location of the text files that will be transferred.

Populating NIS+ Tables From Files—Task Map

TABLE 9–1 Populating NIS+ Tables From Files

Task	Description	For Instructions, Go To
Populating NIS+ Tables From Files	Populate NIS+ tables from files	"How to Populate NIS+ Tables From Files" on page 182

▼ How to Populate NIS+ Tables From Files

1. **Check each file that you will be transferring data from.**

Make sure that there are no spurious or incorrect entries. Make sure that the right data is in the correct place and formatted properly. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. (It is easier to add incomplete entries after setup than to try transferring incomplete or damaged entries from the file.)

2. **Make a working copy of each file you will be transferring.**

   Use this working copy for the actual file transfer steps described in this section. Give each working copy the same filename extension (for example, `.xfr`).

   ```
   rootmaster% cp /etc/hosts /etc/hosts.xfr
   ```

   For safety reasons, you might also copy all of the files you plan to use to some directory other than `/etc`. The `nisaddent` and `nispopulate` commands allow you to specify the file source directory.

3. **Log in to an NIS+ client.**

   You can perform this task from any NIS+ client—just be sure that the client belongs to the same domain as the tables into which you want to transfer the information. The examples in this task use the root master server. Because the administrator in these examples is logged on as superuser, the NIS+ principal actually performing this operation (and therefore needing the proper credentials and access rights) is the root master server.

   However, it is *not necessary* to be a superuser (root) or to be logged in on the root master server to update NIS+ tables. Regular users or superusers on any machine can update NIS+ tables, so long as they have the proper credentials, authorization, file permissions.

4. **Add `/usr/lib/nis` to the search path for this shell.**

   Since you will be using the `/usr/lib/nis/nisaddent` command once per table, adding its prefix to the search path will save you the trouble of typing it each time. Here are two examples, one for C shell users and one for Bourne or Korn shell users:

   For C Shell

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

   For Bourne or Korn Shell

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

5. **Use `nisaddent` to transfer any of these files, one at a time:**

   ```
   aliases
   bootparams
   ethers
   group
   hosts
   ipnodes
   netgroup
   netmasks
   ```

```
networks
protocols
rpc, services
```

The `publickey`, automounter, `passwd`, and `shadow` files require slightly different procedures; for the `publickey` file, go to Step 6; for the automounter files, go to Step 7; for the `passwd` and `shadow` files, go to Step 8.

By default, `nisaddent` *appends* the information. To replace or merge instead, use the `-r` or `-m` options.

To replace:

```
rootmaster# nisaddent -r -f filename table[domain]
```

To append:

```
rootmaster# nisaddent -a -f filename table  [domain]
```

To merge:

```
rootmaster# nisaddent -m -f filename table  [domain]
```

The best option for populating the tables for the first time is the `-a` option, the default. The best option to synchronize the NIS+ tables with NIS maps or `/etc` files is the `-m` (merge) option.

- *filename* is the name of the file. The common convention is to append `.xfr` to the end of these file names to identify them as transfer files created with `nisaddent`.
- *table* is the name of the NIS+ table. The *domain* argument is optional; use it only to populate tables in a different domain. Here are some examples, entered from the root domain's master server. The source files are edited versions of the `/etc` files:

  ```
  rootmaster# nisaddent -m -f /etc/hosts.xfr hosts
  rootmaster# nisaddent -m -f /etc/groups.xfr groups
  ```

If you perform this operation from a non-root server, keep in mind that a non-root server belongs to the domain above the one it supports; therefore, it is a client of another domain. For example, the `sales.doc.com.` master server belongs to the `doc.com.` domain. To populate tables in the `sales.doc.com.` domain from that master server, you must append the `sales.doc.com.` domain name to the `nisaddent` statement.

```
salesmaster# nisaddent -f /etc/hosts.xfr hosts Sales.doc.com.
```

If you perform this operation as a client of the `sales.doc.com.` domain, you do not need to append the domain name to the syntax.

To verify that the entries were transferred into the NIS+ table, use the `niscat` command, as described more fully in Chapter 19.

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

Troubleshooting tip: If `niscat` does not now show the updates immediately, it could be because the changes have not been sent by the master server to one or more of the replica servers. In this case, you can either wait and try again in five or ten minutes or use `niscat`'s `-M` option, which specifies that `niscat` obtain its data from the master server.

6. **Transfer the `publickey` file.**

   Because the domain's `cred` table already stores some credentials, you need to make sure they are not overwritten by the contents of the `publickey` text file that you transfer into the cred table. You can avoid this by removing those credentials from the `publickey` text file. For `rootmaster`, there might be one or more lines like the following, all of which should be removed:

   ```
   unix.rootmaster@doc.com public-key:private-key [alg-type]
   ```

   Then you can transfer the contents of the `publickey` file to the cred table. Use `nisaddent`, with the `-a` (add) option.

```
rootmaster# nisaddent -a -f /etc/publickey.xfr -t cred.org_dir publickey [domain]
```

   Note, however, that this operation transfers only DES credentials into the `cred` table. You still need to create their LOCAL credentials to the cred table.

7. **Transfer the automounter information.**

   Although the `nissetup` utility creates `auto_master` and `auto_home` tables, they are not considered standard NIS+ tables. Therefore, transferring information into them requires a slightly different syntax; in particular, you must use the `-t` flag and specify that the table is of type key-value.

```
rootmaster# nisaddent -f auto.master.xfr -t auto_master.org_dir key-value
rootmaster# nisaddent -f auto.home.xfr -t auto_home.org_dir key-value
```

8. **Build the NIS+ `passwd` table.**

   The NIS+ `passwd` table is composed of data drawn from both the `/etc/passwd` and `/etc/shadow` files. Thus, you must run `nisaddent` twice to build the passwd table: once for the `/etc/passwd` file, using `passwd` as the target table, and once for the `/etc/shadow` file, using `shadow` as the target table. (Note that when running `nisaddent` on the `shadow` file, you specify `shadow` as the target table, even though there is no shadow table and the data is actually being placed in the shadow column of the passwd table.)

   ```
   rootmaster# nisaddent -m -f /etc/passwd.xfr passwd
   rootmaster# nisaddent -m -f /etc/shadow.xfr shadow
   ```

9. **Transfer your updates to your replica servers by running `nisping`.**

   Running `nisping` updates any replica servers with your changes.

   ```
   master1# nisping domain
   master1# nisping org_dir.domaincom.
   master1# nisping groups_dir.domain
   ```

10. **Checkpoint the tables.**

Now that you have updated the in-memory copies of the NIS+ data set on your master and replica servers, you should write those changes into the table files on disk. This is called *checkpointing*. (Checkpoint is not *mandatory* at this time, so long as you have regularly scheduled checkpoints that will take care of it later. But if your changes have been significant, it is a good idea to get them written to disk as soon as convenient.)

This step ensures that all the servers supporting the domain transfer the new information from their `.log` files to the disk-based copies of the tables. If you have just set up the root domain, this step affects only the root master server, since the root domain does not yet have replicas. To checkpoint, use the `nisping` command with the `-C` (uppercase) option.

```
rootmaster# nisping -C org_dir
Checkpointing replicas serving directory org_dir.doc.com. :
Master server is rootmaster.doc.com.
 Last update occurred at July 14, 1997
Master server is rootmaster.doc.com.
checkpoint succeeded.
```

If you do not have enough swap space, the server is unable to checkpoint properly, but it will not notify you. One way to make sure that you have sufficient swap space is to list the contents of a table with the `niscat` command. If you do not have enough swap space, you will see this error message:

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

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

Populating NIS+ Tables From NIS Maps

This task transfers the contents of an NIS map into an NIS+ table. Here is a list of the steps:

1. Check the content of each NIS map that you will be transferring data from.
2. Log in to an NIS+ client.
3. Add `/usr/lib/nis` to the search path for this shell.
4. Use `nisaddent` to transfer any of these maps, one at a time: `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`.
5. Transfer the `publickey` map.
6. Transfer the automounter information.
7. Update your replicas with your changes by running `nisping`.
8. Checkpoint the tables.

Security Considerations When Populating Tables From NIS Maps

You can perform this task from any NIS+ client as long as you (or superuser on the client) have the proper credentials and access rights. If you are going to replace or merge the entries in the table with the entries from the NIS map, you must have create and destroy rights to the table. If you are going to append the new entries, you need only create rights.

After you complete this operation, the table entries are owned by the NIS+ principal that performed the operation (either you or, if logged on as superuser, the client) and the group specified by the `NIS_GROUP` environment variable.

Prerequisites to Populating Tables From NIS Maps

- The domain must have already been set up and its master server must be running.
- The `dbm` files (`.pag` and `.dir` files) for the NIS maps you are going to load into the NIS+ tables must already be in a subdirectory of `/var/yp`.
- Machine and user names cannot be duplicated. All users and all machines must have unique names. You cannot have a machine with the same name as a user.
- Machine names cannot contain dots (periods). For example, a machine named `sales.alpha` is not allowed. A machine named `sales-alpha` is allowed.

Information You Need

You need the name and location of the NIS maps.

Populating NIS+ Tables From NIS Maps—Task Map

TABLE 9–2 Populating NIS+ Tables From NIS Maps

Task	Description	For Instructions, Go To
Populating NIS+ Tables From NIS Maps	Populate NIS+ tables from NIS maps	"How to Populate Tables From Maps" on page 187

▼ How to Populate Tables From Maps

1. **Check each NIS map that you will be transferring data from.**

Make sure that there are no spurious or incorrect entries. Make sure that the right data is in the correct place and format properly. Remove any outdated, invalid, or corrupt entries. You should also remove any incomplete or partial entries. (It is easier to add incomplete entries after setup than to try transferring incomplete or damages entries from the map.)

2. **Log in to an NIS+ client.**

   You can perform this task from any NIS+ client—so long as that client belongs to the same domain as the tables into which you want to transfer the information. The examples in this task use the root master server. Since the administrator in these examples is logged in as superuser, the NIS+ principal actually performing this operation (and therefore needing the proper credentials and access rights) is the root master server.

3. **Add `/usr/lib/nis` to the search path for this shell.**

   Because you will be using the `/usr/lib/nis/nisaddent` command once for each table, adding its prefix to the search path will save you the trouble of typing it each time. Here are two examples, one for C shell users and one for Bourne or Korn shell users:

   For C Shell

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

   For Bourne or Korn Shell

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

4. **Use `nisaddent` to transfer any of these maps, one at a time:**

   `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`.

   The `-publickey` and automounter maps require slightly different procedures; for the `publickey` file, go to Step 6, and for the automounter files, go to Step 7.

   By default, `nisaddent` *appends* the file information to the table information. To replace or merge instead, use the `-r` or `-m` options:

   To replace:

   ```
   rootmaster# nisaddent -r -y nisdomain table
   ```

   To append:

   ```
   rootmaster# nisaddent -a -y nisdomain table
   ```

   To merge:

   ```
   rootmaster# nisaddent -m -y nisdomain table
   ```

   The best option for populating the tables for the first time is the `-a` option, which is the default. The best option to synchronize the NIS+ tables with NIS maps or `/etc` files is the `-m` (merge) option.

   The `-y` (lowercase) option indicates an NIS domain instead of a text file. The *nisdomain* argument is the name of the NIS domain whose map you are going transfer into the NIS+ table. You don't have to name the actual map; the `nisaddent` utility automatically selects the NIS map that corresponds to the *table* argument. Here are some examples:

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

The first example transfers the contents of the `hosts.byname` and `hosts.byaddr` maps in the `olddoc` (NIS) domain to the NIS+ hosts table in the root domain (NIS+). The second transfers the NIS maps that store password-related information into the NIS+ passwd table. The third does the same with group-related information. For more information about the `nisaddent` command, see Chapter 19.

5. **Transfer the `publickey` map.**

   Since the domain's cred table already stores some credentials, you need to make sure they are not overwritten by the contents of the `publickey` map that you transfer into the cred table.

   a. **First, dump the `publickey` map to a file, then open that file with your text editor.**

   ```
   rootmaster# makedbm -u /var/yp/olddoc/publickey.byname \
   /etc/publickey.xfr
   rootmaster# vi /tmp/publickey.tmp
   ```

   b. **Now remove the credentials of the machine you are logged in to from the `publickey` map.**

   For `rootmaster`, there might be one or lines like the following, all of which should be removed:

   ```
   unix.rootmaster@doc.com public-key:private-key [alg-type]
   ```

   c. **Now you can transfer the contents of the *file*—not the map—into the cred table. Use `nisaddent`, with the `-a` (add) option.**

```
rootmaster# nisaddent -a -f /etc/publickey.xfr -t cred.org_dir Publickey
```

   Notice that this operation transfers only DES credentials into the `cred` table. You still need to create their LOCAL credentials to the `cred` table.

6. **Transfer the automounter information.**

   Although the `nissetup` utility creates `auto_master` and `auto_home` tables, they are not considered standard NIS+ tables. Therefore, transferring information into them requires a slightly different syntax:

```
rootmaster# nisaddent -y olddoc -Y auto.master -t auto_master.org_dir key-value
rootmaster# nisaddent -y olddoc -Y auto.home -t auto_home.org_dir key-value
```

   The `-m` and `-y` options are still required, as is the NIS domain name (in this instance, `olddoc`). However, you must precede the name of the NIS map (for example, `auto.master`) with a `-Y` (uppercase). Then, as is required when transferring automounter *text files*, you must use the `-t` option, which indicates that this is a nonstandard NIS+ table. Its arguments are the name of the NIS+ directory object (`auto_master.org_dir`) and the type of table (key-value). Be sure to append the `org_dir` suffixes to the NIS+ table names.

7. **Transfer your updates to your replica servers by running `nisping`.**

   Running `nisping` updates any replica servers with your changes.

   ```
   master1# nisping domain
   master1# nisping org_dir.domaincom.
   master1# nisping groups_dir.domain
   ```

8. **Checkpoint the tables.**

   This step ensures that all the servers supporting the domain transfer the new information from their `.log` files to the disk-based copies of the tables. If you just finished setting up the root domain, this step affects only the root master server, since the root domain does not yet have replicas. Use the `nisping` command with the `-C` (uppercase) option.

   ```
   rootmaster# nisping -C org_dir
   Checkpointing replicas serving directory org_dir.doc.com. :
   Master server is rootmaster.doc.com.
    Last update occurred at July 14, 1994
   Master server is rootmaster.doc.com.
   checkpoint succeeded.
   ```

   If you do not have enough swap space, the server is unable to checkpoint properly, but it does not notify you. One way to make sure you have sufficient swap space is to use list the contents of a table with the `niscat` command. If you do not have enough swap space, you will see this error message:

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

   Even though it does not *seem* to, this message indicates that you do not have enough swap space. Increase the swap space and checkpoint the domain again.

Transferring Information From NIS+ to NIS

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

1. Log in to the NIS+ server.
2. Transfer the NIS+ tables in to output files.
3. Transfer the contents of the output files to the NIS maps.

Security Considerations for NIS to NIS+ Transfers

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

Prerequisites to NIS to NIS+ Transfers

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

Transferring Information From NIS+ to NIS — Task Map

TABLE 9–3 Transferring Information From NIS+ to NIS

Task	Description	For Instructions, Go To
Transferring Information From NIS+ to NIS	Transfer information from NIS+ tables to NIS maps on a Solaris 1.x NIS master server	"How to Transfer Information From NIS+ to NIS" on page 191

▼ How to Transfer Information From NIS+ to NIS

1. **Log in to the NIS+ server.**

   This example uses the server named `dualserver`.

2. **Transfer the NIS+ tables to output files.**

   Use the `nisaddent` command with the `-d` option, once for each table.

   ```
   dualserver% /usr/lib/nis/nisaddent -d -t table tabletype > filename
   ```

   The `-d` option transfers the contents of *table* to *filename*, converting the contents back to standard `/etc` file format.

3. **Transfer the contents of the output files in to the NIS maps.**

   The NIS+ output files are ASCII files that you can use as input files for the NIS maps. Copy them into the NIS master's `/etc` directory, then use `make` as usual.

   ```
   dualserver# cd /var/yp
   dualserver# make
   ```

Limiting Access to the `Passwd` Column to Owners and Administrators

This task describes how to limit read access to the password-related columns of the `passwd` table to the entry owner and the table administrators, without affecting the read access of other authenticated principals (including applications) to the remaining columns of the passwd table.

This task establishes the following rights:

```
                         Nobody  Owner   Group  World
Table Level Rights:      ----    rmcd    rmcd   ----
Passwd Column Rights:    ----    rm--    rmcd   ----
Shadow Column Rights:    ----    rm--    rmcd   ----
```

Security Considerations When Limiting `Passwd` Column Access

- The domain must *not* be running in NIS-compatibility mode.
- All clients of the domain must have DES credentials.
- All clients of the domain must be running Solaris Release 2.3 or a later release.
- Users' network passwords (used to encrypt their DES credentials) must be the same directory as their login passwords.

Prerequisites to Limiting `Passwd` Column Access

- The `passwd` table must have already been set up. It need not have any information in it, however.
- The NIS+ principal performing this task must have modify rights to the `passwd` table.

Information You Need

All you need is the name of the `passwd` table.

Limiting Access to the Passwd Column to Owners and Administrators — Task Map

TABLE 9–4 Limiting Access to the Passwd Column to Owners and Administrators

Task	Description	For Instructions, Go To
Limiting Access to the Passwd Column to Owners and Administrators	Modify `passwd.org_dir`, via NIS+ commands, to restrict access to the passwd column for owners and administrators.	"How to Limit Read Access to the Passwd Column" on page 193

▼ How to Limit Read Access to the Passwd Column

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

   The examples in this task use the root master server, `rootmaster`.

2. **Check the current table and column permissions.**

   Use the `niscat -o` command.

```
rootmaster# niscat -o passwd.org_dir
```

   This task assumes the existing permissions are:

```
Access Rights    : ----rmcdrmcdr---
Columns          :
                    [0]  Name              : name
                         Access Rights : r-----------r---
                    [1]  Name              : passwd
                         Access Rights : -----m----------
                    [2]  Name              : uid
                         Access Rights : r-----------r---
                    [3]  Name              : gid
                         Access Rights : r-----------r---
                    [4]  Name              : gcos
                         Access Rights : r----m------r---
                    [5]  Name              : home
                         Access Rights : r-----------r---
                    [6]  Name              : shell
                         Access Rights : r-----------r---
                    [7]  Name              : shadow
                         Access Rights : r-----------r---
```

   If your permissions are different, you may need to use a different syntax. For instructions, see Chapter 15.

3. **Change the table permissions.**

   Use the `nischmod` command to change the table's object-level permissions to `---- rmcdrmcd ----`

```
rootmaster# nischmod og=rmcd,nw= passwd.org_dir
```

4. **Change the column permissions.**

   Use the `nistbladm` command with the `-u` option to change the permissions of the passwd and shadow columns to:

   ```
   passwd ---- rm-- ---- ----
   shadow ---- r--- ---- ----
   rootmaster# nistbladm -u passwd=o+r, shadow=o+r passwd.org_dir
   ```

5. **Verify the new permissions.**

   Use the `niscat -o` command, as you did in Step 2. The permissions should look the same as they do in that step's output.

Table Population Summaries

Following are summaries of the steps required to populate NIS+ tables. They assume the simplest case, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. For brevity, these summaries do not show the server's responses to each command.

TABLE 9–5 Transferring Files Into NIS+ Tables: Command Summary

Tasks	Commands
Log in to an NIS+ client.	`rootmaster%`
Create working copies of the files to be transferred.	`% cp /etc/hosts /etc/hosts.xfr`
Add `/usr/lib/nis` to search path.	`% PATH=$PATH:/usr/lib/nis; export PATH`
Transfer each file, one at a time.	`% nisaddent -m -f /etc/hosts.xfr hosts`
Remove old server credentials from `publickey` file.	`% vi /etc/publickey.xfer`
Transfer it to the cred table.	`% nisaddent -a -f /etc/publickey.xfr cred`
Transfer the automounter files.	`% nisaddent -f auto.master.xfr -t auto_master.org_dir key-value` `% nisaddent -f auto.home.xfr -t auto_home.org_dir key-value`
Checkpoint the table directory.	`% nisping -C org_dir`

TABLE 9–6 Transferring Maps Into NIS+ Tables: Command Summary

Tasks	Commands
Log in to an NIS+ client.	`rootmaster%`
Add `/usr/lib/nis` to search path.	`% PATH=$PATH:/usr/lib/nis; export PATH`
Transfer each map, one at a time.	`% nisaddent -m -y olddoc hosts`
Dump `publickey` map to a file.	`% makedbm -u /var/yp/olddoc/publickey.byname > /etc/publickey.xfr`
Remove new credentials.	`% vi /etc/publickey.xfr`
Transfer the `publickey` file.	`% nisaddent -a -f /etc/publickey.xfr -t cred.ortg_dir publickey`

Transfer the automounter maps.	`% nisaddent -y olddoc -Y auto.master -t auto_master.org_dir key-value`
	`% nisaddent -y olddoc -Y auto.home -t auto_home.org_dir key-value`

Checkpoint the table directory.	`% nisping -C org_dir`

TABLE 9–7 Transferring NIS+ Tables to NIS Maps: Command Summary

Tasks	Commands
Log in to NIS+ server.	`dualserver%`
Transfer NIS+ tables to files.	`% /usr/lib/nis/nisaddent -d [-t` *table*`]` *tabletype* `>` *filename*
Transfer files to NIS maps.	`% makedbm` *flags output-file NIS-dbm-file*

TABLE 9–8 Limiting Access to Passwd Column: Command Summary

Tasks	Commands
Log into the domain's master server.	`rootmaster#`
Check the table's existing rights.	`# niscat -o passwd.org_dir`
Assign the table new rights.	`# nischmod og=rmcd,nw= passwd.org_dir`
Assign the columns new rights	`# nistbladm -u passwd=o+r, shadow=n+r passwd.org_dir`
Verify the new rights.	`# niscat -o passwd.org_dir`

PART III NIS+ Administration

This part describes the administration and troubleshooting of the NIS+ naming service in the Solaris OS.

CHAPTER **10**

NIS+ Tables and Information

This chapter describes the structure of NIS+ tables and provides a brief overview of how they can be set up.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Table Structure

NIS+ stores a wide variety of network information in tables. 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 10–1 lists the preconfigured NIS+ tables.

Note – As a naming service, NIS+ tables are designed to store references to objects, not the objects themselves. For this reason, NIS+ does not support tables with large entries. If a table contains excessively large entries, `rpc.nisd` may fail.

TABLE 10–1 NIS+ Tables

Table	Information in the Table
hosts	Network address and host name of every machine in the domain
bootparams	Location of the root, swap, and dump partition of every diskless client in the domain
passwd	Password information about every user in the domain
cred	Credentials for principals who belong to the domain
group	The group name, group password, group ID, and members of every UNIX group in the domain
netgroup	The netgroups to which machines and users in the domain may belong
mail_aliases	Information about the mail aliases of users in the domain
timezone	The time zone of every machine 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 machine 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

Because it contains only information related to NIS+ security, the Cred table, is described in Chapter 12.

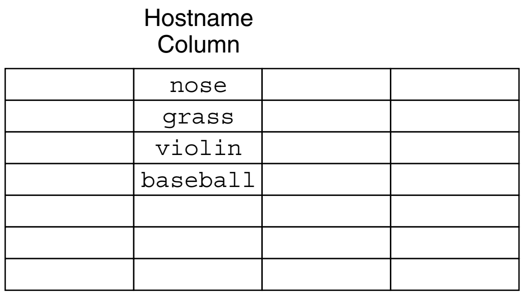
These tables store a wide variety of information, ranging from user names to Internet services. Most of this information is generated during a setup or configuration procedure. For instance, an entry in the passwd table is created when a user account is set up. An entry in the hosts table is created when a machine 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, Chapter 23 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 system and network administration manuals.

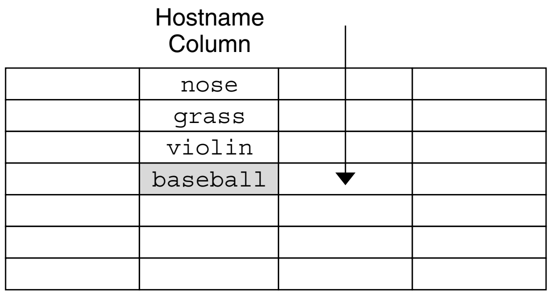
Note – You can create more automounter maps for a domain, but be sure to store them as NIS+ tables and list them in the auto_master table. When creating additional automount maps to supplement `auto_master` (which is created for you), the column names must include `key` and `value`. For more information about the automounter, consult books about the automounter or books that describe the NFS file system.

Columns and Entries

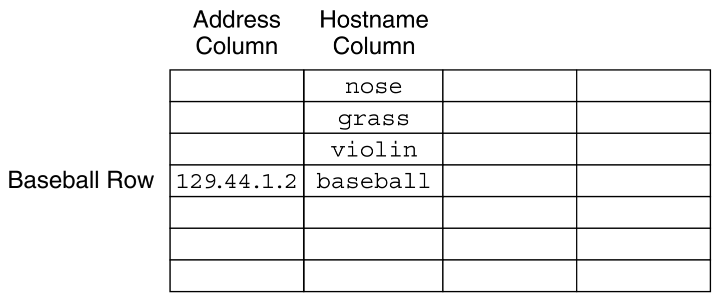
Although NIS+ tables store different types of information, they all have the same underlying structure; they are each made up of rows and columns (the rows are called "entries" or "entry objects"):



A client can access information by a key, or by any column that is searchable. For example, to find the network address of a machine 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 Chapter 15.

Search Paths

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

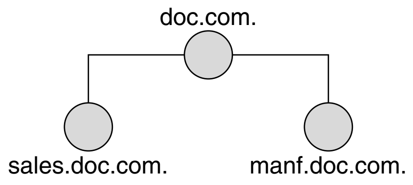
If a client in one domain tries to find information that is stored in another domain, it has to provide a fully qualified name. As described in "`NIS_PATH` Environment Variable" on page 74 if the `NIS_PATH` environment variable is set up properly, the NIS+ service will do this automatically.

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 at 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 tables of the lower two domains contain the following information.

TABLE 10–2 Example Hosts Table

sales.doc.com.		**manf.doc.com.**	
`10.0.0.1`	`localhost`	`172.18.0.1`	`localhost`
`10.22.3.22`	`luna`	`172.29.6.1`	`sirius`
`10.22.3.24`	`phoebus`	`172.29.6.112`	`rigel`
`10.22.3.25`	`europa`	`172.29.6.90`	`antares`
`10.22.3.27`	`ganymede`	`172.29.6.101`	`polaris`
`10.22.3.28`	`mailhost`	`172.29.6.79`	`mailhost`

Assume now that a user logged onto the `luna` machine in the `sales.doc.com.` domain wants to log in remotely to another client. Without providing a fully qualified name, that user can only remotely log on to five machines: `localhost`, `phoebus`, `europa`, `ganymede`, and the `mailhost`.

Now assume that the search path of the `hosts` table in the `sales.doc.com.` domain listed the `hosts` table from the `manf.doc.com.` domain:

```
hosts.org_dir.manf.doc.com.
```

Now a user in the `sales.doc.com.` domain can enter something like `rlogin sirius`, and the NIS+ server will find it. It will first look for `sirius` in the local domain, but when it does not find a match, it will look in the `manf.doc.com.` domain. How does the client know how to find the `manf.doc.com.` domain? As described in Chapter 2, 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 2.

There is a slight drawback, though, to specifying a search path. If the user were to enter an incorrect name, such as `rlogin luba` (rather than "luna"), 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.doc.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 "Using the `nistbladm` Command With Tables" on page 346.

Ways to Set Up Tables

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 "NIS+ Files and Directories" on page 53, 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. The `nisserver` script creates the appropriate directories and a full set of system tables. Running the `nisserver` script is the recommended method.
- Use the `nismkdir` command. The `nismkdir` command simply creates the directory.
- Use the `/usr/lib/nis/nissetup` utility. The `nissetup` utility creates the `org_dir` and `groups_dir` directories and a full set of system tables.

The `nisserver` script and the `nissetup` and `nismkdir` utilities are described in "The `nismkdir` Command" on page 326. Additional information on the `nismkdir` command can be found in "The `nismkdir` Command" on page 326.

A 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 class 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 Chapter 10. 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 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. (Note that if you are creating additional automount maps, the first column must be named `key` and the second column named `value`.)

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.org_dir.log
bootparams.org_dir.log
password.org_dir.log
ipnodes.org_dir.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.

CHAPTER **11**

NIS+ Security Overview

This chapter describes the NIS+ security system and how it affects the entire NIS+ namespace.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Solaris Security—Overview

In essence, Solaris security is provided by gates that users must pass through in order to enter the Solaris system, and permission matrixes that determine what they are able to do once inside. (See Figure 11–1 for a schematic representation of this system.)

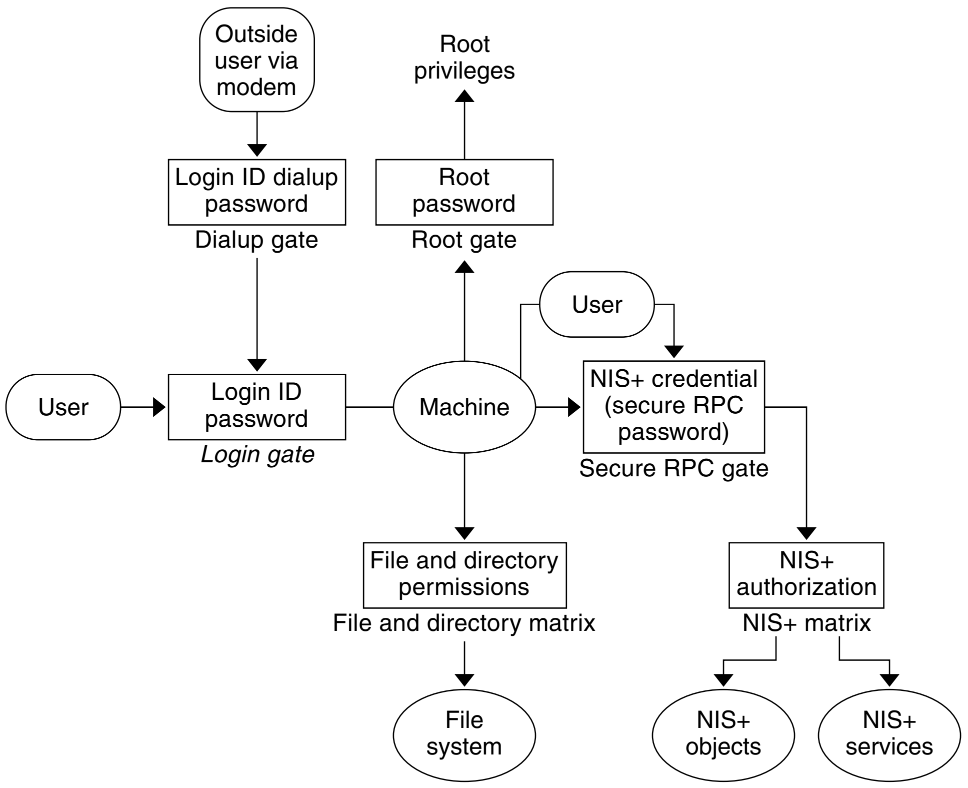


FIGURE 11–1 Solaris Security Gates and Filters

As you can see in the above figure, the overall system is composed of four gates and two permission matrixes:

- *Dialup gate*. To access a given Solaris system from the outside through a modem and phone line, you must provide a valid Login ID and Dialup password.
- *Login gate*. To enter a given Solaris system you must provide a valid login ID and user password.
- *File and Directory Matrix*. Once you have gained access to a Solaris system, your ability to read, execute, modify, create, and destroy files and directories is governed by the applicable permissions matrix.
- *Root gate*. To gain access to root privileges, you must provide a valid super user (root) password.
- *Secure RPC gate*. In an NIS+ environment running at security level 2 (the default), when you try to use NIS+ services and gain access to NIS+ objects (servers, directories, tables, table entries, and so forth) your identity is confirmed by NIS+ using the Secure RPC process.

  Consult your Solaris documentation for detailed descriptions of the Dialup, Login, and Root gates, and the File and Directory permissions matrix.

To enter the Secure RPC gate requires presentation of a Secure RPC password. (In some contexts *Secure RPC passwords* have been referred to as *network passwords*.) Your Secure RPC password and your login password normally are identical and when that is the case you are passed through the gate automatically without having to re-enter your password. (See "Secure RPC Password Versus Login Password Problem" on page 229 for information regarding cases where the two passwords are not the same.)

A set of *credentials* are used to automatically pass your requests through the Secure RPC gate. The process of generating, presenting, and validating your credentials is called *authentication* because it confirms who you are and that you have a valid Secure RPC password. This authentication process is automatically performed every time you request an NIS+ service.

In an NIS+ environment running in NIS-compatibility mode (also known as YP-compatibility mode), the protection provided by the Secure RPC gate is significantly weakened because everyone has read rights for all NIS+ objects and modify rights for those entries that apply to them regardless of whether or not they have a valid credential (that is, regardless of whether or not the authentication process has confirmed their identity and validated their Secure RPC password). Since that allows *anyone* to have read rights for all NIS+ objects and modify rights for those entries that apply to them, an NIS+ network running in compatibility mode is less secure than one running in normal mode.

For details on how to create and administer NIS+ authentication and credentials, see Chapter 12.

- *NIS+ objects matrix*. Once you have been properly authenticated to NIS+ your ability to read, modify, create, and destroy NIS+ objects is governed by the applicable permissions matrix. This process is called NIS+ *authorization*.

  For details NIS+ permissions and authorization, see Chapter 15.

NIS+ Security—Overview

NIS+ security is an integral part of the NIS+ namespace. You cannot set up security and the namespace independently. For this reason, instructions for setting up security are woven through the steps used to set up the other components of the namespace. Once an NIS+ security environment has been set up, you can add and remove users, change permissions, reassign group members, and all other routine administrative tasks needed to manage an evolving network.

The security features of NIS+ protect the information in the namespace, as well as the structure of the namespace itself, from unauthorized access. Without these security features, any NIS+ client could obtain and change information stored in the namespace or even damage it.

NIS+ security does two things:

- *Authentication*. Authentication is used to identify NIS+ principals. Every time a principal (user or machine) tries to access an NIS+ object, the user's identity and Secure RPC password is confirmed and validated.
- *Authorization*. Authorization is used to specify access rights. Every time NIS+ principals try to access NIS+ objects, they are placed in one of four authorization classes (owner, group, world, nobody). The NIS+ security system allows NIS+ administrators to specify different read, modify, create, or destroy rights to NIS+ objects for each class. Thus, for example, a given class could be permitted to modify a particular column in the passwd table but not read that column, or a different class could be allowed to read some entries of a table but not others.

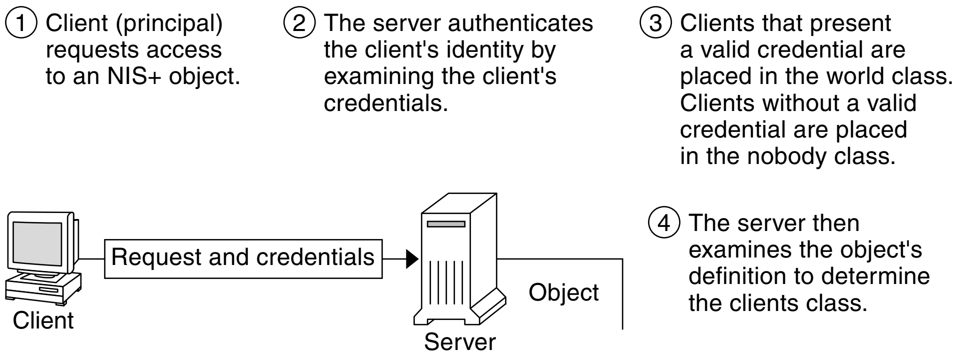
In essence, then, NIS+ security is a two-step process:

1. *Authentication*. NIS+ uses credentials to confirm that you are who you claim to be.
2. *Authorization*. Once your identity is established by the authentication process, NIS+ determines your class. What you can do with a given NIS+ object or service depends on which class you belong to. This is similar in concept to the standard UNIX file and directory permissions system. (See "Authorization Classes" on page 215 for more information on classes.)

This process, for example, prevents someone with root privileges on machine A from using the `su` command to assume the identity of a second user and then accessing NIS+ objects with the second user's NIS+ access privileges.

Note, however, that NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is logged in from the *same* machine.

Figure 11–2 details this process.



1 Client (principal) requests access to an NIS+ object.
2 The server authenticates the client's identity by examining the client's credentials.
3 Clients that present a valid credential are placed in the world class. Clients without a valid credential are placed in the nobody class.
4 The server then examines the object's definition to determine the clients class.
Request and credentials
Client
Server
Object

FIGURE 11–2 Summary of the NIS+ Security Process

NIS+ Principals

NIS+ principals are the entities (clients) that submit requests for NIS+ services. An NIS+ principal may be someone who is logged in to a client machine as a regular user, someone who is logged in as superuser, or any process that runs with superuser permission on an NIS+ client machine. Thus, an NIS+ principal can be a client user or a client machine.

An NIS+ principal can also be the entity that supplies an NIS+ service from an NIS+ server. Since all NIS+ servers are also NIS+ clients, much of this discussion also applies to servers.

NIS+ Security Levels

NIS+ servers operate at one of two security levels. These levels determine the type of credential principals that must submit for their requests to be authenticated. NIS+ is designed to run at the most secure level, which is security level 2. Level 0 is provided only for testing, setup, and debugging purposes. These security levels are summarized in Table 11–1.

TABLE 11–1 NIS+ Security Levels

Security Level	Description
0	Security level 0 is designed for testing and setting up the initial NIS+ namespace. An NIS+ server running at security level 0 grants any NIS+ principal full access rights to all NIS+ objects in the domain. Level 0 is for setup purposes only and should only be used by administrators for that purpose. Level 0 should not be used on networks in normal operation by regular users.
1	Security level 1 uses AUTH_SYS security. This level is not supported by NIS+ and should not be used.
2	Security level 2 is the default. It is the highest level of security currently provided by NIS+. It authenticates only requests that use DES credentials. Requests with no credentials are assigned to the nobody class and have whatever access rights that have been granted to that class. Requests that use invalid DES credentials are retried. After repeated failure to obtain a valid DES credential, requests with invalid credentials fail with an authentication error. (A credential might be invalid for a variety of reasons such as the principal making the request is not keylogged in on that machine, the clocks are out of synch, there is a key mismatch, and so forth.)

Security Levels and Password Commands

In Solaris releases 2.0 through 2.4, you used the `nispasswd` command to change your password. However, `nispasswd` could not function without credentials. (In other words, it could not function under security level 0 unless there were credentials existing from some previous higher level.) Starting with Solaris Release 2.5, the `passwd` command should now be used to change your own password regardless of security level or credential status.

NIS+ Authentication and Credentials—Introduction

The purpose of NIS+ credentials is to *authenticate* (confirm) the identity of each principal requesting an NIS+ service or access to an NIS+ object. In essence, the NIS+ credential/authorization process is an implementation of the Secure RPC system.

The credential/authentication system prevents someone from assuming some other user's identity. That is, it prevents someone with root privileges on one machine from using the `su` command to assume the identity of a second user who is not logged in and then accessing NIS+ objects with the second user's NIS+ access privileges.

Once a server authenticates a principal, the principal is placed in one of four authorization classes. The server then checks the NIS+ object that the principal wants to access to see what activities that class of principal is authorized to perform. (See "NIS+ Authorization and Access—Introduction" on page 214 for further information on authorization.)

User and Machine Credentials

There are two basic types of principal, *users* and *machines,* and thus two different types of credentials:

- *User credentials.* When someone is logged in to an NIS+ client as a regular user, requests for NIS+ services include that person's *user* credentials.
- *Machine credentials.* When a user is logged in to an NIS+ client as superuser, request for services use the *client machine's* credentials.

DES versus LOCAL Credentials

NIS+ principals can have two types of credential: DES and LOCAL.

DES Credentials

Note – DES credentials are only one method of achieving authentication. In the future, other methods may be available. Thus, do not equate DES credentials with NIS+ credentials.

In this document, the term DES credentials is used generically to denote a Diffie-Hellman key based authentication, regardless of key length. The system allows you to specify the key length from a pre-determined set. Use `nisauthconf`(1M) to set or display the Diffie-Hellman key length.

DES (Data Encryption Standard) credentials are the type of credential that provide secure authentication. When this guide refers to NIS+ checking a credential to authenticate an NIS+ principal, it is the DES credential that NIS+ is validating.

Each time a principal requests an NIS+ service or access to an NIS+ object, the software uses the credential information stored for that principal to generate a credential for that principal. DES credentials are generated from information created for each principal by an NIS+ administrator, as explained in Chapter 12.

- When the validity of a principal's DES credential is confirmed by NIS+, that principal is *authenticated*.
- A principal must be authenticated in order to be placed in the owner, group, or world authorization classes. In other words, you must have a valid DES credential in order to be placed in one of those classes. (Principals who do not have a valid DES credential are automatically placed in the nobody class.)
- DES credential information is always stored in the cred table of the principal's home domain, regardless of whether that principal is a client user or a client machine.

LOCAL Credentials

LOCAL credentials are simply a map between a user's User ID number and NIS+ principal name which includes their home domain name. When users log in, the system looks up their LOCAL credential, which identifies their home domain where their DES credential is stored. The system uses that information to get the user's DES credential information.

When users log in to a remote domain, those requests use their LOCAL credential which points back to their home domain; NIS+ then queries the user's home domain for that user's DES credential information. This allows a user to be authenticated in a remote domain even though the user's DES credential information is not stored in that domain.

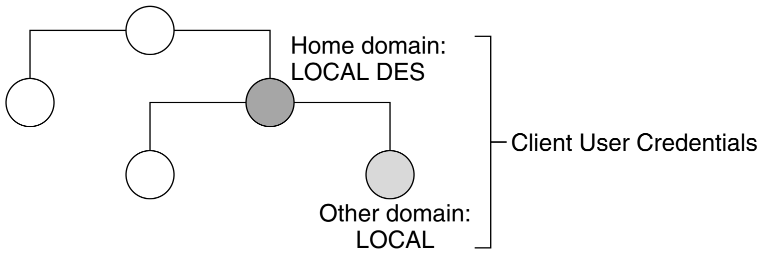


FIGURE 11–3 Credentials and Domains

LOCAL credential information can be stored in any domain. In fact, in order to log into a remote domain and be authenticated, a client user *must* have a LOCAL credential in the cred table of the remote domain. If a user does not have a LOCAL credential in a remote domain the user is trying to access, NIS+ will be unable to locate the user's home domain to obtain the user's DES credential. In such a case the user would not be authenticated and would be placed in the nobody class.

User Types and Credential Types

A user can have both types of credentials, but a machine can only have DES credentials.

Root cannot have NIS+ access, as root, to other machines because the root UID of every machine is always zero. If root (UID=0) of machine A tried to access machine B as root, that would conflict with machine B's already existing root (UID=0). Thus, a LOCAL credential doesn't make sense for a client *machine*; so it is allowed only for a client *user*.

TABLE 11–2 Types of Credentials

Type of Credential	Client User	Client machine
DES	Yes	Yes
LOCAL	Yes	No

NIS+ Authorization and Access—Introduction

The basic purpose of NIS+ authorization is to specify the access rights that each NIS+ principal has for each NIS+ object and service.

Once the principal making an NIS+ request is authenticated, NIS+ places them in an authorization class. The access rights (permissions) that specify which activities a principal may do with a given NIS+ object are assigned on a class basis. In other words, one authorization class may have certain access rights while a different class has different rights.

- *Authorization classes.* There are four authorization classes: owner, group, world, and nobody. (See "Authorization Classes" on page 215 below for details.)
- *Access rights.* There are four types of access rights (permissions): create, destroy, modify, and read. (See "Introduction to NIS+ Access Rights" on page 218 for details.)

Authorization Classes

NIS+ objects do not grant access rights directly to NIS+ principals. Instead, they grant access rights to four *classes of principal*:

- *Owner.* The principal who happens to be the object's owner gets the rights granted to the owner class.
- *Group.* Each NIS+ object has one group associated with it. The members of an object's group are specified by the NIS+ administrator. The principals who belong to the object's group class get the rights granted to the group class. (In this context, *group* refers to NIS+ groups, not UNIX or net groups.)
- *World.* The world class encompasses all NIS+ principals that a server has been able to authenticate. (That is, everyone who has been authenticated but who is not in either the owner or group classes.)
- *Nobody.* Everyone belongs to the nobody class even those who are not authenticated.

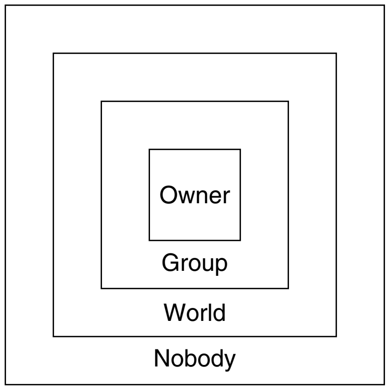


FIGURE 11–4 Authorization Classes

For any NIS+ request, the system determines which class the requesting principal belongs to and the principal then can use whatever access rights belonging to that class.

An object can grant any combination of access rights to each of these classes. Normally, however, a higher class is assigned the same rights as all the lower classes, plus possible additional rights.

For instance, an object could grant read access to the nobody and world classes; both read and modify access to the group class; and read, modify, create, and destroy access to the owner class.

The four classes are described in detail below.

The Owner Class

The owner is a *single* NIS+ principal.

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) before being granted owner access rights.

By default, an object's owner is the principal that created the object. However, an object's owner can cede ownership to another principal in two ways:

- One way is for the principal to specify a different owner at the time the object is created (see "Specifying Access Rights in Commands" on page 270).
- A second way is for the principal to change the ownership of the object after it is created (see "Changing Ownership of Objects and Entries" on page 280).

Once a principal gives up ownership, that principal gives up all owner's access rights to the object and keeps only the rights the object assigns to either the group, the world, or nobody.

The Group Class

The object's group is a *single* NIS+ group. (In this context, *group* refers to NIS+ groups, not UNIX or net groups.)

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) and belong to the group before being granted group access rights.

An NIS+ group is a collection of NIS+ principals, grouped together as a convenience for providing access to the namespace. The access rights granted to an NIS+ group apply to all the principals that are members of that group. (An object's owner, however, does not need to belong to the object's group.)

When an object is created it may be assigned a default group. A nondefault group can be specified for an object when it is created or later. An object's group may be changed at any time.

Note – Information about NIS+ groups is not stored in the NIS+ `group` table. The `group` table stores information about UNIX groups. Information about NIS+ groups is stored in the appropriate `groups_dir` directory object.

Information about NIS+ groups is stored in NIS+ group *objects*, under the `groups_dir` subdirectory of every NIS+ domain.

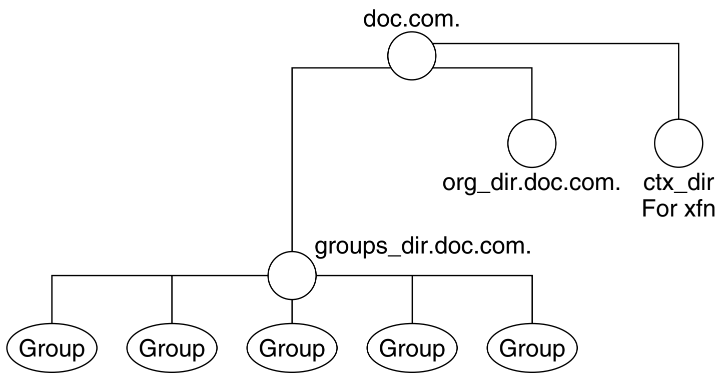


FIGURE 11–5 NIS+ Directory Structure

Instructions for administering NIS+ groups are provided in Chapter 17.

The World Class

The world class contains all NIS+ principals that are authenticated by NIS+. In other words, the world class includes everyone in the owner and group class, plus everyone else who presents a valid DES credential.

Access rights granted to the world class apply to all authenticated principals.

The Nobody Class

The nobody class is composed of anyone who is not properly authenticated. In other words, the nobody class includes everyone who does not present a valid DES credential.

Authorization Classes and the NIS+ Object Hierarchy

There is a hierarchy of NIS+ objects and authorization classes that can apply independently to each level. The standard default NIS+ directory hierarchy is:

- *Directory level*. In each NIS+ domain there are two NIS+ directory objects: `groups_dir` and `org_dir`. Each `groups_dir` directory object contains various groups. Each `org_dir` directory object contains various tables.
- *Group level or table level*. Groups contain individual entries and possibly other groups. Tables contain both columns and individual entries.
- *Column level*. A given table will have one or more columns.
- *Entry (row) level*. A given group or table will have one or more entries.

The four authorization classes apply at each level. Thus, a directory object will have its own owner and group. The individual tables within a directory object will have their own individual owners and groups which may be different than the owner and group of the directory object. Within a table, an entry (row) may have its own individual owner or group which may be different than the owner and group of the table as a whole or the directory object as a whole. Within a table, individual columns have the same owner and group as the table as a whole.

Introduction to NIS+ Access Rights

NIS+ objects specify their access rights as part of their object definitions. (You can examine these by using the `niscat -o` command.)

NIS+ objects specify access rights for NIS+ principals in the same way that UNIX files specify permissions for UNIX users. Access rights specify the types of operations that NIS+ principals are allowed to perform on NIS+ objects.

NIS+ operations vary among different types of objects, but they all fall into one of the four access rights categories: read, modify, create, and destroy.

- *Read* A principal with read rights to an object can view the contents of that object.
- *Modify.* A principal with modify rights to an object can change the contents of that object.
- *Destroy.* A principal with destroy rights to an object can destroy or delete the object.
- *Create.* A principal with create rights to a higher level object can create new objects within that level. In other words, if you have create rights to an NIS+ directory object, you can create new tables within that directory. If you have create rights to an NIS+ table, you can create new columns and entries within that table.

Every communication from an NIS+ client to an NIS+ server is, in effect, a request to perform one of these operations on a specific NIS+ object. For instance, when an NIS+ principal requests the IP address of another machine, it is effectively requesting read access to the *hosts* table object, which stores that type of information. When a principal asks the server to add a directory to the NIS+ namespace, it is actually requesting *modify* access to the directory's parent object.

Keep in mind that these rights logically evolve down from directory to table to table column and entry levels. For example, to create a new table, you must have create rights for the NIS+ directory object where the table will be stored. When you create that table, you become its default owner. As owner, you can assign yourself create rights to the table which allows you to create new entries in the table. If you create new entries in a table, you become the default owner of those entries. As table owner, you can also grant table-level create rights to others. For example, you can give your table's group class table-level create rights. In that case, any member of the table's group can create new entries in the table. The individual member of the group who creates a new table entry becomes the default owner of that entry.

The NIS+ Administrator

An NIS+ administrator is anyone who has *administrative rights* over an NIS+ object. For the purpose of this discussion, administrative rights are defined as create, destroy, and for some objects, modify rights. (See "Introduction to NIS+ Access Rights" on page 218 for a description of NIS+ access rights.)

Whoever creates an NIS+ object sets the initial access rights to that object. If the creator restricts administrative rights to the object's owner (initially the creator), than only the owner has administrative power over that object. On the other hand, if the creator grants administrative rights to the object's group, then everyone in that group has administrative power over that object.

Thus, who ever has administrative rights over an object is considered to be an NIS+ administrator for that object.

In other words, the NIS+ software does not enforce any requirement that there be a single NIS+ administrator.

Theoretically, you could grant administrative rights to the world class, or even the nobody class. The software allows you to do that. But granting administrative rights beyond the group class effectively nullifies NIS+ security. Thus, if you grant administrative rights to either the World or the nobody class you are, in effect, defeating the purpose of NIS+ security.

NIS+ Password, Credential, and Key Commands

Note – The NIS+ service is managed by the Service Management Facility. Administrative actions on this service, such as enabling, disabling, or restarting, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using the Facility with NIS+. For an overview of the Facility, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

Use the following commands to administer passwords, credentials, and keys (see the appropriate man pages for a full description of each command):

- `chkey`. Changes a principal's Secure RPC key pair. Do not use `chkey` unless necessary, use `passwd` instead. See "Changing Keys for an NIS+ Principal" on page 245 for more information.
- `keylogin`. Decrypts and stores a principal's secret key with the `keyserv`.
- `keylogout`. Deletes stored secret key from `keyserv`.
- `keyserv`. Enables the server for storing private encryption keys. See "Keylogin With NIS+" on page 244 for more information about keys. See "NIS+ and the Service Management Facility" on page 85 for information about managing NIS+ by using the Service Management Facility.
- `newkey`. Creates a new key pair in public-key database.
- `nisaddcred`. Creates credentials for NIS+ principals. See "Creating Credential Information" on page 233 and "Administering NIS+ Credential Information" on page 241 for more information.
- `nisauthconf`. Display or set the Diffie-Hellman key length.
- `nisupdkeys`. Updates public keys in directory objects. See "Updating Public Keys" on page 249 for more information.
- `passwd`. Changes and administers principal's password. See Chapter 16 for more information.

CHAPTER 12

Administering NIS+ Credentials

This chapter describes NIS+ credentials and how to administer them.

Tip – Some NIS+ security tasks can be performed more easily with Solaris Management Console tools if you have them available.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Credentials

NIS+ credentials are used to identify NIS+ users. This chapter assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that credentials play in that system.

For a complete description of NIS+ credential-related commands and their syntax and options, see the NIS+ man pages.

Note – The description of DES credentials in this chapter is applicable to 192–bit Diffie-Hellman DES credentials. While similar, authentication using other key lengths differs in details. When the command line interface is used to manipulate the keys, the differences are transparent to both the user and the system administrator. Use `nisauthconf`(1M) to display or set the prescribed key lengths.

How Credentials Work

Note – Some NIS+ security tasks can be performed more easily with Solaris Management Console tools, if you have them available.

The credential/authentication system prevents someone from assuming some other user's identity. That is, it prevents someone with root privileges on one machine from using the `su` command to assume the identity of a second user who is either not logged in at all or logged in on another machine and then accessing NIS+ objects with the second user's NIS+ access privileges.

Caution – NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and the other user's NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is currently logged in on the *same* machine.

Credential Versus Credential Information

To understand how DES credentials are created and how they work, you need to distinguish between the credential itself and the information that is used to create and verify it.

- *Credential information:* The data that is used to generate a DES credential and by the server to verify that credential.
- *DES credential*: The bundle of numbers that is sent by the principal to the server to authenticate the principal. A principal's credential is generated and verified each time the principal makes an NIS+ request. See "The DES Credential in Detail" on page 226 for a detailed description of the DES credential.

Authentication Components

In order for the credential/authentication process to work the following components must be in place:

- *Principal's DES credential information*. This information is initially created by an NIS+ administrator for each principal. It is stored in the cred table of the principal's home domain. A principal's DES credential information consists of:
  - *Principal name.* This would be a user's fully qualified login ID or a machine's fully qualified host name.
  - *Principal's Secure RPC netname.* Each principal has a unique Secure RPC netname. (See "DES Credential Secure RPC Netname" on page 226 for more information on Secure RPC netnames.)
  - *Principal's public key*.
  - *Principal's encrypted private key*.
- *Principal's LOCAL credential*
- *Server's public keys*. Each directory object stores copies of the public keys of all the servers in that domain. Note that each server's DES credentials are also stored in the cred table.
- *Keyserver copy of principal's private key*. The keyserver has a copy of the private key of the principal that is currently logged in (user or machine).

How Principals Are Authenticated

There are three phases to the authorization process:

- *Preparation phase.* This consists of the setup work performed by an NIS+ administrator prior to the user logging in; for example, creating credential information for the user.
- *Login phase*. This consists of the actions taken by the system when a user logs in.
- *Request phase*. This consists of the actions taken by the software when an NIS+ principal makes a request for an NIS+ service or access to an NIS+ object.

These three phases are described in detail in the following subsections.

Credentials Preparation Phase

The easiest way for an NIS+ administrator to create credential information for users is to use the `nisclient` script. This section describes how to create client information using the NIS+ command set.

Prior to an NIS+ principal logging in, an NIS+ administrator must create DES credential information for that principal (user or machine). The administrator must:

- Create a public key and an encrypted private key for each principal. These keys are stored in the principal's home domain cred table. This can be done with the `nisaddcred` command as described in "Creating Credential Information for NIS+ Principals" on page 237.
- Create server public keys. (See "Updating Public Keys" on page 249.)

Login Phase—Detailed Description

When a principal logs into the system the following steps are automatically performed:

1. The `keylogin` program is run for the principal. The `keylogin` program gets the principal's encrypted private key from the cred table and decrypts it using the principal's login password.

   Note – When a principal's login password is different from his or her Secure RPC password, `keylogin` cannot decrypt it and the user starts getting "cannot decrypt" errors or the command fails without a message. For a discussion of this problem, see "Secure RPC Password Versus Login Password Problem" on page 229.

   ---

2. The principal's decrypted private key is passed to the keyserver which stores it for use during the request phase.

   Note – The decrypted private key remains stored for use by the keyserver until the user does an explicit `keylogout`. If the user simply logs out (or goes home for the day without logging out), the decrypted private key remains stored in the server. If someone with root privileges on a user's machine switched to the user's login ID, that person would then have use of the user's decrypted private key and could access NIS+ objects using the user's access authorization. Thus, for added security, users should be cautioned to perform an *explicit* `keylogout` when they cease work. If they also log out of the system, all they need do is log back in when they return. If they do not explicitly log out, they will have to perform an explicit `keylogin` when they return to work.

Request Phase—Detailed Description

Every time an NIS+ principal requests access to an NIS+ object, the NIS+ software performs a multistage process to authenticate that principal:

1. NIS+ checks the cred table of the object's domain.
   - If the principal has LOCAL credential information, NIS+ uses the domain information contained in the LOCAL credential to find the principal's home domain cred table where it obtains the information it needs.

- If the principal has no credential information, the rest of the process is aborted and the principal is given the authorization access class of nobody.

2. NIS+ gets the user's DES credential from the cred table of the user's home domain. The encrypted private key is decrypted with the user's password and saved by the keyserver.
3. NIS+ obtains the server's public key from the NIS+ directory object.
4. The keyserver takes the principal's decrypted private key and the public key of the object's server (the server where the object is stored) and uses them to create a *common key*.
5. The common key is then used to generate an encrypted *DES key*. To do this, Secure RPC generates a random number which is then encrypted using the common key. For this reason, the DES key is sometimes referred to as the *random key* or the *random DES key*.
6. NIS+ then takes the current time of the principal's server and creates a time stamp that is encrypted using the DES key.
7. NIS+ then creates a 15-second window, which is encrypted with the DES key. This *window* is the maximum amount of time that is permitted between the time stamp and the server's internal clock.
8. NIS+ then forms the principal's DES credential, which is composed of the following items.
   - The principal's Secure RPC netname (`unix.`*identifier@domain*) from the principal's cred table
   - The principal's encrypted DES key from the keyserver
   - The encrypted time stamp
   - The encrypted window
9. NIS+ then passes the following information to the server where the NIS+ object is stored.
   - The access request (whatever it might be)
   - The principal's DES credential
   - Window verifier (encrypted), which is the encrypted window plus one
10. The object's server receives this information.
11. The object's server uses the Secure RPC netname portion of the credential to look up the principal's public key in the cred table of the principal's home domain.
12. The server then uses the principal's public key and the server's private key to regenerate the common key. This common key must match the common key that was generated by the principal's private key and the server's public key.
13. The common key is used to decrypt the DES key that arrived as part of the principal's credential.
14. The server decrypts the principal's time stamp with the newly decrypted DES key and verifies it with the window verifier.
15. The server then compares the decrypted and verified time stamp with the server's current time and proceeds as follows.

a. If the time difference at the server *exceeds* the window limit, the request is denied and the process aborts with an error message. For example, suppose the time stamp is 9:00am and the window is one minute. If the request is received and decrypted by the server after 9:01am, it is denied.

b. If the time stamp is within the window limit, the server checks to see if the time stamp is *greater* than the one previously received from the principal. This ensures that NIS+ requests are handled in the correct order.

   - Requests received out of order are rejected with an error message. For example, if the time stamp is 9:00am and the most recently received request from this principal had a time stamp of 9:02am, the request would be rejected.
   - Requests that have a time stamp equal to the previous one are rejected with an error message. This ensures that a replayed request is not acted on twice. For example, if the time stamp is 9:00am and the most recently received request from this principal also had a time stamp of 9:00am, this request would be rejected.

16. If the time stamp is within the window limit, and greater than the previous request from that principal, the server accepts the request.
17. The server then complies with the request and stores the time stamp from this principal as the most recently received and acted on request.
18. To confirm to the principal that the information received from the server in answer to the request comes from a trusted server, the server encrypts the time stamp with the principal's DES key and sends it back to the principal along with the data.
19. At the principal's end, the returned time stamp is decrypted with the principal's DES key.
    - If the decryption succeeds, the information from the server is returned to the requester.
    - If the decryption fails for some reason, an error message is displayed.

The DES Credential in Detail

The DES credential consists of:

- The principal's *Secure RPC netname* (see "DES Credential Secure RPC Netname" on page 226).
- A *verification* field (see "DES Credential Verification Field" on page 227).

DES Credential Secure RPC Netname

Secure RPC netname. This portion of the credential is used to identify the NIS+ principal. Every Secure RPC netname contains the following three components.

- *Prefix*. The prefix is always the word `unix`.
- *Identifier*. If the principal is a client user, the ID field is the user's UID. If the principal is a client machine, the ID field is the machine's hostname.
- *Domain name*. The domain name is the name of the domain that contains the principal's DES credential (in other words, the principal's home domain).

Note – Remember that an NIS+ principal name *always* has a trailing dot, and a Secure RPC netname *never* has a trailing dot.

TABLE 12–1 Secure RPC Netname Format

Principal	Prefix	Identifier	Domain	Example
User	`unix`	UID	Domain containing user's password entry and the DES credential itself	unix.24601@sales.doc.com
machine	`unix`	hostname	The domain name returned by executing the domainname command on that machine	unix.machine7@sales.doc.com

DES Credential Verification Field

The verification field is used to make sure the credential is not forged. It is generated from the credential information stored in the cred table.

The verification field is composed of:

- The principal's encrypted DES key, generated from the principal's private key and the NIS+ server's public key as described in "Request Phase—Detailed Description" on page 224
- The encrypted time stamp
- The time window

How the DES Credential Is Generated

To generate its DES credential, the principal depends on the `keylogin` command, which must have been executed *before* the principal tries to generate its credential. The `keylogin` command (often referred to simply as a *keylogin*) is executed automatically when an NIS+ principal logs in. See Figure 12–2.

Note – Note that if the principal's login password is different from the principal's Secure RPC password, a successful keylogin cannot be performed. See "Secure RPC Password Versus Login Password Problem" on page 229 for a discussion of this situation.

The purpose of the keylogin is to give the principal access to the principal's private key. `keylogin` fetches the principal's private key from the cred table, decrypts it with the principal's *Secure RPC password* (remember that the private key was originally encrypted with the principal's Secure RPC password), and stores it locally with the keyserver for future NIS+ requests.

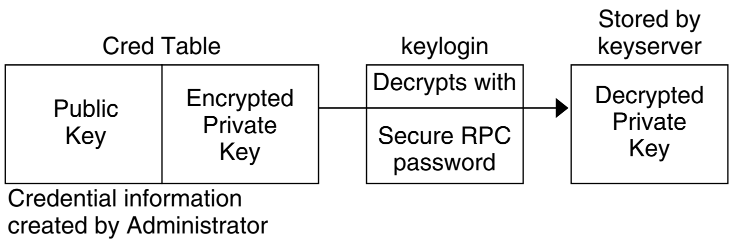


FIGURE 12–1 `keylogin` Generates a Principal's Private Key

To generate its DES credential, the principal still needs the public key of the server to which it will send the request. This information is stored in the principal's directory object. Once the principal has this information, it can form the verification field of the credential.

First, the principal generates a random DES key for encrypting various credential information. The principal uses its own private key (stored in the keyserver) and the server's public key to generate a common key that is used to generate and encrypt the random DES key. It then generates a time stamp that is encrypted with the DES key and combines it with other credential-related information into the verification field.

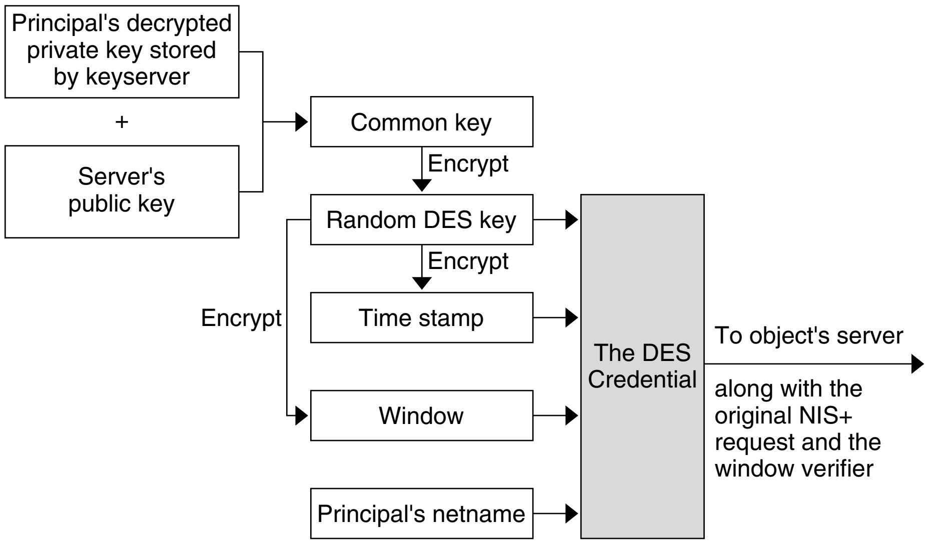


FIGURE 12–2 Creating the DES Credential

Secure RPC Password Versus Login Password Problem

When a principal's login password is different from his or her Secure RPC password, `keylogin` cannot decrypt it at login time because `keylogin` defaults to using the principal's login password, and the private key was encrypted using the principal's Secure RPC password.

When this occurs, the principal can log in to the system, but for NIS+ purposes the principal is placed in the authorization class of nobody because the keyserver does not have a decrypted private key for that user. Since most NIS+ environments are set up to deny the nobody class create, destroy, and modify rights to most NIS+ objects, this results in "permission denied" errors when the user tries to access NIS+ objects.

Note – In this context, *network password* is sometimes used as a synonym for *Secure RPC password*. When prompted for your "network password," enter your Secure RPC password.

To be placed in one of the other authorization classes, a user in this situation must explicitly run the `keylogin` program and give the principal's Secure RPC password when `keylogin` prompts for a password. (See "Keylogin With NIS+" on page 244.)

But an explicit keylogin provides only a temporary solution that is good only for the current login session. The keyserver now has a decrypted private key for the user, but the private key in the user's cred table is still encrypted using the user's Secure RPC password, which is different than the user's login password. The next time the user logs in, the same problem recurs. To permanently solve the problem the user needs to re-encrypt the private key in the cred table to one based on the user's login ID rather than the user's Secure RPC password. To do this, the user needs to run chkey -p as described in "Changing Keys for an NIS+ Principal" on page 245.

Thus, to permanently solve problems related to a difference in Secure RPC password and login password, the user (or an administrator acting for the user) must perform these steps:

1. Log in using the login password.
2. Run the keylogin program to temporarily get a decrypted private key stored in the keyserver and thus gain temporary NIS+ access privileges.
3. Run chkey -p to permanently change the encrypted private key in the cred table to one based on the user's login password.
4. When you are ready to finish this login session, run keylogout.
5. Log off the system with logout.

Cached Public Keys Problems

Occasionally, you might find that even though you have created the proper credentials and assigned the proper access rights, some principal requests still get denied. The most common cause of this problem is the existence of stale objects with old versions of a server's public key. You can usually correct this problem by:

- Running nisupdkeys on the domain you are trying to access. (See "The nisupdkeys Command" on page 250 for information on using the nisupdkeys command and "Stale and Outdated Credential Information" on page 439 for information on how to correct this type of problem.)
- Stopping the NIS+ service on your machine by using the svcadm command, removing /var/nis/NIS_SHARED_DIRCACHE, and then restarting the NIS+ service.

Where Credential-Related Information Is Stored

This section describes where credential-related information is stored throughout the NIS+ namespace.

Credential-related information, such as public keys, is stored in many locations throughout the namespace. NIS+ updates this information periodically, depending on the time-to-live values of the objects that store it, but sometimes, between updates, it gets out of sync. As a result, you may find that operations that should work, do not. lists all the objects, tables, and files that store credential-related information and how to reset it.

Note – The NIS+ service is managed by the Service Management Facility (SMF). Enabling, disabling, or restarting NIS+ daemons such as `rpc.nisd`, `keyserv`, and `nis_cachemgr`, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+. For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

TABLE 12–2 Where Credential-Related Information Is Stored

Item	Stores	To Reset or Change
cred table	NIS+ principal's public key and private key. These are the master copies of these keys.	Use `nisaddcred` to create new credentials; it updates existing credentials. An alternative is `chkey`.
directory object	A copy of the public key of each server that supports it.	Run the `/usr/lib/nis/nisupdkeys` command on the directory object.
keyserver	The secret key of the NIS+ principal that is currently logged in.	Run `keylogin` for a principal user or `keylogin -r`for a principal machine.
NIS+ daemon	Copies of directory objects, which in turn contain copies of their servers' public keys.	Stop the `rpc.nisd` daemon and the cache manager by disabling the NIS+ service, and then remove `NIS_SHARED_DIRCACHE` from `/var/nis`. Then restart the NIS+ service.
Directory cache	A copy of directory objects, which in turn contain copies of their servers' public keys.	Restart the NIS+ cache manager with the `-i` option
cold-start file	A copy of a directory object, which in turn contains copies of its servers' public keys.	Stop the NIS+ service. Remove the `NIS_COLD_START` and `NIS_SHARED_DIRCACHE` files from `/var/nis`. Restart the NIS+ service.

TABLE 12–2 Where Credential-Related Information Is Stored *(Continued)*

Item	Stores	To Reset or Change
`passwd` table	A user's password.	Use the `passwd -r nisplus` command. It changes the password in the NIS+ passwd table and updates it in the cred table.
`passwd` file	A user's password or a machine's superuser password.	Use the `passwd -r nisplus` command, whether logged in as super user or as yourself, whichever is appropriate.
`passwd` map (NIS)	A user's password	Use the `passwd -r nisplus` command.

The `cred` Table in Detail

Credential information for principals is stored in a *cred table*. The `cred` table is one of the 16 standard NIS+ tables. Each domain has one `cred` table, which stores the credential information of client machines that belong to that domain and client users who are allowed to log into them. (In other words, the principals of that domain.) The `cred` tables are located in their domains' `org_dir` subdirectory.

Caution – Never link a `cred` table. Each `org_dir` directory must have its own cred table. Never use a link to some other `org_dir` cred table.

For users, the `cred` table stores LOCAL credential information for all users who are allowed to log into any of the machines in the domain. The `cred` table also stores DES credential information for those users that have the domain as their home domain.

You can view the contents of a `cred` table with the `niscat` command, described in Chapter 19.

The `cred` table, as shown in Table 12–3, has five columns.

TABLE 12–3 `cred` Table Credential Information

	NIS+ Principal Name	Authentication Type	Authentication Name	Public Data	Private Data
Column Name	cname	auth_type	auth_name	public_data	private_data

TABLE 12–3 `cred` Table Credential Information *(Continued)*

	NIS+ Principal Name	Authentication Type	Authentication Name	Public Data	Private Data
User	Fully qualified principal name	LOCAL	UID	GID list	
Machine	Fully qualified principal name	DES	Secure RPC netname	Public key	Encrypted Private key

The Authentication Type column, determines the types of values found in the other four columns.

- *LOCAL*. If the authentication type is LOCAL, the other columns contain a principal user's name, UID, and GID; the last column is empty.
- *DES*. If the authentication type is DES, the other columns contain a principal's name, Secure RPC netname, public key, and encrypted private key. These keys are used in conjunction with other information to encrypt and decrypt a DES credential.

Creating Credential Information

There are several methods of creating and administering credential information:

- Use Solaris Management Console tools if you have them available. They provide easier methods of credential administration and are recommended for administering individual credentials.
- Use the `nisclient` script. This is another easy method of creating or altering credentials for a single principal. Because of its convenience, this is a recommended method of administering individual credentials. gives step by step instructions on using the `nisclient` script to create credential information.
- Use the `nispopulate` script. This is an easy method of creating or altering credentials for a one or more principals who already have information on them stored in NIS maps or `/etc` files. Because of its convenience, this is a recommended method of administering credentials for groups of NIS+ principals. For step by step instructions on using the `nispopulate` script to create credential information, see."Populating NIS+ Tables" on page 96.
- Use the `nisaddcred` command. The section below describes how credentials and credential information are created using `nisaddcred`.

The `nisaddcred` Command

The command used to create credential information is `nisaddcred`.

Note – You can also use the `nispopulate` and `nisclient` scripts to create credential information. They, in turn, use the `nisaddcred` command. These scripts are much easier to use, and more efficient, than the `nisaddcred` command. Unless your network requires special features, you should use the scripts.

The `nisaddcred` command creates, updates, and removes LOCAL and DES credential information. To create credential information, you must have create rights to the proper domain's cred table. To update a credential, you must have modify rights to the cred table or, at least, to that particular entry in the cred table. To delete a credential, you must have destroy rights to the cred table or the entry in the cred table.

- To create or update credentials for another NIS+ principal, use:

  For LOCAL credentials

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

  For DES credentials

  ```
  nisaddcred -p rpc-netname -P principal-name des
  ```

- To update your own credentials, use:

  For LOCAL credentials

  ```
  nisaddcred -local
  ```

  For DES credentials, use:

  ```
  nisaddcred des
  ```

- To remove credentials, use:

  ```
  nisaddcred -r principal-name
  ```

Credential Related Commands

In addition to the `nisaddcred` command described in this chapter, two other commands can provide some useful information about credentials. The `niscat` and `nismatch` commands are described in the following table.

TABLE 12–4 Additional Credential Related Commands

Command	Description	See
`niscat -o`	Lists a directory's properties. By looking in the public key field of the directory's server, you can tell whether the directory object is storing a public key.	"Listing the Object Properties of a Directory" on page 324

TABLE 12–4 Additional Credential Related Commands *(Continued)*

Command	Description	See
`nismatch-`	When run on the cred table, displays credential information for *principal*.	"The `nismatch` and `nisgrep` Commands" on page 362

How `nisaddcred` Creates Credential Information

Use `nisaddcred` to create LOCAL and DES credential information.

LOCAL Credential Information

When used to create LOCAL credential information, `nisaddcred` simply extracts the principal user's UID (and GID) from the principal's login record and places it in the domain's cred table.

DES Credential Information

When used to create DES credential information, `nisaddcred` goes through a two-part process:

1. Forming the principal's Secure RPC netname. A Secure RPC netname is formed by taking the principal's user ID number from the password record and combining it with the domain name (`unix.1050@doc.com`, for example).
2. Generating the principal's private and public keys.

To encrypt the private key, `nisaddcred` needs the principal's Secure RPC password. When the `nisaddcred` command is invoked with the `-des` argument, it prompts the principal for a Secure RPC password. Normally, this password is the same as the principal's login password. (If it is different, the user will have to perform additional steps when logging in, as described in "Secure RPC Password Versus Login Password Problem" on page 229.)

The `nisaddcred` command generates a pair of random, but mathematically related 192-bit authentication keys using the Diffie-Hellman cryptography scheme. These keys are called the Diffie-Hellman key-pair, or simply *key-pair* for short.

One of these is the *private key*, and the other is the *public key*. The public key is placed in the public data field of the cred table. The private key is placed in the private data field, but only after being encrypted with the principal's Secure RPC password.

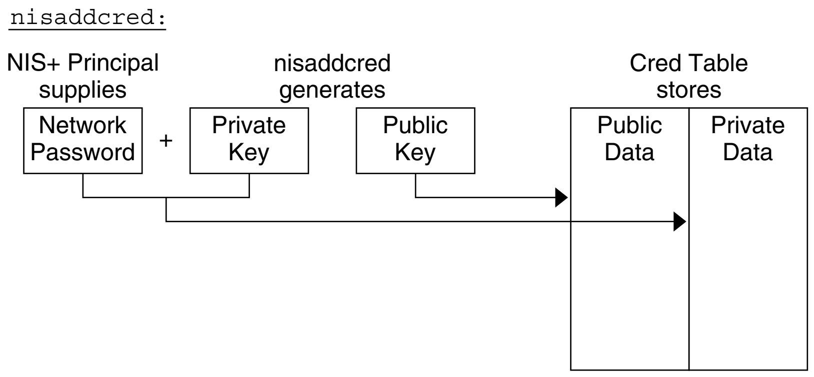


FIGURE 12–3 How nisaddcred Creates a Principal's Keys

The principal's private key is encrypted as a security precaution because the cred table, by default, is readable by all NIS+ principals, even unauthenticated ones.

The Secure RPC Netname and NIS+ Principal Name

When creating credential information, you will often have to enter a principal's *rpc-netname* and *principal-name*. Each has its own syntax:

- *Secure RPC netname*. A Secure RPC netname is a name whose syntax is determined by the Secure RPC protocol. Therefore, it does not follow NIS+ naming conventions:
  - For users, the syntax is: `unix.`*uid@domain*
  - For machines, the syntax is: `unix.`*hostname@domain*

  If a Secure RPC netname identifies a user, it requires the user's UID. If it identifies a machine, it requires the machine's host name. (When used with the `nisaddcred` command it is always preceded by the `-p` (lowercase) flag.)

  A Secure RPC netname always begins with the `unix` (all lowercase) prefix and ends with a domain name. However, because it follows the Secure RPC protocol, the domain name *does not* contain a trailing dot.
- *Principal name*. An NIS+ principal follows the normal NIS+ naming conventions, but it must always be fully qualified. the syntax is: *principal.domain*.

Whether it identifies a client user or a client machine, it begins with the principal's *name*, followed by a dot and the complete domain name, ending in a dot. (When used with `nisaddcred` to create credential information, it is always preceded by the `-P` (uppercase) flag. When used to remove credential information, it does not use the `-P` flag.)

Creating Credential Information for the Administrator

When a namespace is first set up, credential information is created first for the administrators who will support the domain. Once they have credential information, they can create credential information for other administrators, client machines, and client users.

When you try to create your own credential information, you run into a problem of circularity: you cannot create your own credential information unless you have Create rights to your domain's cred table, but if the NIS+ environment is properly set up, you cannot have such rights until you have credentials. You have to step out of the loop somehow. You can do this in one of two ways:

- By creating your credential information while logged in as superuser to your domain's master server.
- By having another administrator create your credential information using a dummy password, then changing your password with the `chkey` command.

In either case, your credential information is thus created by another NIS+ principal. To create your own credential information, follow the instructions in "Creating Credential Information for NIS+ Principals" on page 237.

Creating Credential Information for NIS+ Principals

Credential information for NIS+ principals can be created any time after their domain has been set up; in other words, once a cred table exists.

To create credential information for an NIS+ principal:

- You must have Create rights to the cred table of the principal's home domain.
- The principal must be recognized by the server. This means that:
  - If the principal is a user, the principal must have an entry either in the domain's NIS+ passwd table or in the server's `/etc/passwd` file.
  - If the principal is a machine, it must have an entry either in the domain's NIS+ Hosts table or in the server's.

Once those conditions are met, you can use the `nisaddcred` command with both the `-p` and `-P` options:

For LOCAL credentials

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

For DES credentials

```
nisaddcred -p rpc.netname -P principal-name des
```

Remember these principles:

- You can create both LOCAL and DES credential information for a principal user.
- You can only create DES credential information for a principal machine.
- You can create DES credential information only in the principal's home domain (user or machine).
- You can create LOCAL credential information for a user in both the user's home domain and in other domains.

For User Principals—Example

This example creates both LOCAL and DES credential information for an NIS+ user named `morena` who has a UID of `11177`. She belongs to the `doc.com.` domain, so this example enters her credential information from a principal machine of that domain:

```
client# nisaddcred -p 11177 -P morena.doc.com. local
client# nisaddcred -p unix.11177@sales.doc.com \
   -P morena.doc.com. des
Adding key pair for unix.11177@sales.doc.com
   (morena.doc.com.).
Enter login password:
```

The proper response to the `Enter login password:` prompt is morena's login password. (If you don't know her login password, you can use a dummy password that she can later change using `chkey`, as described in the next example.)

Using a Dummy Password and `chkey`—Example

If you don't know the user's login password, you can use a dummy password as described below.

Table 12–5, shows how another administrator, whose credential information you create using a dummy password, can then use `chkey` to change his or her own password. In this example, you create credential information for an administrator named Eiji who has a UID of 119. Eiji, whose login ID is `eiji`, belongs to the root domain, so you would enter his credential information from the root master server which is named `rootmaster`.

TABLE 12–5 Creating Administrator Credentials: Command Summary

Tasks	Commands
Create LOCAL credential information for Eiji.	`rootmaster# nisaddcred -p 119 -P eiji.doc.com. local`

Create DES credential information for Eiji.	`rootmaster# nisaddcred -p unix.119@doc.com -P eiji.doc.com. des`
	`Adding key pair for unix.119@doc.com (eiji.doc.com.).`

Type dummy password for Eiji.	Enter eiji's login password: nisaddcred: WARNING: password differs from login passwd
Re-enter dummy password.	`Retype password:`
You tell Eiji the dummy password that you used.	
Eiji logs into rootmaster.	`rootmaster% login: eiji`
Eiji enters real login password.	`Password:`
Eiji gets error message but is allowed to log in anyway.	`Password does not decrypt secret key for unix.119@doc.com.`
Eiji runs keylogin.	`rootmaster% keylogin`
Eiji types dummy password.	`Password:` *dummy-password*

Eiji runs `chkey`.	`rootmaster% chkey -p`
	`Updating nisplus publickey database`
	`Generating new key for'unix.119@doc.com'.`

Eiji types real login password.	`Enter login password:`
Eiji re-types real login password.	Retype password: Done.

First, you would create Eiji's credential information in the usual way, but using a dummy login password. NIS+ would warn you and ask you to re-type it. When you did, the operation would be complete. The domain's cred table would contain Eiji's credential information based on the dummy password. The domain's passwd table (or `/etc/passwd` file), however, would still have his login password entry so that he can log on to the system.

Then, Eiji would log in to the domain's master server, typing his *correct* login password (since the login procedure checks the password entry in the passwd table or `/etc/passwd` file). From there, Eiji would first run `keylogin`, using the dummy password (since a `keylogin` checks the cred table), and then use the `chkey -p` command to change the cred entry to the real thing.

Creating in Another Domain—Example

The two previous examples created credential information for a principal user while the principal user was logged in to the master server of the principal's home domain. However, if you have the proper access rights, you can create credential information in another domain. Simply append the domain name to this syntax:

For LOCAL credentials

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

For DES credentials

`nisaddcred -p` *rpc-netname* `-P` *principal-name* `des` *domain-name*

The following example first creates LOCAL and DES credential information for an administrator named Chou in her home domain, which happens to be the root domain, then adds her LOCAL credential information to the `doc.com` domain. Chou's UID is 11155. This command is typed on from the root master server. For simplicity, it assumes you are entering Chou's correct login password.

```
rmaster# nisaddcred -p 11155 -P chou.doc.com. local
rmaster# nisaddcred -p unix.11155@doc.com -P chou.doc.com. des
Adding key pair for unix.11155@doc.com (chou.doc.com.).
Enter login password:
rootmaster# nisaddcred -p 11155 -P chou.doc.com. local doc.com.
```

LOCAL credential information maps a UID to an NIS+ principal name. Although an NIS+ principal that is a client user can have different user IDs in different domains, it can have only one NIS+ principal name. So, if an NIS+ principal such as chou will be logging in from a domain other than her home domain, not only should she have a password entry in that domain, but also a LOCAL credential in that domain's cred table.

For machines—Example

This example creates credential information for a principal *machine*. Its host name is starshine1 and it belongs to the root domain. Therefore, its credential information is created from the root master server. In this example, you create them while logged in as root to the root master; however, if you already have valid credential information and the proper access rights, you could create them while logged in as yourself.

```
rootmaster# nisaddcred -p unix.starshine1@doc.com -P starshine1.doc.com. des
Adding key pair for unix.starshine1@doc.com
 (starshine1.doc.com.).
```

```
Enter starshine1.doc.com.'s root login password:
Retype password:
```

The proper response to the password prompt is the principal machine's superuser password. Of course, you could use a dummy password that would later be changed by someone logged in as superuser to that principal machine.

Administering NIS+ Credential Information

The following sections describe how to use the `nisaddcred` command to administer existing credential information. You must have create, modify, read, and destroy rights to the cred table to perform these operations.

Updating Your Own Credential Information

Updating your own credential information is considerably easier than creating it. Just type the simple versions of the `nisaddcred` command while logged in as yourself:

```
# nisaddcred des
# nisaddcred local
```

To update credential information for someone else, you simply perform the same procedure that you would use to create that person's credential information.

Removing Credential Information

The `nisaddcred` command removes a principal's credential information, but only from the local domain where the command is run.

Thus, to completely remove a principal from the entire system, you must explicitly remove that principal's credential information from the principal's home domain and all domains where the principal has LOCAL credential information.

To remove credential information, you must have modify rights to the local domain's cred table. Use the `-r` option and specify the principal with a full NIS+ principal name:

```
# nisaddcred -r principal-name
```

The following two examples remove the LOCAL and DES credential information of the administrator Morena.`doc.com.` The first example removes both types of credential information from her home domain (`doc.com.`), the second removes her LOCAL credential information from the `sales.doc.com.` domain. Note how they are each entered from the appropriate domain's master servers.

```
rootmaster# nisaddcred -r morena.doc.com.
salesmaster# nisaddcred -r morena.doc.com.
```

To verify that the credential information was indeed removed, run `nismatch` on the cred table, as shown below. For more information about `nismatch`, see Chapter 19.

```
rootmaster# nismatch morena.doc.com. cred.org_dir
salesmaster# nismatch morena.doc.com. cred.org_dir
```

CHAPTER **13**

Administering NIS+ Keys

This chapter describes NIS+ keys and how to administer them.

Tip – Some NIS+ security tasks can be performed more easily with Solaris Management Console tools if you have them available.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Keys

NIS+ keys are used to encrypt NIS+ related information.

This chapter assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that keys play in that system. See Chapter 11 for more information.

For a complete description of NIS+ key-related commands and their syntax and options, see the NIS+ man pages. (The `nisaddcred` command also performs some key-related operations. See Chapter 12 for more information.)

Note – The NIS+ service is managed by the Service Management Facility (SMF). Administrative actions on the NIS+ services, such as enabling, disabling, or restarting, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+. For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

Keylogin With NIS+

When a principal logs in, the login process prompts for a password. That password is used to pass the user through the login security gate and give the user access to the network. The login process also decrypts the user's private key stored in the user's home domain `cred` table and passes that private key to the keyserver. The keyserver then uses that decrypted private key to authenticate the user each time the user accesses an NIS+ object.

Normally, this is the only time the principal is asked to provide a password. However, if the principal's private key in the cred table was encrypted with a password that was *different* from the user's login password, `login` cannot decrypt it using the login password at login time, and thus cannot provide a decrypted private key to the keyserver. (This most often occurs when a user's private key in the cred table was encrypted with a Secure RPC password different from the user's login password.)

Note – In this context, *network password* is sometimes used as a synonym for *Secure RPC password*.

To temporarily remedy this problem, the principal must perform a keylogin, using the `keylogin` command, after every login. (The `-r` flag is used to keylogin the superuser principal and to store the superuser's key in `/etc/.rootkey` on a host.)

For a principal user

```
keylogin
```

For a principal machine (only once)

```
keylogin -r
```

Note, however, that performing an explicit keylogin with the original password provides only a temporary solution good for the current login session only. The private key in the cred table is still encrypted with a password different than the user's login password so the *next* time the user logs in the problem will reoccur. To permanently solve this problem, the user must run `chkey` to change the password used to encrypt the private key to the user's login password (see "Changing Keys for an NIS+ Principal" on page 245).

Changing Keys for an NIS+ Principal

The `chkey` command changes an NIS+ principal's public and private keys that are stored in the cred table. It does not affect the principal's entry either in the passwd table or in the `/etc/passwd` file.

The `chkey` command:

- Generates new keys and encrypts the private key with the password. Run `chkey` with the `-p` option to re-encrypt the existing private key with a new password.
- Generates a new Diffie-Hellman key pair and encrypts the private key with the password you provide. (Multiple Diffie-Hellman key pairs can exist for each principal.) In most cases, however, you do not want a new keypair, you want to re-encrypt your *current* existing private key with the new password. To do this, run `chkey` with the `-p` option.

See the man pages for more information on these subjects.

Note – In an NIS+ environment, when you change your login password with any of the current administration tools or the `passwd` (or `nispasswd`) commands, your private key in the cred table is automatically re-encrypted with the new password for you. Thus, you do not need to explicitly run `chkey` after a change of login password.

The `chkey` command interacts with the keyserver, the cred table, and the passwd table. In order to run `chkey`, you:

- Must have an entry in the passwd table of your home domain. Failure to meet this requirement will result in an error message.
- Must run `keylogin` to make sure that the keyserver has a decrypted private key for you.
- Must have modify rights to the cred table. If you do not have modify rights you will get a "permission denied" type of error message.
- Must know the original password with which the private key in the cred table was encrypted. (In most cases, this your Secure RPC password.)

To use the `chkey` command to re-encrypt your private key with your login password, you first run `keylogin` using the original password, and then use `chkey -p`, as shown in Table 13–1, which illustrates how to perform a `keylogin` and `chkey` for a principal user.

TABLE 13–1 Re-encrypting Your Private Key : Command Summary

Tasks	Commands
Log in.	`Sirius% login` *Login-name*
Provide login password.	`Password:`
If login password and Secure RPC password are different, perform a `keylogin`.	`Sirius% keylogin`
Provide the original password that was used to encrypt the private key.	`Password:` *Secure RPC password*
Run `chkey`.	Sirius% `chkey -p` Updating nisplus publickey database Updating new key for 'unix.1199@Doc.com'.
Enter login password.	`Enter login password:` *login-password*
Re-enter login password	`Retype password:`

Changing the Keys

The following sections describe how to change the keys of an NIS+ principal.

Note – Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information" on page 251.

Changing Root Keys From Root

Table 13–2, shows how to change the keys for the root master server from the root master (as root).

TABLE 13–2 Changing a Root Master's Keys: Command Summary

Tasks	Commands
Create new DES credentials	rootmaster# nisaddcred des
Find the NIS+ service	`rootmaster# svcs \*nisplus\*`
Stop the NIS+ service	`rootmaster# svcadm disable -t /network/rpc/nisplus:default`
Remove the `-S 0` security option	Edit the `/lib/svc/method/nisplus` file to remove the `-S 0` option
Restart NIS+ service with no security	`# svcadm enable network/rpc/nisplus`
Perform a keylogout (previous keylogin is now out of date)	`rootmaster# keylogout -f`
Update the keys in the directories served by the master	`rootmaster# nisupdkeys` *dirs*
Find the NIS+ service	`rootmaster# svcs \*nisplus\*`
Stop the NIS+ service	`rootmaster# svcadm disable -t /network/rpc/nisplus:default`
Add the `-S 0` security option	Edit the `/lib/svc/method/nisplus` file to add the `-S 0` option
Restart NIS+ daemon with default security	`# svcadm enable network/rpc/nisplus`
Perform a keylogin	`rootmaster# keylogin`

Where:

- *dirs* are the directory objects you wish to update. (That is, the directory objects that are served by `rootmaster`.)

In the first step of the process outlined in Table 13–2, `nisaddcred` updates the cred table for the root master, updates `/etc/.rootkey` and performs a keylogin for the root master. At this point the directory objects served by the master have not been updated and their credential information is now out of synch with the root master. The subsequent steps described in Table 13–2 are necessary to successfully update all the objects.

Note – Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information" on page 251.

Changing Root Keys From Another Machine

To change the keys for the root master server from some other machine you must have the required NIS+ credentials and authorization to do so.

TABLE 13–3 Remotely Changing Root Master Keys: Command Summary

Tasks	Commands
Create the new DES credentials	`othermachine% nisaddcred -p` *principal* `-P` *nisprincipal* `des`
Update the directory objects.	`othermachine% nisupdkeys`*dirs*
Update `/etc.rootkey.`	`othermachine% keylogin -r`
Reinitialize othermachine as client	`othermachine% nisinit -cH`

Where:

- *principal* is the root machine's Secure RPC netname. For example:`unix.rootmaster@doc.com` (no dot at the end).
- *nis-principal* is the root machine's NIS+ principal name. For example, `rootmaster.doc.com.` (a dot at the end).
- *dirs* are the directory objects you want to update (that is, the directory objects that are served by `rootmaster`).

When running `nisupdkeys` be sure to update all relevant directory objects at the same time. In other words, do them all with one command. Separate updates may result in an authentication error.

Note – Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information" on page 251.

Changing the Keys of a Root Replica From the Replica

To change the keys of a root replica from the replica, use these commands:

```
replica# nisaddcred des
replica# nisupdkeys dirs
```

Where:

dirs are the directory objects you wish to update, (that is, the directory objects that are served by `replica`).

When running `nisupdkeys` be sure to update all relevant directory objects at the same time. In other words, do them all with one command. Separate updates may result in an authentication error.

Note – Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information" on page 251.

Changing the Keys of a Nonroot Server

To change the keys of a nonroot server (master or replica) from the server, use these commands:

```
subreplica# nisaddcred des
subreplica# nisupdkeys parentdir dirs
```

Where:

- *parentdir* is the non-root server's parent directory (that is, the directory containing `subreplica`'s NIS+ server).
- *dirs* are the directory objects you want to update (that is, the directory objects that are served by `subreplica`).

When running `nisupdkeys` be sure to update all relevant directory objects at the same time. In other words, do them all with one command. Separate updates may result in an authentication error.

Note – Whenever you change a server's keys, you must also update the key information of all the clients in that domain, as explained in "Updating Client Key Information" on page 251.

Updating Public Keys

The public keys of NIS+ servers are stored in several locations throughout the namespace. When new credential information is created for the server, a new key pair is generated and stored in the cred table. However, namespace directory objects still have copies of the server's *old* public key. The `nisupdkeys` command is used to update those directory object copies.

The `nisupdkeys` Command

If a new keypair is generated because the old key pair has been compromised or the password used to encrypt the private key is forgotten, the `nisupdkeys` can be used to update the old public key in the directory objects.

The `nisupdkeys` command can:

- Update the key of one particular server
- Update the keys of all the servers that support an NIS+ directory object
- Remove a server's public key from the directory object
- Update a server's IP address, if that has changed

However, `nisupdkeys` cannot update the `NIS_COLD_START` files on the principal machines. To update their copies of a server's keys, NIS+ clients should run the `nisclient` command. Or, if the NIS+ cache manager is running and more than one server is available in the coldstart file, the principals can wait until the time-to-live expires on the directory object. When that happens, the cache manager automatically updates the cold-start file. The default time-to-live is 12 hours.

To use the `nisupdkeys` command, you must have modify rights to the NIS+ directory object.

Updating Public Keys Arguments and Examples

The `nisupdkeys` command is located in `/usr/lib/nis`. The `nisupdkeys` command uses the following arguments (for a complete description of the `nisupdkeys` command and a full list of all its arguments, see the `nisupdkeys` man page).

TABLE 13–4 `nisupdkeys` Arguments

Argument	Effect
(no argument)	Updates all keys of servers for current domain.
directoryname	Updates the keys of the directory object for the named directory.
`-H` *servername*	Updates the keys of the named server for the current domain directory object. A fully qualified host name can be used to update the keys of servers in other domains.
`-s -H` *servername*	Updates the keys of all the directory objects served by the named server.
`-C`	Clears the keys.

Table 13–5 gives an example of updating a public key.

TABLE 13–5 Updating a Public Key: Command Examples

Tasks	Commands

Update all keys of all servers of the current domain (`doc.com`).	`rootmaster# /usr/lib/nis/nisupdkeys`
	Fetch Public key for server rootmaster.doc.com.
	`netname='unix.rootmaster@doc.com'`
	Updating rootmaster.doc.com.'s public key.
	`Public key:` *public-key*

Update keys of all servers supporting the `sales.doc.com` domain directory object.	`salesmaster# nisupdkeys sales.doc.com` (Screen notices not shown)
Update keys for a server named `master7` in all the directories that store them.	`rootmaster# nisupdkeys -H master7`
Clear the keys stored by the `sales.doc.com` directory object.	`rootmaster# nisupdkeys -C sales.doc.com`
Clear the keys for the current domain directory object for the server named `master7`.	`rootmaster# nisupdkeys -C -H master7`

Updating IP Addresses

If you change a server's IP address, or add additional addresses, you need to run `nisupdkeys` to update NIS+ address information.

To update the IP addresses of one or more servers, use the `nisupdkeys` command `-a` option.

To update the IP addresses of servers of a given domain

```
rootmaster# nisupdkeys -a domain
```

To update the IP address of a particular server

```
rootmaster# nisupdkeys -a -H server
```

Updating Client Key Information

Whenever you change any server's keys, you must update all of the clients as well. Remember, that all NIS+ servers are also NIS+ clients, so if you update the keys on one server, you must update key information on all other machines in the domain regardless of whether or not they are NIS+ servers or ordinary clients.

There are three ways to update client key information:

- The easiest way to update an individual client's key information is by running the `nisclient` script on the client.
- Another way to update an individual client's key information is by running the `nisinit` command on the client as described in "Initializing a Client" on page 333.
- You can globally update client key information for all the machines in a domain by shortening the Time To Live value of the domain's directory object as explained in "Globally Updating Client Key Information" on page 252.

Globally Updating Client Key Information

After changing a server's keys, you can globally update client key information for all the machines in a domain by:

1. **Use the `nischttl` command to reduce the Time To Live (TTL) value of the domain's directory object so that the value expires almost immediately.**

   For example, if you have changed the keys for a server in the `sales.doc.com.` domain, to reduce the directory's TTL value to one minute you would enter:

   ```
   client% nischttl 60 sales.doc.com.
   ```

2. **When the directory's TTL value expires, the cache manager expires the entry and then obtains the new, updated information for clients.**

3. **Once the directory object's TTL value has expired, reset the directory object's TTL to its default value.**

   For example, to reset the TTL value to 12 hours for the `sales.doc.com.` domain's directory object, you would enter:

   ```
   client% nischttl 12h sales.doc.com.
   ```

   See "The `nischttl` Command" on page 341 for more information on working with TTL values.

CHAPTER **14**

Administering Enhanced Security Credentials

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Diffie-Hellman Extended Key

NIS+ offers increased security at the `RPC`(3N) layer beyond 192 bit Diffie-Hellman (`RPC`(3N) security flavor AUTH_DH) using the RPCSEC_GSS `RPC`(3N) security flavor. See the `nisauthconf`(1M) command for a list of which security mechanisms are available on the system. Along with more stringent cryptographic strength, these security mechanisms also provide integrity for each NIS+ transaction. That is, the data for each NIS+ transaction is verified that it has not been modified.

System administraters can take advantage of the more stringent security mechanisms either by running `nisauthconf`(1M) before the NIS+ server environment is constructed or after, using the guidelines below.

Note – AUTH_DH was formerly referred to as AUTH_DES.

Transitioning to a New Public Key-based Security Mechanism

The more stringent security mechanisms of the public key cryptography family such as Diffie-Hellman 640 bit (dh640-0) will require new credentials for each principal to be added to the existing `cred` table. The procedure outlined below is for a system currently running with Diffie-Hellman 192 bit (RPC security flavor AUTH_DH) security that will be converted to running with Diffie-Hellman 640 bit (RPC security flavor RPCSEC_GSS) security. Although this transition document highlights the most likely case, the principles are the same for converting from any one security mechanism type of the public key cryptography family to another security mechanism of the public key cryptography family.

Note – The following example assumes that $PATH includes `/usr/lib/nis`.

Configuring NIS+ Security Mechanisms

NIS+ security configuration is done with the `nisauthconf`(1M) command. `nisauthconf` takes a list of security mechanisms in order of preference. A security mechanism may use one or more authentication flavors specified in `secure_rpc`(3N). If **`des`** is the only specified mechanism, then NIS+ only uses AUTH_DH (formerly AUTH_DES) for authentication with other NIS+ clients and servers. Any other security mechanisms after des will be ignored by NIS+, except for `nisaddcred`(1M).

```
nisauthconf [-v] [mechanism, ...]
```

Where mechanism is a RPC security mechanism that is available on the system. See `nisauthconf`(1M) for the list of mechanisms available. If no mechanisms are specified, then a list of currently configured mechanisms is printed.

Creating New Security Mechanism Credentials

Credential information for the new mechanism must be created for each NIS+ user and host principal. In order to do this, on one of the machines running NIS+, the `nisauthconf`(1M) command must be run to allow the creation of new credentials while the system continues to authenticate with the current mechanism. Also see "Creating Credential Information for NIS+ Principals" on page 237 for details on credential creation basics.

New Security Mechanism Credentials –Example

Converting des to dh640-0; the `nisauthconf` should be run as root and the `nisaddcred` should be run as any principal that has Create rights in the principal's home directory. The server is named server1 and the user principal is named morena. User morena has UID 11177.

```
client# nisauthconf des dh640-0
client% nisaddcred -P server1.doc.com. -p unix.server1@doc.com dh640-0
      (screen notices not shown)
client% nisaddcred -P morena.doc.com. -p unix.11177@doc.com -ldummy-password dh640-0
      (screen notices not shown)
```

Adding New Keys to NIS+ Directory Objects

Once the new credentials have been generated for all the servers, run `nisupdkeys`(1m) to add the new public keys to all the directory objects served by these servers. To use the `nisupdkeys`(1m) command, you must have modify rights to the NIS+ directory object. See "Updating Public Keys" on page 249 for more details.

Caution – All servers that serve these NIS+ directories and all clients that access these directories must be running Solaris 7 or later.

Adding New Public Keys to NIS+ Directory Objects—Example

In this example, the directories that are being served by the servers with new public keys are `doc.com`, `org_dir.doc.com.`, `groups_dir.doc.com.`. The update will be done as the master server principal. Before running the new mechanism, `nisupdkeys` needs to be configured with `nisauthconf`(1M). In this example, the current authentication mechanism is des and the new mechanism is dh640-0.

```
masterserver#    nisauthconf dh640-0 des
masterserver#    nisupdkeys doc.com.
            (screen notices not shown)
masterserver#  nisupdkeys org_dir.doc.com.
            (screen notices not shown)
masterserver#    nisupdkeys groups_dir.doc.com.
            (screen notices not shown)
```

Configuring NIS+ Servers to Accept New Security Mechanism Credentials

On each server, configure NIS+ authentication so that it accepts both the old and new credentials. This will require running `nisauthconf`(1m) and `keylogin`(1) and restarting `keyserv`(1m). The `keylogin`(1) command stores the keys for each mechanism in the `/etc/.rootkey`. See "Keylogin With NIS+" on page 244 for basic `keylogin` details.

Configuring NIS+ Servers to Accept New Security Mechanism Credentials—Example

In this example, the current authentication mechanism is des and the new mechanism is dh640-0. Note the ordering is significant here; any mechanisms after the des entry will be ignored for client and server NIS+ authentication.

```
server# nisauthconf dh640-0 des
server# keylogin -r
        (screen notices not shown)
server# /etc/reboot
```

Configuring Machines to Use New Security Mechanism Credentials

Now that the servers can accept the new credentials, the machines can be converted to authenticate via the new credentials. To do this, run `nisauthconf`(1M) and `keylogin`(1) as root and reboot.

Configuring Machines to Use New Security Mechanism Credentials—Examples

In this example, the new mechanism is dh640-0 but the system will also attempt authentication with des credentials if the dh640-0 ones are not available or do not succeed.

```
workstation# nisauthconf dh640-0 des
workstation#  keylogin -r
        (screen notices not shown)
workstation# /etc/reboot
```

In the next example, the new mechanism is dh640-0 and authentication will *only* be attempted with this mechanism. Before configuring any system to authenticate via the new mechanism exclusively, the cached directory objects must be refreshed to include the keys for the new mechanism. This can be verified with `nisshowcache`(1M) . An alternative to waiting for the cached directory objects to time out and be refreshed is the following: stop the NIS+ service, then construct a new `NIS_COLD_START` by using `nisinit`(1M), and then restart the NIS+ service.

Manually Refresh Directory Objects—Example NETNAMER

To manually refresh directory objects, use the svcadm command. See the `svcadm`(1M) man page for more information.

```
# svcadm disable -t /network/rpc/nisplus:default
# nisinit -cH masterserver
# svcadm enable /network/rpc/nisplus:default
```

Caution – The machine principal and all users of this machine must have dh640-0 credentials in the `cred` table before the system can be configured to authenticate exclusively with dh640–0.

```
workstation# nisauthconf dh640-0
workstation#  keylogin -r
        (screen notices not shown)
workstation# /etc/reboot
```

Changing the Password Protecting New Credentials

For each user given new credentials with the dummy passwd, they need to run chkey(1) to convert to their login password. For more information, see "Changing Keys for an NIS+ Principal" on page 245.

Change Password Protecting New Credentials—Example

Run chkey(1) to convert to your login password:

```
# chkey -p
        (screen notices not shown)
```

Configuring Servers to Accept only New Security Mechanism Credentials

When converting from a lower grade security mechanism to a higher one, the maximum security benefit is achieved by configuring the NIS+ servers to only accept credentials of the new higher grade security mechanism type. Do this only after the servers have been successfully configured to authenticate via the old and the new mechanism.

Before configuring any system to authenticate via the new mechanism exclusively, the cached directory objects must be refreshed to include the keys for the new mechanism and verified with nisshowcache(1M).

Configuring Servers to Accept only New Security Mechanism Credentials—Example

Run `nisauthconf`(1m) on each NIS+ server and reboot. In this example, the NIS+ server will be configured to *only* accept authentication of dh640-0 credentials.

```
server#  nisauthconf dh640-0
server# /etc/reboot
```

Optionally, the directory objects can now be updated to remove the old public keys. This should be done from the master server and `nisupdkeys`(1m) should be run once for each directory served by the servers authenticating only with the new security mechanism. In this example, the directories to be updated are `doc.com`, `org_dir.doc.com.`, and `groups_dir.doc.com.`

```
masterserver#    nisupdkeys doc.com.
            (screen notices not shown)
masterserver#  nisupdkeys org_dir.doc.com.
            (screen notices not shown)
masterserver#    nisupdkeys groups_dir.doc.com.
```

Removing Old Credentials from the cred Table

If desired, the credentials of the old security mechanism can be removed from the `cred` table. You must have modify rights to the local domain's `cred` table. See "Removing Credential Information" on page 241 for more details.

Removing old Credentials from the cred Table—Example

In this example, the principal morena.doc.com will have her des credentials removed from the `cred` table.

```
master# nisaddcred -r morena.doc.com. dh192-0
```

CHAPTER **15**

Administering NIS+ Access Rights

This chapter describes NIS+ access rights and how to administer them.

Tip – Some NIS+ security tasks can be performed more easily with Solaris Management Console tools if you have them available.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Access Rights

NIS+ access rights determine what operations NIS+ users can perform and what information they have access to. This chapter assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that access rights play in that system (see Chapter 11 for this information).

For a complete description of NIS+ access-related commands and their syntax and options, see the NIS+ man pages.

Introduction to Authorization and Access Rights

See "NIS+ Authorization and Access—Introduction" on page 214 for a description of how authorization and access rights work with NIS+ credentials and authentication to provide security for the NIS+ namespace.

Authorization Classes—Review

As described more fully in "Authorization Classes" on page 215, NIS+ access rights are assigned on a class basis. There are four different NIS+ classes:

- *Owner*. The owner class is a *single* NIS+ principal. By default, an object's owner is the principal that created the object. However, an object's owner can transfer ownership to another principal who then becomes the new owner.
- *Group.* The group class is a *collection* of one or more NIS+ principals. An NIS+ object can have only one NIS+ group.
- *World*. The world class contains all NIS+ principals that are authenticated by NIS+ (in other words, everyone in the owner and group class, plus everyone else who presents a valid DES credential).
- *Nobody*. The nobody class is composed of anyone who is not properly authenticated (in other words, anyone who does not present a valid DES credential).

Access Rights—Review

As described more fully in "Introduction to NIS+ Access Rights" on page 218, there are four types of NIS+ access rights:

- *Read.* A principal with read rights to an object can view the contents of that object.
- *Modify.* A principal with modify rights to an object can change the contents of that object.
- *Destroy.* A principal with Destroy rights to an object can delete the object.
- *Create.* A principal with create rights to a higher level object can create new objects within that level. In other words, if you have create rights to an NIS+ directory object, you can create new tables within that directory. If you have create rights to an NIS+ table, you can create new columns and entries within that table.

Keep in mind that these rights logically evolve down from directory to table to table column and entry levels. For example, to create a new table, you must have create rights for the NIS+ directory object where the table will be stored. When you create that table, you become its default owner. As owner, you can assign yourself create rights to the table which allows you to create new entries in the table. If you create new entries in a table, you become the default owner of those entries. As table owner, you can also grant table level create rights to others. For example, you can give your table's group class table level create rights. In that case, any member of the table's group can create new entries in the table. The individual member of the group who creates a new table entry becomes the default owner of that entry.

Concatenation of Access Rights

Authorization classes are concatenated. In other words, the higher class usually belongs to the lower class and automatically gets the rights assigned to the lower class. It works like this:

- *Owner class.* An object's owner may, or may not, belong to the object's group. If the owner does belong to the group, then the owner gets whatever rights are assigned to the group. The object's owner automatically belongs to the world and nobody classes, so the owner automatically gets whatever rights that object assigns to those two classes.
- *Group class.* Members of the object's group automatically belong to the world and nobody classes, so the group members automatically get whatever rights that object assigns to world and nobody.
- *World class*. The world class automatically gets the same rights to an object that are given to the nobody class.
- *Nobody class*. The nobody class only gets those rights an object specifically assigns to the nobody class.

The basic principle that governs this is that access rights override the absence of access rights. In other words, a higher class can have *more* rights than a lower class, but not *fewer* rights. (The one exception to this rule is that if the owner is not a member of the group, it is possible to give rights to the group class that the owner does not have.)

How Access Rights Are Assigned and Changed

When you create an NIS+ object, NIS+ assigns that object a default set of access rights for the owner and group classes. By default, the owner is the NIS+ principal who creates the object. The default group is the group named in the `NIS_GROUP` environment variable.

Specifying Different Default Rights

NIS+ provides two different ways to change the default rights that are automatically assigned to an NIS+ object when it is created.

- The `NIS_DEFAULTS` environment variable. `NIS_DEFAULTS` stores a set of security-related default values, one of which is access rights. These default access rights are the ones automatically assigned to an object when it is created. (See "Displaying NIS+ Defaults—The `nisdefaults` Command" on page 273 for details.)

  If the value of the `NIS_DEFAULTS` environment variable is changed, objects created after the change are assigned the new values. However, previously created objects are not affected.
- The `-D` option, which is available with several NIS+ commands. When you use the `-D` option as part of the command to create an NIS+ object, it overrides the default rights specified by the `NIS_DEFAULTS` environment variable and allows you to explicitly specify an initial set of rights for that object. (See "Specifying Nondefault Security Values at Creation Time" on page 277 for details.)

Changing Access Rights to an Existing Object

When an NIS+ object is created, it comes into existence with a default set of access rights (from either the `NIS_DEFAULTS` environment variable or as specified with the `-D` option). These default rights can be changed with the

- `nischmod` command
- `nistbladm` command for table columns

Table, Column, and Entry Security

NIS+ tables allow you to specify access rights on the table three ways:

- You can specify access rights to the *table* as a whole.
- You can specify access rights to each *entry* (row) by itself.
- You can specify access rights to each table *column* individually.

A *field* is the intersection between a column and an entry (row). All data values are entered in fields.

These column- and entry level access rights allow you to specify *additional* access to individual rows and columns that override table level restrictions, but column and entry level rights cannot be *more* restrictive than the table as a whole:

- *Table.* The table level is the base level. Access rights assigned at the table level apply to every piece of data in the table unless specifically modified by a column or entry exception. Thus, the table level rights should be the *most* restrictive.

  Note – Remember that authorization classes concatenate. A higher class gets the rights assigned to lower classes. See "Concatenation of Access Rights" on page 263.

- *Column.* Column-level rights allow you to grant additional access rights on a column-by-column basis. For example, suppose the table level granted no access rights whatsoever to the world and nobody classes. In such a case, no one in those two classes could read, modify, create, or destroy any data in the table. You could use column-level rights to override that table level restriction and permit members of the world class the right to view data in a particular column.

  On the other hand, if the table level grants table-wide read rights to the owner and group classes, you cannot use column-level rights to prevent the group class from having read rights to that column.
- *Entry (row).* entry level rights allow you to grant additional access rights on a row-by-row basis. For example, this allows you to permit individual users to change entries that apply to them, but not entries that apply to anyone else.

  Keep in mind that an entry's group does not have to be the same as the table's group. Tables and entries can have different groups. This means that you can permit members of a particular group to work with one set of entries while preventing them from affecting entries belonging to other groups.

Table, Column, Entry Example

Column- or entry level access rights can provide additional access in two ways: by extending the rights to additional principals or by providing additional rights to the same principals. Of course, both ways can be combined. Following are some examples.

Assume a table object granted read rights to the table's owner:

TABLE 15–1 Table, Column, Entry Example 1

	Nobody	Owner	Group	World
Table Access Rights:	`----`	`r---`	`----`	`----`

This means that the table's owner could read the contents of the entire table but no one else could read anything. You could then specify that Entry-2 of the table grant read rights to the group class:

TABLE 15–2 Table, Column, Entry Example 2

	Nobody	Owner	Group	World
Table Access Rights:	`----`	`r---`	`----`	`----`
Entry-2 Access Rights:	`----`	`----`	`r---`	`----`

Although only the owner could read all the contents of the table, any member of the table's group could read the contents of that particular entry. Now, assume that a particular column granted read rights to the world class:

TABLE 15–3 Table, Column, Entry Example 3

	Nobody	Owner	Group	World
Table Access Rights:	`----`	`r---`	`----`	`----`
Entry-2 Access Rights:	`----`	`----`	`r---`	`----`
Column-1 Access Rights:	`----`	`----`	`----`	`r---`

Members of the world class could now read that column *for all entries* in the table . Members of the group class could read everything in Column-1 (because members of the group class are also members of the world class) and also all columns of Entry-2. Neither the world nor the group classes could read any cells marked `*NP*` (for Nor Permitted).

TABLE 15–4 Table, Column, Entry Example 4

	Col 1	Col 2	Col 2
Entry-1	contents	`*NP*`	`*NP*`
Entry-2	contents	contents	contents
Entry-3	contents	`*NP*`	`*NP*`
Entry-4	contents	`*NP*`	`*NP*`
Entry-5	contents	`*NP*`	`*NP*`

Rights at Different Levels

This section describes how the four different access rights (read, create, modify, and destroy) work at the four different access levels (directory, table, column, and entry).

The objects that these various rights and levels act on are summarized in Table 15–5:

TABLE 15–5 Access Rights and Levels and the Objects They Act Upon

	Directory	Table	Column	Entry
Read	List directory contents	View table contents	View column contents	View entry (row) contents
Create	Create new directory or table objects	Add new entries (rows)	Enter new data values in a column	Enter new data values in an entry (row)
Modify	Move objects and change object names	Change data values anywhere in table	Change data values in a column	Change data values in an entry (row)

TABLE 15–5 Access Rights and Levels and the Objects They Act Upon *(Continued)*

	Directory	Table	Column	Entry
Destroy	Delete directory objects such as tables	Delete entries (rows)	Delete data values in a column	Delete data values in an entry (row)

Read Rights

- *Directory*. If you have read rights to a directory, you can list the contents of the directory.
- *Table*. If you have read rights to a table, you can view all the data in that table.
- *Column*. If you have read rights to a column, you can view all the data in that column.
- *Entry*. If you have read rights to an entry, you can view all the data in that entry.

Create Rights

- *Directory*. If you have create rights at the directory level, you can create new objects in the directory such as new tables.
- *Table*. If you have create rights at the table level, you can create new entries. (You cannot add new columns to an existing table regardless of what rights you have.)
- *Column*. If you have create rights to a column, you can enter new data values in the fields of that column. You cannot create new columns.
- *Entry*. If you have create rights to an entry, you can enter new data values in the fields of that row. (Entry level create rights do not permit you to create new rows.)

Modify Rights

- *Directory*. If you have modify rights at the directory level, you can move or rename directory objects.
- *Table*. If you have modify rights at the table level, you can change any data values in the table. You can create (add) new rows, but you cannot create new columns. If an existing field is blank, you can enter new data in it.
- *Column*. If you have modify rights to a column, you can change the data values in the fields of that column.
- *Entry*. If you have modify rights to an entry, you can change the data values in the fields of that row.

Destroy Rights

- *Directory*. If you have destroy rights at the directory level, you can destroy existing objects in the directory such as tables.

- *Table*. If you have destroy rights at the table level, you can destroy existing entries (rows) in the table but not columns. You cannot destroy existing columns in a table: you can only destroy entries.
- *Column*. If you have destroy rights to a column, you can destroy existing data values in the fields of that column.
- *Entry*. If you have destroy rights to an entry, you can destroy existing data values in the fields of that row.

Where Access Rights Are Stored

An object's access rights are specified and stored as part of the object's definition. This information is not stored in an NIS+ table.

Viewing an NIS+ Object's Access Rights

The access rights can be viewed by using the `niscat` command:

```
niscat -o objectname
```

Where *objectname* is the name of the object whose access rights you want to view.

This command returns the following information about an NIS+ object:

- *Owner*. The single NIS+ principal who has ownership rights. This is usually the person who created the object, but it could be someone to whom the original owner transferred ownership rights.
- *Group*. The object's NIS+ group.
- *Nobody class access rights*. The access rights granted to everyone, whether they are authenticated (have a valid DES credential) or not.
- *Owner class access rights*. The access rights granted to the object's owner.
- *Group class access rights*. The access rights granted to the principals in the object's group.
- *World class access rights*. The access rights granted to all authenticated NIS+ principals.

Access rights for the four authorization classes are displayed as a list of 16 characters, like this:

```
    r---rmcdr---r---
```

Each character represents a type of access right:

- `r` represents read rights.
- `m` represents modify rights.
- `d` represents destroy rights.

- `c` represents create rights.
- `-` represents no access rights.

The first four characters represent the access rights granted to nobody, the next four to the owner, the next four to the group, and the last four to the world:

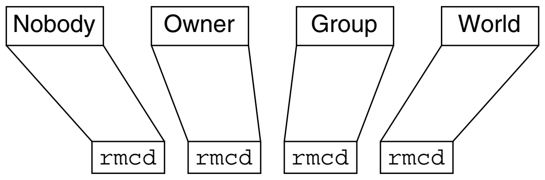


FIGURE 15–1 Access Rights Display

Note – Unlike UNIX file systems, the first set of rights is for nobody, not for the owner.

Default Access Rights

When you create an object, NIS+ assigns the object a default owner and group, and a default set of access rights for all four classes. The default owner is the NIS+ principal who creates the object. The default group is the group named in the `NIS_GROUP` environment variable. Table 15–6, shows the default access rights.

TABLE 15–6 Default Access Rights

Nobody	Owner	Group	World
-	read	read	read
-	modify	-	-
-	create	-	-
-	destroy	-	-

If you have the `NIS_DEFAULTS` environment variable set, the values specified in `NIS_DEFAULTS` will determine the defaults that are applied to new objects. When you create an object from the command line, you can use the `-D` flag to specify values other than the default values.

How a Server Grants Access Rights to Tables

This section discusses how a server grants access to tables objects, entries, and columns during each type of operation: read, modify, destroy, and create.

Note – At security level 0, a server enforces no NIS+ access rights and all clients are granted full access rights to the table object. Security level 0 is only for administrator setup and testing purposes. Do not use level 0 in any environment where ordinary users are performing their normal work.

The four factors that a server must consider when deciding whether to grant access are:

- The type of operation requested by the principal
- The table, entry, or column the principal is trying to access
- The authorization class the principal belongs to for that particular object
- The access rights that the table, entry, or column has assigned to the principal's authorization class

After authenticating the principal making the request by making sure the principal has a valid DES credential, an NIS+ server determines the type of operation and the object of the request.

- *Directory*. If the object is a directory or group, the server examines the object's definition to see what rights are granted to the four authorization classes, determines which class the principal belongs to, and then grants or denies the request based on the principal's class and the rights assigned to that class.
- *Table*. If the object is a table, the server examines the table's definition to see what table level rights are granted to the four authorization classes, and determines which class the principal belongs to. If the class to which the principal belongs does not have table level rights to perform the requested operation, the server then determines which row or column the operation concerns and determines if there are corresponding row- or column-level access rights permitting the principal to perform the requested operation.

Specifying Access Rights in Commands

This section assume an NIS+ environment running at security level 2 (the default level).

This section describes how to specify access rights, as well as owner, group owner, and object, when using any of the commands described in this chapter.

Syntax for Access Rights

This subsection describes the access rights syntax used with the various NIS+ commands that deal with authorization and access rights.

Class, Operator, and Rights Syntax

Access rights, whether specified in an environment variable or a command, are identified with three types of arguments: *class*, *operator*, and *right*.

- *Class*. Class refers to the type of NIS+ principal (authorization class) to which the *rights* will apply.

TABLE 15–7 Access Rights Syntax—Class

Class	Description
`n`	Nobody: all unauthenticated requests
`o`	The owner of the object or table entry
`g`	The group owner of the object or table entry
`w`	World: all authenticated principals
`a`	All: shorthand for owner, group, and world (this is the default)

- *Operator.* The operator indicates the kind of operation that will be performed with the *rights*.

TABLE 15–8 Access Rights Syntax—Operator

Operator	Description
+	Adds the access rights specified by *right*
-	Revokes the access rights specified by *right*
=	Explicitly changes the access rights specified by *right*; in other words, revokes all existing rights and replaces them with the new access rights.

- *Rights*. The rights are the access rights themselves. The accepted values for each are listed below.

TABLE 15–9 Access Rights Syntax—Rights

Right	Description
`r`	Reads the object definition or table entry

TABLE 15–9 Access Rights Syntax—Rights *(Continued)*

Right	Description
`m`	Modifies the object definition or table entry
`c`	Creates a table entry or column
`d`	Destroys a table entry or column

You can combine operations on a single command line by separating each operation from the next with a comma (,).

TABLE 15–10 Class, Operator, and Rights Syntax—Examples

Operations	Syntax
Add read access rights to the *owner* class	`o+r`
Change owner. group, and world classes' access rights to modify only from whatever they were before	`a=m`
Add read and modify rights to the world and nobody classes	`wn+m`
Remove all four rights from the group, world, and nobody classes	`gwn-rmcd`
Add create and destroy rights to the owner class and add read and modify rights to the world and nobody classes	`o+cd,wn+rm`

Syntax for Owner and Group

- *Owner*. To specify an owner, use an NIS+ principal name.
- *Group*. To specify an NIS+ group, use an NIS+ group name with the domain name appended.

Remember that principal names are fully qualified (*principalname.domainname*).

For owner

principalname

For group

groupname.domainname

Syntax for Objects and Table Entries

Objects and table entries use different syntaxes.

- Objects use simple object names.
- Table entries use indexed names.

For objects

objectname

For table entries

columnname=value] ,*tablename*

Note – In this case, the brackets are part of the syntax.

Indexed names can specify more than one column-value pair. If so, the operation applies only to the entries that match *all* the column-value pairs. The more column-value pairs you provide, the more stringent the search.

For example:

TABLE 15–11 Object and Table Entry—Examples

Type	Example
Object	`hosts.org_dir.sales.doc.com.`
Table entry	`'[uid=33555],passwd.org_dir.Eng.doc.com.'`
Two-value table entry	`'[name=sales,gid=2],group.org_dir.doc.com.'`

Columns use a special version of indexed names. Because you can only work on columns with the `nistbladm` command, see "Using the `nistbladm` Command With Tables" on page 346 for more information.

Displaying NIS+ Defaults—The `nisdefaults` Command

The `nisdefaults` command displays the seven default values currently active in the namespace. These default values are either

- Preset values supplied by the NIS+ software
- The defaults specified in the `NIS_DEFAULTS` environment variable (if you have `NIS_DEFAULTS` values set)

Any object that you create on this machine will automatically acquire these default values unless you override them with the `-D` option of the command you are using to create the object.

TABLE 15–12 The Seven NIS+ Default Values and `nisdefaults` Options

Default	Option	From	Description
Domain	`-d`	`/etc/defaultdomain`	Displays the home domain of the machine from which the command was entered.
Group	`-g`	`NIS_GROUP` environment variable	Displays the group that would be assigned to the next object created from this shell.
Host	`-h`	`uname -n`	Displays the machine's host name.
Principal	`-p`	`gethostbyname()`	Displays the fully qualified user name or host name of the NIS+ principal who entered the `nisdefaults` command.
Access Rights	`-r`	`NIS_DEFAULTS` environment variable	Displays the access rights that will be assigned to the next object or entry created from this shell. Format: `----rmcdr---r---`
Search path	`-s`	`NIS_PATH` environment variable	Displays the syntax of the search path, which indicate the domains that NIS+ will search through when looking for information. Displays the value of the `NIS_PATH` environment variable if it is set.
Time-to-live	`-t`	`NIS_DEFAULTS` environment variable	Displays the time-to-live that will be assigned to the next object created from this shell. The default is 12 hours.
All (terse)	`-a`		Displays all seven defaults in terse format.
Verbose	`-v`	Display specified values in verbose mode.	

You can use these options to display all default values or any subset of them:

- To display all values in verbose format, type the `nisdefaults` command without arguments.

  ```
  master% nisdefaults
  Principal Name : topadmin.doc.com.
  Domain Name : doc.com.
  Host Name : rootmaster.doc.com.
  Group Name : salesboss
  Access Rights : ----rmcdr---r---
  Time to live : 12:00:00:00:00
  Search Path : doc.com.
  ```

- To display all values in terse format, add the `-a` option.
- To display a subset of the values, use the appropriate options. The values are displayed in terse mode. For example, to display the rights and search path defaults in terse mode, type the following.

```
rootmaster% nisdefaults -rs
----rmcdr---r---
doc.com.
```

- To display a subset of the values in verbose mode, add the `-v` flag.

Setting Default Security Values

This section describes how to perform tasks related to the `nisdefaults` command, the `NIS_DEFAULTS` environment variable, and the `-D` option. The `NIS_DEFAULTS` environment variable specifies the following default values:

- Owner
- Group
- Access rights
- Time-to-live

The values that you set in the `NIS_DEFAULTS` environment variable are the default values applied to all NIS+ objects that you create using that shell (unless overridden by using the `-D` option with the command that creates the object).

You can specify the default values (owner, group, access rights, and time-to-live) specified with the `NIS_DEFAULTS` environment variable. Once you set the value of `NIS_DEFAULTS`, every object you create from that shell will acquire those defaults, unless you override them by using the `-D` option when you invoke a command.

Displaying the Value of `NIS_DEFAULTS`

You can check the setting of an environment variable by using the `echo` command, as shown below:

```
client% echo $NIS_DEFAULTS
owner=butler:group=gamblers:access=o+rmcd
```

You can also display a general list of the NIS+ defaults active in the namespace by using the `nisdefaults` command as described in "Displaying NIS+ Defaults—The `nisdefaults` Command" on page 273.

Changing Defaults

You can change the default access rights, owner, and group, by changing the value of the `NIS_DEFAULTS` environment variable. Use the environment command that is appropriate for your shell (`setenv` for C-shell or `$NIS_DEFAULTS=, export` for Bourne and Korn shells) with the following arguments:

- `access=`*right*, where *right* are the access rights using the formats described in "Specifying Access Rights in Commands" on page 270.
- `owner=`*name*, where *name* is the user name of the owner.
- `group=`*group*, where *group* is the name of the default group.

You can combine two or more arguments into one line separated by colons:

`-owner=`*principal-name*`:-group=`*group-name*

Table 15–13 shows some examples:

TABLE 15–13 Changing Defaults—Examples

Tasks	Examples
This command grants owner read access as the default access right.	`client% setenv NIS_DEFAULTS access=o+r`
This command sets the default owner to be the user abe whose home domain is doc.com.	`client% setenv NIS_DEFAULTS owner=abe.doc.com.`
This command combines the first two examples on one code line.	`client% setenv NIS_DEFAULTS access=o+r:owner=abe.doc.com.`

All objects and entries created from the shell in which you changed the defaults will have the new values you specified. You cannot specify default settings for a table column or entry; the columns and entries simply inherit the defaults of the table.

Resetting the Value of `NIS_DEFAULTS`

You can reset the `NIS_DEFAULTS` variable to its original values, by typing the name of the variable without arguments, using the format appropriate to your shell:

For C shell

```
client# unsetenv NIS_DEFAULTS
```

For Bourne or Korn shell

```
client$ NIS_DEFAULTS=; export NIS_DEFAULTS
```

Specifying Nondefault Security Values at Creation Time

You can specify different (that is, nondefault) access rights, owner, and group, any time that you create an NIS+ object or table entry with any of the following NIS+ commands:

- `nismkdir`—Creates NIS+ directory objects
- `nisaddent`—Transfers entries into an NIS+ table
- `nistbladm`—Creates entries in an NIS+ table

To specify security values other than the default values, insert the `-D` option into the syntax of those commands, as described in "Specifying Access Rights in Commands" on page 270.

As when setting defaults, you can combine two or more arguments into one line. Remember that column and entry's owner and group are always the same as the table, so you cannot override them.

For example, to use the `nismkdir` command to create a `sales.doc.com` directory and override the default access right by granting the owner only read rights you would type:

```
client% nismkdir -D access=o+r sales.doc.com
```

Changing Object and Entry Access Rights

The `nischmod` command operates on the access rights of an NIS+ object or table entry. It does not operate on the access rights of a table column; for columns, use the `nistbladm` command with the `-D` option. For all `nischmod` operations, you must already have modify rights to the object or entry.

Using `nischmod` to Add Rights

To add rights for an object or entry use:

For object

`nischmod class+`*right object-name*

For table entry

```
nischmod class+right [column-name=value],table-name
```

For example, to add read and modify rights to the group of the `sales.doc.com.` directory object you would type:

```
client% nischmod g+rm sales.doc.com.
```

For example to add read and modify rights to group for the `name=abe` entry in the `hosts.org_dir.doc.com.` table you would type:

```
client% nischmod g+rm '[name=abe],hosts.org_dir.doc.com.'
```

Using `nischmod` to Remove Rights

To remove rights for an object or entry use:

For object

```
nischmod class-right object-name
```

For entry

```
nischmod class-right [column-name=value],table-name
```

For example, to remove create and destroy rights from the group of the `sales.doc.com.` directory object you would type:

```
client% nischmod g-cd sales.doc.com.
```

For example to remove destroy rights from group for the `name=abe` entry in the `hosts.org_dir.doc.com.` table, you would type:

```
client% nischmod g-d '[name=abe],hosts.org_dir.doc.com.'
```

Specifying Column Access Rights

The `nistbladm` command performs a variety of operations on NIS+ tables. Most of these tasks are described in "Using the `nistbladm` Command With Tables" on page 346. However, two of its options, `-c` and `-u`, enable you to perform some security-related tasks:

- The `-c` option. The `-c` option allows you to specify initial column access rights when creating a table with the `nistbladm` command.
- *The* `-u` option. The `-u` option allows you to change column access rights with the `nistbladm` command.

Setting Column Rights When Creating a Table

When a table is created, its columns are assigned the same rights as the table object. These table level, rights are derived from the `NIS_DEFAULTS` environment variable, or are specified as part of the command that creates the table. You can also use the `nistbladm -c` option to specify initial column access rights when creating a table with `nistbladm`. To use this option you must have create rights to the directory in which you will be creating the table. To set column rights when creating a table use:

`nistbladm -c` *type* `'`*columnname*`=`[*flags*] [`,`*access*]... *tablename*'

Where:

- *type* is a character string identifying the kind of table. A table's *type* can be anything you want it to be.
- *columnname* is the name of the column.
- *flags* is the type of column. Valid flags are:
  - `S` for searchable
  - `I` for case insensitive
  - `C` for encrypted
  - `B` for binary data
  - `X` for XDR encoded data
- *access* is the access rights for this column that you specify using the syntax described in "Specifying Access Rights in Commands" on page 270.
- ... indicates that you can specify multiple columns each of the own type and with their own set of rights.
- *tablename* is the fully qualified name of the table you are creating.

To assign a column its own set of rights at table creation time, append access rights to each column's equal sign after the column type and a comma. Separate the columns with a space:

column=type`,`*rights* *column=type*`,`*rights* *column=type*`,`*rights*

The example below creates a table named `depts` in the `doc.com` directory, of type `div`, with three columns (`Name`, `Site`, and `Manager`), and adds modify rights for the group to the second and third columns:

```
rootmaster% nistbladm -c div Name=S Site=S,g+m Manager=S,g+m depts.doc.com.
```

For more information about the `nistbladm` and the `-c` option, see Chapter 19.

Adding Rights to an Existing Table Column

The `nistbladm -u` option allows you to add additional column access rights to an existing table column with the `nistbladm` command. To use this option you must have modify rights to the table column. To add additional column rights use:

```
nistbladm -u [column=access,...],tablename
```

Where:

- *column* is the name of the column.
- *access* is the access rights for this column that you specify using the syntax described in "Specifying Access Rights in Commands" on page 270 .
- ... indicates that you can specify rights for multiple columns.
- *tablename* is the fully qualified name of the table you are creating.

Use one *column=access* pair for each column whose rights you want to update. To update multiple columns, separate them with commas and enclose the entire set with square brackets:

```
[column=access, column=access, column=access]
```

The full syntax of this option is described in Chapter 2 .

The example below adds read and modify rights to the group for the `name` and `addr` columns in the `hosts.org_dir.doc.com.` table.

```
client% nistbladm -u '[name=g+rm,addr=g+rm],hosts.org_dir..doc.com.'
```

Removing Rights to a Table Column

To remove access rights to a column in an NIS+ table, you use the `-u` option as described above in "Adding Rights to an Existing Table Column" on page 279 except that you subtract rights with a minus sign (rather than adding them with a plus sign).

The example below removes group's read and modify rights to the hostname column in the `hosts.org_dir.doc.com.` table.

```
client% nistbladm -u 'name=g-rm,hosts.org_dir.doc.com.'
```

Changing Ownership of Objects and Entries

The `nischown` command changes the owner of one or more objects or entries. To use it, you must have modify rights to the object or entry. The `nischown` command cannot change the owner of a column, since a table's columns belong the table's owner. To change a column's owner, you must change the table's owner.

Changing Object Owner With `nischown`

To change an object's owner, use the following syntax:

```
nischown new-owner object
```

Where:

- *new-owner* is the fully qualified user ID of the object's new owner.
- *object* is the fully qualified name of the object.

Be sure to append the domain name to both the object name and new owner name.

The example below changes the owner of the hosts table in the `doc.com.` domain to the user named lincoln whose home domain is `doc.com.`:

```
client% nischown lincoln.doc.com. hosts.org_dir.doc.com.
```

Changing Table Entry Owner With `nischown`

The syntax for changing a table entry's owner uses an indexed entry to identify the entry, as shown below:

```
nischown new-owner [column=value,...],tablename
```

Where:

- *new-owner* is the fully qualified user ID of the object's new owner.
- *column* is the name of the column whose value will identify the particular entry (row) whose owner is to be changed.
- *value* is the data value that identified the particular entry (row) whose owner is to be changed.
- ... indicates that you can specify ownership changes for multiple entries.
- *tablename* is the fully qualified name of the tables containing the entry whose owner is to be changed.

Be sure to append the domain name to both the new owner name and the table name.

The example below changes the owner of an entry in the hosts table of the `doc.com.` domain to `takeda` whose home domain is `doc.com.` The entry is the one whose value in the name column is `virginia`.

```
client% nischown takeda.doc.com. '[name=virginia],hosts.org_dir.doc.com.'
```

Changing an Object or Entry's Group

The `nischgrp` command changes the group of one or more objects or table entries. To use it, you must have modify rights to the object or entry. The `nischgrp` command cannot change the group of a column, since the group assigned to a table's columns is the same as the group assigned to the table. To change a column's group owner, you must change the table's group owner.

Changing an Object's Group With `nischgrp`

To change an object's group, use the following syntax:

```
nischgrp group object
```

Where:

- *group* is the fully qualified name of the object's new group.
- *object* is the fully qualified name of the object.

Be sure to append the domain name to both the object name and new group name.

The example below changes the group of the hosts table in the `doc.com.` domain to `admins.doc.com.`:

```
client% nischgrp admins.doc.com. hosts.org_dir.doc.com.
```

Changing a Table Entry's Group With `nischgrp`

The syntax for changing a table entry's group uses an indexed entry to identify the entry, as shown below (this syntax is fully described in "Syntax for Objects and Table Entries" on page 272).

```
nischgrp new-group [column=value,...],tablename
```

Where:

- *new-group* is the fully qualified name of the object's new group.
- *column* is the name of the column whose value will identify the particular entry (row) whose group is to be changed.
- *value* is the data value that identified the particular entry (row) whose group is to be changed.
- *tablename* is the fully qualified name of the tables containing the entry whose group is to be changed.
- ... indicates that you can specify group changes for multiple entries.

Be sure to append the domain name to both the new group name and the table name.

The example below changes the group of an entry in the hosts table of the `doc.com.` domain to `sales.doc.com.` The entry is the one whose value in the host name column is `virginia`.

```
client% nischgrp sales.doc.com. '[name=virginia],hosts.org_dir.doc.com.'
```

CHAPTER 16

Administering Passwords

This chapter describes how to use the `passwd` command from the point of view of an ordinary user (NIS+ principal) and how an NIS+ administrator manages the password system.

Tip – Some NIS+ security tasks can be performed more easily with Solaris Management Console tools if you have them available.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Using Passwords

When logging in to a machine, users must enter both a user name (also known as a *login ID*) and a password. Although login IDs are publicly known, passwords must be kept secret by their owners.

Logging In

Logging in to a system is a two-step process:

1. **Type your login ID at the `Login:` prompt.**

2. **Type your password at the `Password:` prompt.**

   (To maintain password secrecy, your password is not displayed on your screen when you type it.)

   If your login is successful you will see your system's message of the day (if any) and then your command-line prompt, windowing system, or normal application.

The `Login incorrect` Message

The `Login incorrect` message indicates that:

- You have entered the wrong login ID or the wrong password. This is the most common cause of the `Login incorrect` message. Check your spelling and repeat the process. Note that most systems limit to five the number of unsuccessful login tries you can make:
  - If you exceed a number of tries limit, you will get a `Too many failures - try later` message and not be allowed to try again until a designated time span has passed.
  - If you fail to successfully log in within a specified amount of time you will receive a `Too many tries; try again later` message, and not be allowed to try again until a designated time span has passed.
- Another possible cause of the `Login incorrect` message is that an administrator has locked your password and you cannot use it until it is unlocked. If you are sure that you are entering your login ID and password correctly, and you still get a `Login incorrect` message, contact your system administrator.
- Another possible cause of the `Login incorrect` message is that an administrator has expired your password privileges and you cannot use your password until your privileges are restored. If you are sure that you are entering your login ID and password correctly, and you still get a `Login incorrect` message, contact your system administrator.

The `will expire` Message

If you receive a `Your password will expire in` *N* `days` message (where *N* is a number of days), or a `Your password will expire within 24 hours` message, it means that your password will reach its age limit and expire in that number of days (or hours).

In essence, this message is telling you to change your password now. (See "Changing Your Password" on page 287.)

The `Permission denied` Message

After entering your login ID and password, you may get a `Permission denied` message and be returned to the `login:` prompt. This means that your login attempt has failed because an administrator has either locked your password, or terminated your account, or your password privileges have expired. In these situations you cannot log in until an administrator unlocks your password or reactivates your account or privileges. Consult your system administrator.

Changing Your Password

To maintain security, you should change your password regularly. (See "Choosing a Password" on page 288 for password requirements and criteria.)

Note – The `passwd` command now performs all functions previously performed by `nispasswd`. For operations specific to an NIS+ name space, use `passwd -r nisplus`.

Changing your password is a four-step process:

1. **Run the `passwd` command at a system prompt.**
2. **Type your old password at the `Enter login password` (or similar) prompt.**

   Your keystrokes are not shown on your screen.

   - If you receive a `Sorry: less than` *N* `days since the last change` message, it means that your old password has not been in use long enough and you will not be allowed to change it at this time. You are returned to your system prompt. Consult your system administrator to find out the minimum number of days a password must be in use before it can be changed.
   - If you receive a `You may not change this password` message, it means that your network administrator has blocked any change.
3. **Type your new password at the `Enter new password` prompt.**

   Your keystrokes are not shown on your screen.

   At this point the system checks to make sure that your new password meets the requirements:

   - If it does meet the requirements, you are asked to enter it again.
   - If your new password does not meet the system requirements, a message is displayed informing you of the problem. You must then enter a new password that does meet the requirements.

   See "Password Requirements" on page 289 for the requirements a password must meet.
4. **Type your new password again at the `Re-enter new password` prompt.**

   Your keystrokes are not shown on your screen.

   If your second entry of the new password is not identical to your first entry, you are prompted to repeat the process.

Note – When changing root's password, you must always run `chkey -p` immediately after changing the password. (See "Changing Root Keys From Root" on page 246 and "Changing Root Keys From Another Machine" on page 248 for information on using `chkey -p` to change root's keys.) Failure to run `chkey -p` after changing root's password will result in root being unable to properly log in.

If you receive a `Your password has expired` message it means that your password has reached its age limit and expired. In other words, the password has been in use for too long and you must choose a new password at this time. (See "Choosing a Password" on page 288, for criteria that a new password must meet.)

In this case, choosing a new password is a three-step process:

1. **Type your old password at the `Enter login password` (or similar) prompt.**

   Your keystrokes are not shown on your screen.

2. **Type your new password at the `Enter new password` prompt.**

   Your keystrokes are not shown on your screen.

3. **Type your new password again at the `Re-enter new password` prompt.**

   Your keystrokes are not shown on your screen.

Password Change Failures

Some systems limit either the number of failed attempts you can make in changing your password or the total amount of time you can take to make a successful change. (These limits are implemented to prevent someone else from changing your password by guessing your current password.)

If you (or someone posing as you) fails to successfully log in or change your password within the specified number of tries or time limit, you will get a `Too many failures - try later` or `Too many tries: try again later` message. You will not be allowed to make any more attempts until a certain amount of time has passed. (That amount of time is set by your administrator.)

Choosing a Password

Many breaches of computer security involve guessing another user's password. While the `passwd` command enforces some criteria for making sure the password is hard to guess, a clever person can sometimes figure out a password just by knowing something about the user. Thus, a good password is one that is easy for you to remember but hard for someone else to guess. A bad password is one that is so hard for you to remember that you have to write it down (which you are not supposed to do), or that is easy for someone who knows about you to guess.

Password Requirements

A password must meet the following requirements:

- *Length*. By default, a password must have at least six characters. Only the first eight characters are significant. (In other words, you can have a password that is longer than eight characters, but the system only checks the first eight.) Because the minimum length of a password can be changed by a system administrator, it may be different on your system.
- *Characters*. A password must contain at least two letters (either upper- or lower-case) and at least one numeral or symbol such as `@`, `#`, `%`. For example, you can use `dog#food` or `dog2food` as a password, but you cannot use `dogfood`.
- *Not your login ID*. A password cannot be the same as your login ID, nor can it be a rearrangement of the letters and characters of your login ID. (For the purpose of this criteria, upper and lower case letters are considered to be the same.) For example, if your login ID is `Claire2` you cannot have `e2clair` as your password.
- *Different from old password*. Your new password must differ from your old one by at least three characters. (For the purpose of this criterion, upper- and lower-case letters are considered to be the same.) For example, if your current password is `Dog#fooD` you can change it to `dog#Meat` but you cannot change it to `daT#Food`.

Bad Choices for Passwords

Bad choices for passwords include:

- Any password based on your name
- Names of family members or pets
- Car license numbers
- Telephone numbers
- Social Security numbers
- Employee numbers
- Names related to a hobby or interest
- Seasonal themes, such as Santa in December
- Any word that is in a standard dictionary

Good Choices for Passwords

Good choices for passwords include:

- Phrases plus numbers or symbols (`beam#meup`)
- Nonsense words made up of the first letters of every word in a phrase plus a number or symbol (`swotrb7` for SomeWhere Over The RainBow)
- Words with numbers, or symbols substituted for letters (`sn00py` for snoopy)

Administering Passwords

This section describes how to administer passwords in an NIS+ namespace. This section assumes that you have an adequate understanding of the NIS+ security system in general, and in particular of the role that login passwords play in that system (see Chapter 11, for this information).

Note – The `passwd` command now performs all functions previously performed by `nispasswd`. For operations specific to an NIS+ namespace, use `passwd -r nisplus`.

`nsswitch.conf` File Requirements

In order to properly implement the `passwd` command and password aging on your network, the `passwd` entry of the `nsswitch.conf` file on every machine must be correct. This entry determines where the `passwd` command will go for password information and where it will update password information.

Only five `passwd` configurations are permitted:

- `passwd: files`
- `passwd: files nis`
- `passwd: files nisplus`
- `passwd: compat`
- `passwd: compat passwd_compat: nisplus`

Caution – All of the `nsswitch.conf` files on all of your network's machines must use one of the `passwd` configurations shown above. If you configure the `passwd` entry in any other way, users may not be able to log in.

The `nispasswd` Command

All functions previously performed by the `nispasswd` command are now performed by the `passwd` command. When issuing commands from the command line, you should use `passwd`, not `nispasswd`.

(The `nispasswd` command is still retained with all of its functionality for the purpose of backward compatibility.)

The `yppasswd` Command

All functions previously performed by the `yppasswd` command are now performed by the `passwd` command. When issuing commands from the command line, you should use `passwd`, not `yppasswd`.

(The `yppasswd` is still retained with all of its functionality for the purpose of backward compatibility.)

The `passwd` Command

The `passwd` command performs various operations regarding passwords. The `passwd` command replaces the `nispasswd` command. You should use the `passwd` command for all activities which used to be performed with the `nispasswd` command. (See the `passwd` command man page for a complete description of all `passwd` flags, options, and arguments.)

The `passwd` command allows users to perform the following operations:

- Change their passwords
- List their password information

Administrators can use the `passwd` command to perform the following operations:

- Force users to change their passwords the next time the log in
- Lock a user's password (prevent it from being used)
- Set a minimum number of days before a user can change passwords
- Specified when a user is warned to change passwords
- Set a maximum number of days a password can be used without being changed

`passwd` and the `nsswitch.conf` File

The name service switch determines where the `passwd` command (and other commands) obtains and stores password information. If the `passwd` entry of the applicable `nsswitch.conf` file points to:

- `nisplus`. Password information will be obtained, modified, and stored in the `passwd` and `cred` tables of the appropriate domain.
- `nis`. Password information will be obtained, modified, and stored in `passwd` maps.
- `files`. Password information will be obtained, modified, and stored in the `/etc/passwd` and `/etc/shadow` files.

The `passwd -r` Option

When you run the `passwd` command with the `-r nisplus`, `-r nis`, or `-r files` arguments, those options override the `nsswitch.conf` file setting. You will be warned that this is the case. If you continue, the `-r` option will cause the `passwd` command to ignore the `nsswitch.conf` file sequence and update the information in the password information storage location pointed to by the `-r` flag.

For example, if the `passwd` entry in the applicable `nsswitch.conf` file reads:

```
passwd: files nisplus
```

`files` is the first (primary) source, and `passwd` run without the `-r` option will get its password information from the `/etc/passwd` file. If you run the command with the `-r nisplus` option, `passwd` will get its information from the appropriate NIS+ passwd table and make its changes to that table, not to the `/etc/passwd` file.

The `-r` option should only be used when you cannot use the `nsswitch.conf` file because the search sequence is wrong. For example, when you need to update password information that is stored in two places, you can use the order specified in the `nsswitch.conf` file for the first one, but for the second one you have to force the use of the secondary or tertiary source.

The message:

```
Your specified repository is not defined in the nsswitch file!
```

indicates that your change will be made to the password information in the repository specified by the `-r` option, but that change will not affect anyone until the `nsswitch.conf` file is changed to point to that repository. For example, suppose the `nsswitch.conf` file reads `passwd: files nis` and you use the `-r nisplus` option to establish password-aging limits in an NIS+ passwd table. Those password-aging rules will sit in that table unused because the `nsswitch.conf` file is directing everyone to other places for their password information.

The `passwd` Command and "NIS+ Environment"

In this chapter, the phrase *NIS+ environment* refers to situations where the passwd entry of the applicable `nsswitch.conf` file is set to `nisplus`, or the `passwd` command is run with the `-r nisplus` argument.

The `passwd` Command and Credentials

When run in an NIS+ environment (see above), the `passwd` command is designed to function with or without credentials. Users without credentials are limited to changing their own password. Other password operations can only be performed by users who have credentials (are authenticated) and who have the necessary access rights (are authorized).

The `passwd` Command and Permissions

In this discussion of authorization and permissions, it is assumed that everyone referred to has the proper credentials.

By default, in a normal NIS+ environment the owner of the passwd table can change password information at any time and without constraints. In other words, the owner of the passwd table is normally granted full read, modify, create, and destroy authorization (permission) for that table. An owner can also:

- Assign table ownership to someone else with the `nischown` command.
- Grant some or all of read, modify, create, and destroy rights to the table's group, or even to the world or nobody class. (Of course, granting such rights to world or nobody seriously weakens NIS+ security.)
- Change the permissions granted to any class with the `nisdefaults`, `nischmod`, or `nistbladm` commands.

Note – Regardless of what permissions they have, everyone in the world, and nobody classes are forced to comply with password-aging constraints. In other words, they cannot change a password for themselves or anyone else unless that password has aged past its minimum. Nor can members of the group, world, and nobody classes avoid having to change their own passwords when the age limit has been reached. However, age constraints do not apply to the owner of the passwd table.

To use the `passwd` command in an NIS+ environment, you must have the required authorization (access rights) for the operation:

TABLE 16–1 Access Rights for `passwd` Command

This Operation	Requires These Rights	To This Object
Displaying information	read	passwd table entry
Changing Information	modify	passwd table entry
Adding New Information	modify	passwd table

The `passwd` Command and Keys

If you use `passwd` in an NIS+ environment to change a principal's password, it tries to update the principal's private (secret) key in the cred table.

- If you have modify rights to the DES entry in the cred table and if the principal's login and Secure RPC passwords are the same, `passwd` will update the private key in the cred table.
- If you do not have modify rights to the DES entry in the cred table or if the principal's login and Secure RPC passwords are not the same, the `passwd` command will change the password, but not change the private key.

If you do not have modify rights to the DES entry, it means that the private key in the cred table will have been formed with a password that is now different from the one stored in the passwd table. In this case, the user will have to change keys with the `chkey` command or run `keylogin` after each login.

The `passwd` Command and Other Domains

To operate on the passwd table of another domain, use:

```
passwd [options] -D domainname
```

The `nistbladm` Command

The `nistbladm` command allows you to create, change, and display information about any NIS+ table, including the `passwd` table.

Caution – To perform password operations using the `nistbladm` command you must apply `nistbladm` to the shadow column of the passwd table. Applying `nistbladm` to the shadow column is complex and tricky. Therefore, you should not use the `nistbladm` command for any operation that can more easily be performed by the `passwd` command or by using the Solaris Management Console tools.

You should use the `passwd` command or Solaris Management Console tools to perform the following operations:

- Changing a password
- Setting the maximum period that a password can be used (password aging)
- Setting the minimum period that a password must be used
- Setting the password warning period
- Turning off password aging

It is possible to use the `nistbladm` command to:

- Create new passwd table entries
- Delete an existing entry
- Change the UID and GID fields in the passwd table
- Change access rights and other security-related attributes of the passwd table
- Set expiration and inactivity periods for a user's account (see "Password Privilege Expiration" on page 306 and "Specifying Maximum Number of Inactive Days" on page 307)

`nistbladm` and Shadow Column Fields

You use the `nistbladm` command to set password parameters by specifying the values of the different fields in the shadow column. These fields are entered in the format:

```
                        Min    Warn   Expire
                         |       |       |
nistbladm -m shadow=n1 :n2 :n3 :n4 :n5 :n6 :n7 [name=login], passwd.org_dir
                     |       |       |       |
                Lastchange  Max  Inactive Unused
```

Where:

- *N1 Lastchange*. The date of the last password change expressed as a number of days since January 1, 1970. The value in this field is automatically updated each time the user changes passwords. (See "`nistbladm` and the Number of Days" on page 297 for important information regarding the number of days.) If the field is blank, or contains a zero, it indicates that there has not been any change in the past.

  Note that the number of days in the lastchange field is the base from which other fields and operations are calculated. Thus, an incorrect change in this field could have unintended consequence in regards to minimum, maximum, warning, and inactive time periods.

- *N2 Min*. The minimum number of days that must pass since the last time the password was changed before the user can change passwords again. For example, if the value in the lastchange field is 9201 (that is, 9201 days since 1/1/70) and the value in the min field is 8, the user is unable to change passwords until after day 9209. See "Setting Minimum Password Life" on page 303 for additional information on password minimums.

  Where *min* is one of the following values:

  - *Zero (0)*. A value of zero in this field (or a blank space) means that there is no minimum period.
  - *Greater than zero*. Any number greater than zero sets that number of days as the minimum password life.
  - *Greater than max*. A value in this field that is greater than the value in the max field prevents the user from ever changing passwords. The message: `You may not change this password` is displayed when the user attempts to change passwords.

- *N3 Max*. The maximum number of days that can pass since the last time the password was changed. Once this maximum number of days is exceeded, the user is forced to choose a new password the next time the user logs in. For example, if the value in the lastchange field is 9201 and the value in the max field is 30, after day 9231 (figured 9201+30=9231), the user is forced to choose a new password at the next login. See "Setting a Password Age Limit" on page 302 for additional information on password maximums.

  Where *max* is one of the following values:

  - *Zero (0)*. A value of zero (0) forces the user to change passwords the next time the user logs in, and it then turns off password aging.
  - *Greater than zero*. Any number greater than zero sets that number of days before the password must be changed.

  - *Minus one (-1)*. A value of minus one (-1) turns off password aging. In other words, entering **`passwd -x -1`** *username* cancels any previous password aging applied to that user. A blank space in the field is treated as if it were a minus one.

- *N4 Warn*. The number of days before a password reaches its maximum that the user is warned to change passwords. For example, suppose the value in the lastchange field is 9201, the value in the max field is 30, and the value in the warn field is 5. Then after day 9226 (figured 9201+30-5=9226) the user starts receiving "change your password" type warnings at each longing time. See "Establishing a Warning Period" on page 304 for additional information on password warning times.

  Where *warn* is one of the following values:

  - *Zero (0)*. No warning period.
  - *Greater than zero*. A value of zero (0) sets the warning period to that number of days.

- *N5 Inactive*. The maximum number of days between logins. If this maximum is exceeded, the user is not allowed to log in. For example, if the value of this field is 6, and the user does not log in for six days, on the seventh day the user is no longer allowed to log in. See "Specifying Maximum Number of Inactive Days" on page 307 for additional information on account inactivity.

  Where *inactive* is one of the following values:

  - *Minus one (-1*). A value of minus one (-1) turns off the inactivity feature. The user can be inactive for any number of days without losing login privileges. This is the default.
  - *Greater than zero*. A value greater than zero sets the maximum inactive period to that number of days.

- *N6 Expire*. The date on which a password expires, expressed as a number of days since January 1, 1970. After this date, the user can no longer log in. For example, if this field is set to 9739 (September 1, 1995) on September 2, 1995 GMT, the user will not be able to login and will receive a `Login incorrect` message after each try. See "Password Privilege Expiration" on page 306 for additional information on password expiration.

  Where *expire* is one of the following values:

  - *Minus one (-1)*. A value of minus one (-1) turns off the expiration feature. If a user's password has already expired, changing this value to -1 restores it. If you do not want to set any expiration date, type a `-1` in this field.
  - *Greater than zero*. A value greater than zero sets the expiration date to that number of days since 1/1/70. If you enter today's date or earlier, you immediately deactivate the users password.

- *N7 Unused*. This field is not currently used. Values entered in this field will be ignored.
- *Login* is the user's login ID.

Caution – When using `nistbladm` on the shadow column of the password table, all of the numeric fields must contain appropriate values. You cannot leave a field blank, or enter a zero, as a *no change* placeholder.

For example, to specify that the user amy last changed her password on day 9246 (May 1, 1995), cannot change her password until it has been in use for 7 days, must change her password after 30 days, will be warned to change her password after the 25th day, must not remain inactive more than 15 days, and has an account that will expire on day number 9285, you would type:

```
nistbladm -m shadow=9246:7:30:5:15:9285 [name=amy], passwd.org.dir
```

`nistbladm` and the Number of Days

Most password aging parameters are expressed in number of days. The following principles and rules apply:

- Days are counted from January 1, 1970. That is day zero. January 2, 1970, is day 1.
- NIS+ uses Greenwich mean time (GMT) in figuring and counting days. In other words, the day count changes at midnight GMT.
- When you specify a number of days, you must use a whole number. You cannot use fractions of days.
- When the number of days is used to specify some action, such as locking a password, the change takes effect on the day. For example, if you specify that a user's password privilege expires on day 9125 (January 2, 1995), that is the last day that the user can use the password. On the next day, the user can no longer use the password.

Values are entered in both the `Lastchange` and the `Expire` fields as a number of days since January 1, 1970. For example:

TABLE 16–2 Number of Days Since 1/1/70

Date	Day Number
January 1, 1970	0
January 2, 1970	1
January 2, 1971	365
January 1, 1997	9863

Password Related Commands

The `passwd` and `nistbladm` commands provide capabilities that are similar to those offered by other commands. Table 16–3 summarizes their differences.

TABLE 16–3 Related Commands

Command	Description
`yppasswd`	Is now linked to the `passwd` command. Using `yppasswd` simply invokes the `passwd` command.
`nispasswd`	Is now linked to the `passwd` command. Using `nispasswd` simply invokes the `passwd` command.
`niscat`	Can be used to display the contents of the passwd table.

Displaying Password Information

You can use the `passwd` command to display password information about all users in a domain or about one particular user:

For your password information

```
passwd -s
```

For all users in current domain

```
passwd -s -a
```

For a particular user

```
passwd -s username
```

Only the entries and columns for which you have read permission will be displayed. Entries are displayed with the following format:

- Without password aging: *username status*
- With password aging: *username status mm/dd/yy min max warn expire inactive*

TABLE 16–4 NIS+ Password Display Format

Field	Description	For Further Information
username	The user's login name.	
status	The user's password status. PS indicates the account has a password. LK indicates the password is locked. NP indicates the account has no password.	See "Locking a Password" on page 300.
mm/dd/yy	The date, based on Greenwich mean time, that the user's password was last changed.	
min	The minimum number of days since the last change that must pass before the password can be changed again.	See "Setting Minimum Password Life" on page 303.

TABLE 16–4 NIS+ Password Display Format *(Continued)*

Field	Description	For Further Information
max	The maximum number of days the password can be used without having to change it.	See "Setting a Password Age Limit" on page 302.
warn	The number of days' notice that users are given before their passwords have to be changed.	See "Establishing a Warning Period" on page 304.
expire	A date on which users loose the ability to log in to their accounts.	See "Password Privilege Expiration" on page 306.
inactive	A limit on the number of days that an account can go without being logged in to. Once that limit is passed without a log in users can no longer access their accounts.	See "Specifying Maximum Number of Inactive Days" on page 307.

To display entries from a passwd table in another domain, use the `-D` option:

For all users in another domain

```
passwd -s -a -D domainname
```

For a particular user

```
passwd -s -D domainname username
```

Changing Passwords

New passwords must meet the criteria described in "Password Requirements" on page 289.

Changing Your Own Password

To change your password, type

```
station1% passwd
```

You will be prompted for your old password and then the new password and then the new password a second time to confirm it.

Changing Someone Else's Password

To change another user's password in the same domain, use:

```
passwd username
```

To change another user's password in a different domain, use:

```
passwd -D domainname username
```

When using the passwd command in an NIS+ environment (see "The passwd Command and "NIS+ Environment"" on page 292) to change someone else's password you must have modify rights to that user's entry in the passwd table (this usually means that you are a member of the group for the passwd table and the group has modify rights). You do not have to enter either the user's old password or your password. You will be prompted to enter the new password twice to make sure that they match. If they do not match, you will be prompted to enter them again.

Changing Root's Password

When changing root's password, you must always run `chkey -p` immediately after changing the password with the passwd command. Failure to run `chkey -p` after changing root's password will result in root being unable to properly log in.

To change a root password, follow these steps:

1. **Log in as root.**
2. **Change root's password using `passwd`.**

   Do not use nispasswd.
3. **Run `chkey -p`.**

   You *must* use the `-p` option.

Locking a Password

When operating in an NIS+ environment (see "The passwd Command and "NIS+ Environment"" on page 292), an administrator (a group member) with modify rights to a user's entry in the passwd table can use the passwd command to lock a password. An account with a locked password cannot be used. When a password is locked, the user will receive a `Login incorrect` message after each login attempt.

Keep in mind that locked passwords have no effect on users who are already logged in. A locked password only prevents users from performing those operations that require giving a password such as login, rlogin, ftp, or telnet.

Note also that if a user with a locked password is already logged in, and that user uses the passwd command to change passwords, the lock is broken.

You can use this feature to:

- Temporarily lock a user's password while that user is on vacation or leave. This prevents anyone from logging in as the absent user.
- Immediately lock one or more user passwords in the case of suspected security problem.
- Quickly lock a dismissed employee out of the system. This is quicker and easier than eliminating that user's account and is an easy way of preserving any data stored in that account.

- If you have assigned passwords to UNIX processes, you can lock those passwords. This allows the process to run, but prevents anyone from logging in as those processes even if they know the process password. (In most cases, processes would not be set up as NIS+ principals, but would maintain their password information in `/etc` files. In such a case you would have to run the `passwd` command in files mode to lock `/etc` stored passwords.)

To lock a password, use:

```
passwd -l username
```

Unlocking a Password

To unlock a user's password, you simply change it. You can "change" it back to the exact same password that it was when it was locked. Or you can change it to something new.

For example, to unlock `jody`'s password, you would enter:

```
station1% passwd jody
```

Managing Password Aging

Password aging is a mechanism you can use to force users to periodically change their passwords.

Password aging allows you to:

- Force a user to choose a new password the next time the user logs in. (See "Forcing Users to Change Passwords" on page 302 for details.)
- Specify a maximum number of days that a password can be used before it has to be changed. (See "Setting a Password Age Limit" on page 302 for details.)
- Specify a minimum number of days that a password has to be in existence before it can be changed. (See "Setting Minimum Password Life" on page 303 for details.)
- Specify that a warning message be displayed whenever a user logs in a specified number of days before the user's password time limit is reached. (See "Establishing a Warning Period" on page 304 for details.)
- Specify a maximum number of days that an account can be inactive. If that number of days pass without the user logging in to the account, the user's password will be locked. (See "Specifying Maximum Number of Inactive Days" on page 307 for details.)
- Specify an absolute date after which a user's password cannot be used, thus denying the user the ability to log on to the system. (See "Password Privilege Expiration" on page 306 for details.)

Keep in mind that users who are already logged in when the various maximums or dates are reached are not affected by the above features. They can continue to work as normal.

Password aging limitations and activities are only activated when a user logs in or performs one of the following operations:

- `login`
- `rlogin`
- `telnet`
- `ftp`

These password aging parameters are applied on user-by-user basis. You can have different password aging requirements for different users. (You can also set general default password aging parameters as described in "Managing Password Aging" on page 301.)

Forcing Users to Change Passwords

There are two ways to force a user to change passwords the next time the user logs in:

Force change keeping password aging rules in effect

`passwd -f` *username*

Force change and turn off password aging rules

`passwd -x 0` *username*

Setting a Password Age Limit

The `-max` argument to the `passwd` command sets an age limit for the current password. In other words, it specifies the number of days that a password remains valid. After that number of days, a new password must be chosen by the user. Once the maximum number of days have passed, the next time the user tries to login with the old password a `Your password has been expired for too long` message is displayed and the user is forced to choose a new password in order to finish logging in to the system.

The max argument uses the following format:

`passwd -x` *max username*

Where:

- *username* is the login ID of the user
- *max* is one of the following values:
  - *Greater than zero*. Any number greater than zero sets that number of days before the password must be changed.
  - *Zero (0)*. A value of zero (0) forces the user to change passwords the next time the user logs in, and it then turns off password aging.

- *Minus one (-1)*. A value of minus one (-1) turns off password aging. In other words, entering **`passwd -x -1`** *username* cancels any previous password aging applied to that user.

For example, to force the user `schweik` to change passwords every 45 days, you would type the command:

```
station1% passwd -x 45 schweik
```

Setting Minimum Password Life

The *min* argument to the `passwd` command specifies the number of days that must pass before a user can change passwords. If a user tries to change passwords before the minimum number of days has passed, a `Sorry less than N days since the last change` message is displayed.

The *min* argument uses the following format:

```
passwd -x max -n min username
```

Where:

- *username* is the login ID of the user
- *max* is the maximum number of days a password is valid as described in the section above
- *min* is the minimum number of days that must pass before the password can be changed

For example, to force the user `eponine` to change passwords every 45 days, and prevent him from changing it for the first 7 days you would type the command:

```
station1% passwd -x 45 -n 7 eponine
```

The following rules apply to the *min* argument:

- You do not have to use a *min* argument or specify a minimum number of days before a password can be changed.
- If you do use the *min* argument, it must always be used in conjunction with the `-max` argument. In other words, in order to set a minimum value you must also set a maximum value.
- If you set *min* to be greater than *max*, the user is unable to change passwords at all. For example, the command `passwd -x 7 -n 8` prevents the user from changing passwords. If the user tries to change passwords, the `You may not change this password` message is displayed. Setting the *min* value greater than the *max* value has two effects:
  - The user is unable to change password. In this case, only someone with administer privileges could change the password. For example, in situations where multiple users share a common group password, setting the *min* value for that password greater than the *max* value would prevent any individual

user from changing the group password.

- The password is only valid for the length of time set by the *max* value, but the user cannot change it because the *min* value is greater than the *max* value. Thus, there is no way for the user to prevent the password from becoming invalid at the expiration of the *max* time period. In effect, this prevents the user from logging in after the *max* time period unless an administrator intervenes.

Establishing a Warning Period

The *warn* argument to the `passwd` command specifies the number of days before a password reaches its age limit that users will start to seeing a `Your password will expire in N days` message (where *N* is the number of days) when they log in.

For example, if a user's password has a maximum life of 30 days (set with the `-max` argument) and the warn value is set to 7 days, when the user logs in on the 24th day (one day past the warn value) the warning message `Your password will expire in 7 days` is displayed. When the user logs in on the 25th day the warning message `Your password will expire in 6 days` is displayed.

Keep in mind that the warning message is not sent by Email or displayed in a user's console window. It is displayed only when the user logs in. If the user does not log in during this period, no warning message is given.

Keep in mind that the *warn* value is *relative* to the *max* value. In other words, it is figured backwards from the deadline set by the *max* value. Thus, if the *warn* value is set to 14 days, the `Your password will expire in N days` message will begin to be displayed two weeks before the password reaches its age limit and must be changed.

Because the *warn* value is figured relative to the *max* value, it only works if a *max* value is in place. If there is no *max* value, *warn* values are meaningless and are ignored by the system.

The *warn* argument uses the following format:

```
passwd -x max -w warn username
```

Where:

- *username* is the login ID of the user.
- *max* is the maximum number of days a password is valid as described on "Setting a Password Age Limit" on page 302.
- *warn* is the number of days before the password reaches its age limit that the warning message will begin to be displayed.

For example, to force the user nilovna to change passwords every 45 days, and display a warning message 5 days before the password reaches its age limit you would type the command:

```
station1% passwd -x 45 -w 5 nilovna
```

The following rules apply to the *warn* argument:

- You do not have to use the *warn* argument or specify a warning message. If no *warn* value is set, no warning message is displayed prior to a password reaching its age limit.
- If you do use the *warn* argument, it must always be used in conjunction with the *max* argument. In other words, in order to set a warning value you must also set a maximum value.

Note – You can also use Solaris Management Console to set a warn value for a user's password.

Turning Off Password Aging

There are two ways to turn off password aging for a given user:

Turn off aging while allowing user to retain current password

`passwd -x -1` *username*

Force user to change password at next login, and then turn off aging

`passwd -x 0` *username*

This sets the *max* value to either zero or -1 (see "Setting a Password Age Limit" on page 302 for more information on this value).

For example, to force the user `mendez` to change passwords the next time he logs in and then turn off password aging you would type the command:

```
station% passwd -x 0 mendez
```

Note – You can also use Solaris Management Console to set this parameter for a user's password.

You can also use the `nistbladm` command to set this value. For example, to turn off password aging for the user otsu and allow her to continue using her current password, you would type:

```
station1% nistbladm -m 'shadow=0:0:-1:0:0:0:0' [name=otsu],passwd.org_dir
```

For additional information on using the `nistbladm` command, see "The `nistbladm` Command" on page 294.

Password Privilege Expiration

You can set a specific date on which a user's password privileges expires. When a user's password privilege expires, that user can no longer have a valid password at all. In effect, this locks the user out of the system after the given date because after that date the user can no longer log in.

For example, if you specify an expire date of December 31, 1997, for a user named pete, on January 1, 1998 he will not be able to log in under that user ID regardless of what password he uses. After each login attempt he will receive a `Login incorrect` message.

Password Aging Versus Expiration

Expiration of a user's password privilege is not the same as password aging.

- *Password aging*. A password that has not been changed for longer than the aging time limit is sometimes referred to as an *expired password*. But that password can still be used to log in *one* more time. As part of that last login process the user is forced to choose a new password.
- *Expiration of password privilege*. When a user's password *privilege* expires, the user cannot log in at all with *any* password. In other words, it is the user's permission to log in to the network that has expired.

Setting an Expiration Date

Password privilege expiration dates only take effect when the user logs in. If a user is *already* logged in, the expiration date has no affect until the user logs out or tries to use `rlogin` or `telnet` to connect to another machine at which time the user will not be able to log in again. Thus, if you are going to implement password privilege expiration dates, you should require your users to log out at the end of each day's work session.

Note – If you have Solaris Management Console tools available, do not use `nistbladm` to set an expiration date. Use Solaris Management Console tools because they are easier to use and provide less chance for error.

To set an expiration date with the `nistbladm` command:

```
nistbladm -m 'shadow=n:n:n:n:n:n6:n' [name=login],passwd.org_dir
```

Where:

- *login* is the user's login ID
- *n* indicates the values in the other fields of the shadow column
- *n6* is the date on which the user's password privilege expires This date is entered as a number of days since January 1, 1970 (see Table 16–2). *n6* can be one of the following values:

- *Minus one (-1)*. A value of minus one (-1) turns off the expiration feature. If a user's password has already expired, changing this value to -1 restores (un-expires) it. If you do not want to set any expiration date, type `-1` in this field.
- *Greater than zero*. A value greater than zero sets the expiration date to that number of days since 1/1/70. If you enter today's date or earlier, you immediately expire the user's password.

For example, to specify an expiration date for the user `pete` of December 31, 1995 you would type:

```
station1% nistbladm -m 'shadow=n:n:n:n:n:9493:n' [name=pete],passwd.org_dir
```

Caution – All of the fields must be filled in with valid values.

Turning Off Password Privilege Expiration

To turn off or deactivate password privilege expiration, you must use the `nistbladm` command to place a `-1` in this field. For example, to turn off privilege expiration for the user huck, you would type:

```
station1% nistbladm -m 'shadow=n:n:n:n:n:-1:n' [name=huck],passwd.org_dir
```

Or you can use the `nistbladm` command reset the expiration date to some day in the future by entering a new number of days in the *n6* field.

Specifying Maximum Number of Inactive Days

You can set a maximum number of days that a user can go without logging in on a given machine. Once that number of days passes without the user logging in, that machine will no longer allow that user to log in. In this situation, the user will receive a `Login incorrect` message after each login attempt.

This feature is tracked on a machine-by-machine basis, not a network-wide basis. That is, in an NIS+ environment, you specify the number of days a user can go without logging in by placing an entry for that user in the passwd table of the user's home domain. That number applies for that user on all machines on the network.

For example, suppose you specify a maximum inactivity period of 10 days for the user sam. On January 1, sam logs in to both machine-A and machine-B, and then logs off both machines. Four days later on January 4, sam logs in on machine-B and then logs out. Nine days after that on January 13, sam can still log in to machine-B because only 9 days have elapsed since the last time he logged in on that machine, but he can no longer log in to machine-A because thirteen days have passed since his last log in on that machine.

Keep in mind that an inactivity maximum cannot apply to a machine the user has never logged in to. No matter what inactivity maximum has been specified or how long it has been since the user has logged in to some other machine, the user can always log in to a machine that the user has never logged in to before.

Caution – Do not set inactivity maximums unless your users are instructed to log out at the end of each workday. The inactivity feature only relates to logins; it does not check for any other type of system use. If a user logs in and then leaves the system up and running at the end of each day, that user will soon pass the inactivity maximum because there has been no login for many days. When that user finally does reboot or log out, he or she won't be able to log in.

Note – If you have Solaris Management Console tools available, do not use `nistbladm` to set an inactivity maximum. Use Solaris Management Console tools because they are easier to use and provide less chance for error.

To set a login inactivity maximum, you must use the `nistbladm` command in the format:

```
nistbladm -m 'shadow=n:n:n:n:n5:n:n' [name=login],passwd.org_dir
```

Where:

- *login* is the user's login ID
- *n* indicates the values in the other fields of the shadow column
- *n5* is the number of days the user is allowed to go between logins. *Inactive* can be one of the following values:
  - *Minus one (-1)*. A value of minus one (-1) turns off the inactivity feature. The user can be inactive for any number of days without losing login privileges. This is the default.
  - *Greater than zero*. A value greater than zero sets the maximum inactive period to that number of days.

For example, to specify that the user `sam` must log in at least once every seven days, you would type:

```
station1% nistbladm -m 'shadow=n:n:n:n:7:n:n' [name=sam],passwd.org_dir
```

To clear an inactivity maximum and allow a user who has been prevented from logging in to log in again, use `nistbladm` to set the inactivity value to -1.

Specifying Password Criteria and Defaults

The following subsections describe various password-related defaults and general criteria that you can specify.

The `/etc/defaults/passwd` File

The `/etc/defaults/passwd` file is used to set four general password defaults for users whose `nsswitch.conf` file points to `files`. The defaults set by the `/etc/defaults/passwd` file apply only to those users whose operative password information is taken from `/etc` files; they do not apply to anyone using either NIS maps or NIS+ tables. An `/etc/defaults/passwd` file on an NIS+ server only affects local users who happen to be obtaining their password information from those local files. An `/etc/defaults/passwd` file on an NIS+ server has no effect on the NIS+ environment or users whose `nsswitch.conf` file points to either `nis` or `nisplus`.

The four general password defaults governed by the `/etc/defaults/passwd` file are:

- Maximum number of weeks the password is valid
- Minimum number of weeks the password is valid
- The number of weeks before the password becomes invalid that the user is warned
- The minimum number of characters that a password must contain

The following principles apply to defaults set with an `/etc/defaults/passwd` file:

- For users who obtain password information from local `/etc` files, individual password aging maximums, minimums and warnings set by the `password` command or Solaris Management Console override any `/etc/defaults/passwd` defaults. In other words, defaults set in the `/etc/defaults/passwd` file are not only applied to those users who do not have corresponding individual settings in their entries in their `passwd` table.
- Except for password length, all the `/etc/defaults/passwd` file defaults are expressed as a number of weeks. (Remember that *individual* password aging times are expressed as a number of days.)
- The `MAXWEEKS`, `MINWEEKS`, and `WARNWEEKS` defaults are all counted forward from the date of the user's last password change. (Remember that *individual warn* values are counted backwards from the maximum date.)

By default, `/etc/defaults/passwd` files already contain the entries:

```
MAXWEEKS=
MINWEEKS=
PASSLENGTH=
```

To implement an entry, simply type the appropriate number after the equal sign. Entries that do not have a number after the equal sign are inactive and have no affect on any user. Thus, to set a `MAXWEEKS` default of `4`, you would change the `/etc/defaults/passwd` file to read:

```
MAXWEEKS=4
MINWEEKS=
PASSLENGTH=
```

Maximum weeks

You can use the `MAXWEEKS` default in the `/etc/defaults/passwd` file to set the maximum number of weeks that a user's password is valid. To set a default maximum time period, type the appropriate number of weeks after the equal sign in the `MAXWEEKS=` line:

```
MAXWEEKS=N
```

Where *N* is a number of weeks. For example, `MAXWEEKS=9`.

Minimum Weeks

You can use the `MINWEEKS` default in the `/etc/defaults/passwd` file to set the minimum nuber of weeks that must pass before a user can change passwords. To set a default minimum time period, type the appropriate number of weeks after the equal sign on the `MINWEEKS=` line:

```
MINWEEKS=N
```

Where *N* is a number of weeks. For example, `MINWEEKS=2`.

Warning Weeks

You can add a `WARNWEEKS` default to the `/etc/defaults/passwd` file set the number of weeks prior to a password becoming invalid due to aging that user is warned. for example, if you have set the `MAXWEEKS` default to `9`, and you want users to be warned two weeks before their passwords become invalid, you would set the `MAXWEEKS` default to `7`.

There is no point in setting the `WARNWEEKS` default unless you also set a `MAXWEEKS` default.

Remember that `WARNWEEKS` are counted forward from the date of the user's last password change, not backwards from the `MAXWEEKS` expiration date. Thus, `WARNWEEKS` must always be less than `MAXWEEKS` and cannot be equal to or greater than `MAXWEEKS`.

A `WARNWEEKS` default will not work unless there is also a `MAXWEEKS` default.

To set the warning time period, type the appropriate number of weeks after the equal sign on the `WARNWEEKS=` line.

```
WARNWEEKS=N
```

Where *N*is the number of weeks. For example, `WARNWEEKS=1`.

Minimum Password Length

By default, the `passwd` command assumes a minimum length of six characters. You can use the `PASSLENGTH` default in the `/etc/defaults/passwd` files to change that by setting the minimum number of characters that a user's password must contain to some other number.

To set the minimum number of characters to something other than six, type the appropriate number of characters after the equal sign in the `PASSLENGTH=` line:

```
PASSLENGTH=N
```

Where *N*is the number of characters. For example, `PASSLENGTH=7`.

Password Failure Limits

You can specify a number-of-tries limit or an amount-of-time limit (or both) for a user's attempt to change passwords. These limits are specified by adding arguments when starting the `rpc.nispasswdd` daemon.

Limiting the number of attempts or setting a time frame provides a limited (but not foolproof) defense against unauthorized persons attempting to change a valid password to one that they discover through trial and error.

Maximum Number of Tries

To set the maximum number of times a user can try to change a password without succeeding, use the `-a` *number* argument with `rpc.nispasswdd`, where *number* is the number of allowed tries. (You must have superuser privileges on the NIS+ master server to run `rpc.nispasswdd`.)

For example, to limit users to no more than four attempts (the default is 3), you would type:

```
station1# rpc.nispasswdd -a 4
```

In this case, if a user's fourth attempt at logging in is unsuccessful, the message `Too many failures - try later` is displayed. No further attempts are permitted for that user ID until a specified period of time has passed.

Maximum Login Time Period

To set the maximum amount a time a user can take to successfully change a password, use the `-c` *minutes* argument with `rpc.nispasswdd,` where *minutes* is the number of minutes a user has to log in. (You must have superuser privileges on the NIS+ master server to run `rpc.nispasswdd`.)

For example, to specify that users must successfully log in within 2 minutes, you would type:

```
station1# rpc.nispasswdd -c 2
```

In this case, if a user is unable to successfully change a password within 2 minutes, the message is displayed at the end of the two-minute period. No further attempts are permitted for that user ID until a specified period of time has passed.

CHAPTER 17

Administering NIS+ Groups

This chapter describes NIS+ groups and how to administer them.

Tip – Some NIS+ security group tasks can be performed more easily with Solaris Management Console tools if you have them available.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Solaris Groups

In a Solaris-NIS+ environment, there are three kinds of groups: UNIX groups, net groups, and NIS+ groups.

- *UNIX groups*. A UNIX group is simply a collection of users who are given additional UNIX access permissions. In an NIS+ namespace, UNIX group information is stored in the group table located in the `org_dir` directory object (`group.org_dir`). See Chapter 19 for information on how to add, modify, or delete members of a UNIX group.
- *Net groups*. A net group is a group of machines and users that have permission to perform remote operations on other machines. In an NIS+ namespace, net groups information is stored in the netgroup table located in the `org_dir` directory object (`netgroup.org_dir`). See Chapter 19 for information on how to add, modify, or delete members of a net groups.

- *NIS+ groups*. An NIS+ group is a set of NIS+ users that are assigned specific access rights to NIS+ objects, usually for the purpose of administering the namespace. NIS+ group information is stored in tables located in the groups_dir directory object.

NIS+ Groups

NIS+ groups are used to assign access rights to NIS+ objects to one or more NIS+ principles. These access rights are described in Chapter 11. Information about NIS+ groups is stored in tables located in the NIS+ `groups_dir` directory object. Information about each group is stored in a table of the same name. For example, information about the `admin` group is stored in `admin.groups_dir`.

It is recommended practice to create at least one NIS+ group called `admin`. The `admin` NIS+ group is normally used to designate those users who are to have NIS+ access rights. You can name this group anything you want, but the NIS+ manual set assumes that the group with NIS+ administrator privileges is named `admin`. You can also create multiple NIS+ groups with different sets of users and different sets of rights.

Note – Always use the `nisgrpadm` command to work with NIS+ group membership. You can also use the `nisls` and `nischgrp` commands on the group table. Do not use the `nistbladm` command on the group table.

For a complete description of NIS+ group-related commands and their syntax and options, see the NIS+ man pages.

Related Administration Commands

The `nisgrpadm` command performs most group administration tasks but several other commands affect groups as well:

TABLE 17–1 Commands That Affect Groups

Command	Description	See
`nissetup`	Creates, among other things, the directory in which a domain's groups are stored: `groups_dir`.	
`nisls`	Lists the contents of the `groups_dir` directory; in other words, all the groups in a domain. For each named groups there will be a table of that name in `groups_dir`.	"Using the `nisls` Command With Directories" on page 324
`nischgrp`	Changes or assigns a group to any NIS+ object.	"Changing an Object or Entry's Group" on page 282
`niscat`	Lists the object properties and membership of an NIS+ group.	"Using `niscat` With NIS+ Groups" on page 317
`nisdefaults`	Lists, among other things, the group that will be assigned to any new NIS+ object.	"Displaying NIS+ Defaults—The `nisdefaults` Command" on page 273

For a complete description of these commands and their syntax, and options, see the NIS+ man pages.

Note – Do not use the `nistbladm` command to work with the NIS+ groups table.

NIS+ Group Member Types

NIS+ groups can have three types of members: explicit, implicit, and recursive; and three types of nonmembers, also explicit, implicit, and recursive. These member types are used when adding or removing members of a group as described in "The `nisgrpadm` Command" on page 318.

Member Types

- *Explicit*. An individual principal. Identified by principal name. The name does not have to be fully qualified if entered from its default domain.
- *Implicit*. All the NIS+ principals who belong to an NIS+ domain. They are identified by their domain name, preceded by the `*` symbol and a dot. The operation you select applies to all the members in the group.

- *Recursive.* All the NIS+ principals that are members of another NIS+ group. They are identified by their NIS+ group name, preceded by the @ symbol. The operation you select applies to all the members in the group.

NIS+ groups also accept nonmembers in all three categories: explicit, implicit, and recursive. Nonmembers are principals specifically excluded from a group that they otherwise would be part of.

Nonmember Types

Nonmembers are identified by a minus sign in front of their name:

- *Explicit-nonmember.* Identified by a minus sign in front of the principal name.
- *Implicit-nonmember.* Identified by a minus sign, * symbol, and dot in front of the domain name.
- *Recursive nonmember.* Identified by a minus sign and @ symbol in front of the group name.

Group Syntax

The order in which inclusions and exclusions are entered does not matter. Exclusions always take precedence over inclusions. Thus, if a principal is a member of an included implicit domain and *also* a member of an excluded recursive group, then that principal is not included.

Thus, when using the `nisgrpadm` command, you can specify group members and nonmembers as shown in Table 17–2:

TABLE 17–2 Specifying Group Members and Nonmembers

Type of member	Syntax
Explicit member	*username.domain*
Implicit member	`*.`*domain*
Recursive member	@*groupname.domain*
Explicit nonmember	`-`*username.domain*
Implicit nonmember	`-*.`*domain*
Recursive nonmember	@*groupname.domain*

Using `niscat` With NIS+ Groups

The `niscat -o`command can be used to list the object properties and membership of an NIS+ group.

Listing the Object Properties of a Group

To list the object properties of a group, you must have read access to the `groups_dir` directory in which the group is stored. Use `niscat -o` and the group's fully qualified name, which must include its `groups_dir` subdirectory:

```
niscat -o group-name.groups_dir.domain-name
```

For example:

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

Note – A better list of members is provided by the `nisgrpadm -l` command.

Several of the group's properties are inherited from the `NIS_DEFAULTS` environment variable, unless they were overridden when the group was created. The group flags field is currently unused. In the list of group members, the * symbol identifies member domains and the @ symbol identifies member groups.

The `nisgrpadm` Command

The `nisgrpadm` command creates, deletes, and performs miscellaneous administration operations on NIS+ groups. To use `nisgrpadm`, you must have access rights appropriate for the operation.

TABLE 17–3 Rights Required for nisgrpadm Command

This Operation	Requires This Access Right	To This Object
Create a group	Create	`groups_dir` directory
Destroy a group	Destroy	`groups_dir` directory
List the Members	Read	the group object
Add Members	Modify	the group object
Remove Members	Modify	the group object

The nisgrpadm has two main forms, one for working with groups and one for working with group members.

To create or delete a group, or to lists its members use these forms:

```
nisgrpadm -c group-name.domain-name
nisgrpadm -d group-name
nisgrpadm -l group-name
```

To add or remove members, or determine if they belong to the group use this form (where *member...* can be any combination of the six membership types listed in Table 17–2):

```
nisgrpadm -a group-name member...
nisgrpadm -r group-name member...
nisgrpadm -t group-name member...
```

All operations except create (`-c`) accept a partially qualified *group-name*. However, even for the `-c` option, `nisgrpadm` does not require the use of `groups_dir` in the *group-name* argument. In fact, it won't accept it.

Creating an NIS+ Group

To create an NIS+ group, you must have create rights to the `groups_dir` directory of the group's domain. Use the `-c` option and a fully qualified group name:

```
nisgrpadm -c group-name.
domainname
```

When you create a group, an NIS+ groups table with the name you have given is created in groups_dir. You can use nisls to confirm that the new group table now exists in groups_dir, and niscat to list the groups members listed in the table.

A newly created group contains no members. See "Adding Members to an NIS+ Group" on page 320 for information on how to specify who belongs to a group.

The example below creates three groups named admin. The first is in the doc.com. domain, the second in sales.doc.com., and the third in manf.doc.com. All three are created on the master server of their respective domains.

```
rootmaster# nisgrpadm -c admin.doc.com.
Group admin.doc.com. created.
salesmaster# nisgrpadm -c admin.sales.doc.com.
Group admin.sales.doc.com. created.
manfmaster# nisgrpadm -c admin.manf.doc.com.
Group admin.manf.doc.com. created.
```

The group you create will inherit all the object properties specified in the NIS_DEFAULTS variable; that is, its owner, owning group, access rights, time-to-live, and search path. You can view these defaults by using the nisdefaults command (described in Chapter 15). Used without options, it provides this output:

```
rootmaster# nisdefaults
Principal Name : rootmaster.doc.com.
Domain Name : doc.com.
Host Name : rootmaster.doc.com.
Group Name :
Access Rights : ----rmcdr---r---
Time to live : 12:0:0
Search Path : doc.com.
```

The owner is listed in the Principal Name field. The Group Name of the owning group is listed only if you have set the NIS_GROUP environment variable. For example, assuming a C-shell, to set NIS_GROUP to net_admins.doc.com:

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

You can override any of these defaults at the time you create the group by using the -D option:

```
salesmaster# nisgrpadm -D group=special.sales.doc.com.-c
admin.sales.doc.com. Group admin.sales.doc.com. created.
```

Deleting an NIS+ Group

To delete an NIS+ group, you must have destroy rights to the groups_dir directory in the group's domain. Use the -d option:

```
nisgrpadm -d group-name
```

If the default domain is set properly, you don't have to fully-qualify the group name. However, you should check first (use `nisdefaults`), because you could unintentionally delete a group in another domain. The example below deletes the `test.sales.doc.com.` group.

```
salesmaster% nisgrpadm -d test.sales.doc.com.
Group 'test.sales.doc.com.' destroyed.
```

Adding Members to an NIS+ Group

To add members to an NIS+ group you must have modify rights to the group object. Use the `-a` option:

`nisgrpadm -a` *group-name members. . .*

As described in "NIS+ Group Member Types" on page 315, you can add principals (explicit members), domains (implicit members), and groups (recursive members). You don't have to fully qualify the name of the group or the name of the members who belong to the default domain. This example adds the NIS+ principals panza and valjean, both from the default domain, `sales.doc.com.`, and the principal `makeba`, from the `manf.doc.com.` domain, to the group `top-team.sales.doc.com.`

```
client% nisgrpadm -a Ateam panza valjean makeba.manf.doc.com.
Added panza.sales.doc.com to group Ateam.sales.doc.com
Added valjean.sales.doc.com to group Ateam.sales.doc.com
Added makeba.manf.doc.com to group Ateam.sales.doc.com
```

To verify the operation, use the `nisgrpadm -l` option. Look for the members under the Explicit members heading.

This example adds all the NIS+ principals in the `doc.com.` domain to the `staff.doc.com.` group. It is entered from a client in the `doc.com.` domain. Note the `*` symbol and the dot in front of the domain name.

```
client% nisgrpadm -a Staff *.doc.com.
Added *.doc.com. to group Staff.manf.doc.com.
```

This example adds the NIS+ group `admin.doc.com.` to the `admin.manf.doc.com.` group. It is entered from a client of the `manf.doc.com.` domain. Note the @ symbol in front of the group name.

```
client% nisgrpadm -a admin @admin.doc.com.
Added @admin.doc.com. to group admin.manf.doc.com.
```

Listing the Members of an NIS+ Group

To list the members of an NIS+ group, you must have read rights to the group object. Use the `-l` option:

`nisgrpadm -l` *group-name*

This example lists the members of the admin.manf.doc.com. group. It is entered from a client in the manf.doc.com. group:

```
client% nisgrpadm -l admin
Group entry for admin.manf.doc.com. group:
 No explicit members
 No implicit members:
 Recursive members:
 @admin.doc.com.
 No explicit nonmembers
 No implicit nonmembers
 No recursive nonmembers
```

Removing Members From an NIS+ Group

To remove members from an NIS+ group, you must have modify rights to the group object. Use the -r option:

```
nisgrpadm -r group-name members. . .
```

This example removes the NIS+ principals allende and hugo.manf.doc.com. from the Ateam.sales.doc.com group. It is entered from a client in the sales.doc.com. domain:

```
client% nisgrpadm -r Ateam allende hugo.manf.doc.com.
Removed allende.sales.doc.com. from group Ateam.sales.doc.com.
Removed hugo.manf.doc.com. from group Ateam.sales.doc.com.
```

This example removes the admin.doc.com. group from the admin.manf.doc.com. group. It is entered from a client in the manf.doc.com. domain:

```
client% nisgrpadm -r admin @admin.doc.com.
Removed @admin.doc.com. from group admin.manf.doc.com.
```

Testing for Membership in an NIS+ Group

To find out whether an NIS+ principal is a member of a particular NIS+ group you must have read access to the group object. Use the -t option:

```
nisgrpadm -t group-name members. . .
```

This example tests whether the NIS+ principal topadmin belongs to the admin.doc.com. group. It is entered from a client in the doc.com. domain.

```
client% nisgrpadm -t admin topadmin
topadmin.doc.com. is a member of group admin.doc.com.
```

This example tests whether the NIS+ principal jo, from the sales.doc.com. domain, belongs to the admin.sales.doc.com. group. It is entered from a client in the doc.com. domain.

```
client% nisgrpadm -t admin.sales.doc.com. jo.sales.doc.com.
jo.sales.doc.com. is a member of group admin.sales.doc.com.
```

CHAPTER 18

Administering NIS+ Directories

This chapter describes NIS+ directory objects and how to administer them.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Directories

NIS+ directory objects are used to store information related to an NIS+ domain. For each NIS+ domain, there is a corresponding NIS+ directory structure. See Chapter 2, for more information about NIS+ directories.

For a complete description of NIS+ directory-related commands and their syntax and options, see the NIS+ man pages.

Using the `niscat` Command With Directories

The `niscat -o` command can be used to list the object properties of an NIS+ directory. To use it, you must have read access to the directory object itself.

Listing the Object Properties of a Directory

To list the object properties of a directory, use `niscat -o` and the directory's name:

```
niscat -o directory-name
```

For example:

```
rootmaster# niscat -o doc.com.
Object Name : doc
Owner : rootmaster.doc.com.
Group :
Domain : Com.
Access Rights : r---rmcdr---r---
Time to Live : 24:0:0
Object Type : DIRECTORY
.
.
```

Using the `nisls` Command With Directories

The `nisls` command lists the contents of an NIS+ directory. To use it, you must have read rights to the directory object.

To display in terse format, use:

```
nisls [-dgLmMR] directory-name
```

To display in verbose format, use:

```
nisls -l [-gm] [-dLMR] directory-name
```

TABLE 18–1 Options for the `nisls` Command

Option	Purpose
`-d`	Directory object. Instead of listing a directory's contents, treat it like another object.
`-L`	Links. If the directory name is actually a link, the command follows the link and displays information about the linked directory.
`-M`	Master. Get the information from the master server only. Although this provides the most up-to-date information, it may take longer if the master server is busy.

TABLE 18–1 Options for the `nisls` Command *(Continued)*

Option	Purpose
`-R`	Recursive. List directories recursively. That is, if a directory contains other directories, their contents are displayed as well.
`-l`	Long. Display information in long format. Long format displays an object's type, creation time, owner, and access rights.
`-g`	Group. When displaying information in long format, display the directory's group owner instead of its owner.
`-m`	Modification time. When displaying information in long format, display the directory's modification time instead of its creation time.

Listing the Contents of a Directory—Terse

To list the contents of a directory in the default short format, use one or more of the options listed below and a directory name. If you don't supply a directory name, NIS+ will use the default directory.

`nisls [-dLMR]` *directory-name*

or

```
nisls [-dLMR]
```

For example, this instance of `nisls` is entered from the root master server of the root domain `doc.com.`:

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

Here is another example entered from the root master server:

```
rootmaster% nisls -R sales.doc.com.
sales.doc.com.:
org_dir
groups_dir
groups_dir.sales.doc.com.:
admin
org_dir.sales.doc.com.:
auto_master
auto_home
bootparams
cred
.
```

Listing the Contents of a Directory—Verbose

To list the contents of a directory in the verbose format, use the `-l` option and one or more of the options listed below. The `-g` and `-m` options modify the attributes that are displayed. If you don't supply a directory name, NIS+ will use the default directory.

```
nisls -l [-gm] [-dLMR] directory-name
```

or

```
nisls -l [-gm] [-dLMR]
```

Here is an example, entered from the master server of the root domain `doc.com.`:

```
rootmaster% nisls -l
doc.com.
D r---rmcdr---r--- rootmaster.doc.com. date org_dir
D r---rmcdr---r--- rootmaster.doc.com. date groups_dir
```

The `nismkdir` Command

Note – This section describes how to add a nonroot server to an existing domain using the `nismkdir` command. An easier way to do this is with the `nisserver` script as described in Chapter 4

The `nismkdir` command creates a nonroot NIS+ directory and associates it with a master server. (To create a root directory, use the `nisinit -r` command, described in "The `nisinit` Command" on page 333. The `nismkdir` command can also be used to add a replica to an existing directory.

There are several prerequisites to creating an NIS+ directory, as well as several related tasks.

To create a directory, use:

```
nismkdir [-m master-server] \
 directory-name
```

To add a replica to an existing directory, use:

```
nismkdir -s replica-server \
 directory-name
nismkdir -s replica-server \
 org_dir.directory-name
nismkdir -s replica-server \
```

`groups_dir.`*directory-name*

Creating a Directory

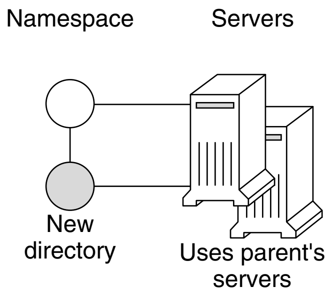
To create a directory, you must have create rights to its parent directory on the domain master server. First use the `-m` option to identify the master server and then the `-s` option to identify the replica, use:

```
nismkdir -m master directory
nismkdir -s replica directory
```

Caution – Always run `nismkdir` on the master server. Never run `nismkdir` on the replica machine. Running `nismkdir` on a replica creates communications problems between the master and the replica.

This example creates the `sales.doc.com.` directory and specifies its master server, `smaster.doc.com.` and its replica, `rep1.doc.com.`. It is entered from the root master server.

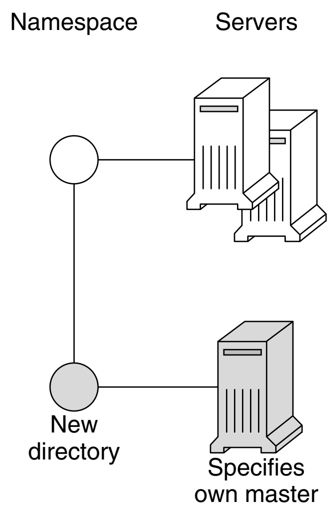
```
rootmaster% nismkdir -m smaster.doc.com. sales.doc.com.
rootmaster% nismkdir -m smaster.doc.com. org_dir.sales.doc.com.
rootmaster% nismkdir -m smaster.doc.com. groups_dir.sales.doc.com.
rootmaster% nismkdir -s rep1.doc.com. sales.doc.com.
rootmaster% nismkdir -s rep1.doc.com. org_dir.sales.doc.com.
rootmaster% nismkdir -s rep1.doc.com. groups_dir.sales.doc.com.
```



The `nismkdir` command allows you to use the parent directory's servers for the new directory instead of specifying its own. However, this should not be done except in the case of small networks. Here are two examples:

The first example creates the `sales.doc.com.` directory and associates it with its parent directory's master and replica servers.

```
rootmaster% nismkdir sales.doc.com
```



The second example creates the `sales.doc.com.` directory and specifies its own master server, `smaster.doc.com.`

```
rootmaster% nismkdir -m smaster.doc.com. sales.doc.com.
```

Since no replica server is specified, the new directory will have only a master server until you use `nismkdir` again to assign it a replica. If the `sales.doc.com.` domain already existed, the `nismkdir` command as shown above would have made `salesmaster.doc.com.` its new master server and would have relegated its old master server to a replica.

Adding a Replica to an Existing Directory

This section describes how to add a replica server to an existing system using the nismkdir command. An easier way to do this is with the `nisserver` script.

Keep in mind the following principles:

- Root domain servers reside in (are part of) the root domain.
- Subdomain servers reside in (are part of) the parent domain immediately above the subdomain in the hierarchy. For example, if a namespace has one root domain named `prime` and a subdomain named `sub1`:
  - The master and replica servers that serve the `prime` domain are themselves part of the `prime` domain because `prime` is the root domain.
  - The master and replica servers that serve the `sub1` subdomain are also part of the `prime` domain because `prime` is the parent of `sub1`.

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

To assign a new replica server to an existing directory, use `nismkdir` on the master server with the `-s` option and the name of the existing directory, `org_dir`, and `groups_dir`:

```
nismkdir -s replica-server existing-directory-name
nismkdir -s replica-server org_dir. existing-directory-name
nismkdir -s replica-server groups_dir. existing-directory-name
```

The `nismkdir` command realizes that the directory already exists, so it does not recreate it. It only assigns it the additional replica. Here is an example with `rep1` being the name of the new replica machine:

```
rootmaster% nismkdir -s rep1.doc.com. doc.com.
rootmaster% nismkdir -s rep1.doc.com. org_dir.doc.com.
rootmaster% nismkdir -s rep1.doc.com. groups_dir.doc.com.
```

Caution – Always run `nismkdir` on the master server. Never run `nismkdir` on the replica machine. Running `nismkdir` on a replica creates communications problems between the master and the replica.

After running the three iterations of `nismkdir` as shown above, you need to run `nisping` from the master server on the three directories:

```
rootmaster# nisping doc.com.
rootmaster# nisping org_dir.doc.com.
rootmaster# nisping group_dir.doc.com.
```

You should see results similar to these:

```
rootmaster# nisping doc.com.
Pinging replicas serving directory doc.com. :
Master server is rootmaster.doc.com.
 Last update occurred at Wed Nov 18 19:54:38 1995
Replica server is rep1.doc.com.
 Last update seen was Wed Nov 18 11:24:32 1995
 Pinging ... rep1.doc.com
```

It is good practice to include `nisping` commands for each of these three directories in the master server's `cron` file so that each directory is "pinged" at least once every 24 hours after being updated.

The nisrmdir Command

The `nisrmdir` command can remove a directory or simply dissociate a replica server from a directory. (When a directory is removed or disassociated from a replica server, that machine no longer functions as an NIS+ replica server for that NIS+ domain.)

When it removes a directory, NIS+ first disassociates the master and replica servers from the directory, and then removes the directory.

- To remove the directory, you must have destroy rights to its parent directory.
- To dissociate a replica server from a directory, you must have modify rights to the directory.

If problems occur, see "Removal or Disassociation of NIS+ Directory from Replica Fails" on page 429.

Removing a Directory

To remove an entire directory and dissociate its master and replica servers, use the `nisrmdir` command without any options:

```
nisrmdir directory-name
nisping domain
```

This example removes the `manf.doc.com.` directory from beneath the `doc.com.` directory:

```
rootmaster% nisrmdir manf.doc.com.
rootmaster% nisping doc.com.
```

Disassociating a Replica From a Directory

To disassociate a replica server from a directory, you must first remove the directory's `org_dir` and `groups_dir` subdirectories. To do this, use the `nisrmdir` command with the `-s` option. After each of the subdirectories are removed, you must run `nisping` on parent domain.

```
nisrmdir -s replicanameorg_dir.domain
nisrmdir -s replicanamegroups_dir.domain
nisrmdir -s replicaname domain
nisping domain
```

This example disassociates the `manfreplica1` server from the `manf.doc.com.` directory:

```
rootmaster% nisrmdir -s manfreplica1 org_dir.manf.doc.com.
rootmaster% nisrmdir -s manfreplica1 groups_dir.manf.doc.com.
```

```
rootmaster% nisrmdir -s manfreplica1 manf.doc.com.
rootmaster% nisping manf.doc.com.
```

If the replica server you are trying to dissociate is down or out of communication, the `nisrmdir -s` command returns a `Cannot remove replica`*name*`: attempt to remove a non-empty table` error message. In such cases, you can run `nisrmdir -f -s` *replicaname* on the master to force the dissociation. Note, however, that if you use `nisrmdir -f -s` to dissociate an out-of-communication replica, you *must* run `nisrmdir -f -s` *again* as soon as the replica is back on line in order to clean up the replica's `/var/nis` file system. If you fail to rerun `nisrmdir -f -s` *replicaname* when the replica is back in service, the old out-of-date information left on the replica could cause problems.

The `nisrm` Command

The `nisrm` command is similar to the standard `rm` system command. It removes any NIS+ object from the namespace, except directories and nonempty tables. To use the `nisrm` command, you must have destroy rights to the object. However, if you don't, you can use the `-f` option, which tries to force the operation in spite of permissions.

You can remove group objects with the `nisgrpadm -d` command (see "Deleting an NIS+ Group" on page 319), and you can empty tables with `nistbladm -r` or `nistbladm -R` (see "Deleting a Table" on page 353).

To remove a nondirectory object, use:

`nisrm [-if]` *object-name*

TABLE 18–2 `nisrm` Syntax Options

Option	Purpose
`-i`	Inquire. Asks for confirmation prior to removing an object. If the *object-name* you provide is not fully qualified, this option is used automatically.
`-f`	Force. Attempts to force a removal even if you don't have the proper permissions. It attempts to change the permission by using the `nischmod` command, and then tries to remove the object again.

Removing Nondirectory Objects

To remove nondirectory objects, use the `nisrm` command and provide the object names:

`nisrm` *object-name*`...`

This example removes a group and a table from the namespace:

```
rootmaster% nisrm -i admins.doc.com. groups.org_dir.doc.com.
Remove admins.doc.com.? y
Remove groups.org_dir.doc.com.? y
```

The `rpc.nisd` Daemon

When you enable the NIS+ service, the `rpc.nisd` daemon starts. You don't need any access rights to start the NIS+ daemon, but you should be aware of all its prerequisites and related tasks. They are described in "Prerequisites to Setting Up a NIS+ Server" on page 107.

Note – The NIS+ service is managed by the Service Management Facility (SMF). Administrative actions on this service, such as enabling, disabling, or restarting, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+. For an overview of the SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

Starting the `rpc.nisd` Daemon

When you enable the NIS+ service, the `rpc.nisd` daemon starts. Use the `svcadm enable` command to start the service.

```
rootmaster# svcadm enable /network/rpc/nisplus:default
```

Stopping the `rpc.nisd` Daemon

When you disable the NIS+ service, the `rpc.nisd` daemon stops. Use the `svcs` command to determine the FMRI and the state of the service. Then use the `svcadm disable` command to stop the service, whether the service is running in normal or NIS-compatibility mode.

```
rootmaster# svcs \*nisplus\*
STATE          STIME    FMRI
online         Oct_07   svc:/network/rpc/nisplus:default

rootmaster# svcadm disable /network/rpc/nisplus:default
```

Changing `rpc.nisd` Syntax Options

By default, the NIS+ daemon starts with security level 2 and runs in normal mode. If you want to include specific options when you invoke the `rpc.nisd` daemon with the Service Management Facility, modify the `/lib/svc/method/nisplus` file to include the desired options. See "NIS+ and the Service Management Facility" on page 85 for more information.

Descriptions of some common `rpc.nisd` syntax options are included in Table 18–3.

TABLE 18–3 Common `rpc.nisd` Syntax Options

Option	Purpose
`-Y`	Specifies that the daemon run in NIS-compatibility mode, which enables it to answer requests from NIS clients. You can start the NIS+ daemon in NIS-compatibility mode in any server, including the root master.
`-B`	Adds DNS forwarding capabilities to an NIS+ daemon running in NIS-compatibility mode. This option requires that the /etc/resolv.conf file be set up for communication with a DNS nameserver.
`-S` *security-level*	Specifies a security level, where `0` means no NIS+ security and `2` provides full NIS+ security. (Level 1 is not supported.)
`-F`	Forces a checkpoint of the directory served by the daemon. This has the side effect of emptying the directory's transaction log and freeing disk space.

The `nisinit` Command

This section describes how to initialize a machine client using the `nisinit` command. An easier way to do this is with the `nisclient` script as described in "Setting Up NIS+ Client Machines" on page 102.

The `nisinit` command initializes a machine to be an NIS+ client or server. As with the `rpc.nisd` command, you don't need any access rights to use the `nisinit` command, but you should be aware of its prerequisites and related tasks. These are described in "Initializing an NIS+ Client" on page 150.

Initializing a Client

You can initialize a client in three different ways:

- By host name
- By broadcast

- By cold-start file

Each way has different prerequisites and associated tasks. For instance, before you can initialize a client by host name, the client's `/etc/hosts` or `/etc/inet/ipnodes` file must list the host name you will use and `nsswitch.conf` file must have `files` as the first choice on the `hosts` line. For IPv6 addresses, specify `ipnodes` as the first choice on the `hosts` line. Following is a summary of the steps that use the `nisinit` command.

To initialize a client by host name, use the `-c` and `-H` options, and include the name of the server from which the client will obtain its cold-start file:

```
nisinit -c -H hostname
```

To initialize a client by cold-start file, use the `-c` and `-C` options, and provide the name of the cold-start file:

```
nisinit -c -C filename
```

To initialize a client by broadcast, use the `-c` and `-B` options:

```
nisinit -c -B
```

Initializing the Root Master Server

To initialize the root master server, use the `nisinit -r`command:

```
nisinit -r
```

You will need the following information

- The superuser password of the machine that will become the root master server.
- The name of the new root domain. The root domain name must have at least two elements (labels) and end in a dot (for example, *something*`.com`.). The last element must be either an Internet organizational name (as shown in Table 18–4), or a two or three character geographic identifier such as `.jp`. for Japan.

TABLE 18–4 Internet Organizational Domains

Domain	Purpose
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
mil	Military groups
net	Major network support centers

TABLE 18–4 Internet Organizational Domains *(Continued)*

Domain	Purpose
org	Nonprofit organizations and others
int	International organizations

The `nis_cachemgr` Daemon

The `nis_cachemgr` should run on all NIS+ clients. The cache manager maintains a cache of location information about the NIS+ servers that support the most frequently used directories in the namespace, including transport addresses, authentication information, and a time-to-live value.

At start-up, the cache manager obtains its initial information from the client's cold-start file, and downloads it into the `/var/nis/NIS_SHARED_DIRCACHE` file.

The cache manager makes requests as a client machine. Make sure the client machine has the proper credentials, or instead of improving performance, the cache manager will degrade it.

Starting and Stopping the Cache Manager

When using the Service Management Facility (SMF), the cache manager has a dependency on the NIS+ service, so cache manager starts and stops along with the NIS+ service. Use the `svcadm` command to start, stop, or restart the NIS+ service.

```
client% svcadm enable /network/rpc/nisplus:default
client% svcadm disable /network/rpc/nisplus:default
client% svcadm restart /network/rpc/nisplus:default
```

When you stop and start the NIS+ service, the cache manager is restarted but it retains the information in the `/var/nis/NIS_SHARED_DIRCACHE` file. The information in the cold-start file is simply appended to the existing information in the cache file. Use the `-i` option to clear the cache file and re-initialize it from the contents of the client's cold-start file.

The nisshowcache Command

The nisshowcache command displays the contents of a client's directory cache.

Displaying the Contents of the NIS+ Cache

The nisshowcache command is located in /usr/lib/nis. It displays only the cache header and the directory names. Here is an example entered from the root master server:

```
rootmaster# /usr/lib/nis/nisshowcache -v
Cold Start directory:
Name : doc.com.
Type : NIS
Master Server :
 Name : rootmaster.doc.com.
 Public Key : Diffie-Hellman (192 bits)
 Universal addresses (3)
 . .
Replicate:
 Name : rootreplica1.doc.com.
 Public Key : Diffie-Hellman (192 bits)
 Universal addresses (3)
 .
 .
 .
Time to live : 12:0:0
Default Access Rights :
```

Pinging and Checkpointing

When a change is made to the NIS+ data set, that change is made in the memory of the master server for the NIS+ domain (or subdomain). A record of the change is also logged in the master server's transaction log (/var/nis/data/trans.log).

Normally, the master server transfers a change in the NIS+ data set to the domain's replica servers 120 seconds (2 minutes) after the change was made. This transfer process is called *pinging*. When the master server pings a replica, it updates the replica's data set with the change. The changed NIS+ data now resides in memory of the master and replica servers.

If the automatic ping process fails to update one or more replica servers, you need to manually force a ping as described in "Forcing a Ping" on page 338. If you suspect that a replica has not been correctly updated with the most current NIS+ data, you can check when the replica was last updated as described in "Displaying When Replicas Were Last Updated" on page 337.

Changes to the NIS+ data set stored in server memory and recorded in the transaction log need to be written into the NIS+ tables stored on disk. The process of updating the NIS+ tables is called *checkpointing*.

Checkpointing is not an automatic process. You must issue the checkpoint command as described in "Checkpointing a Directory" on page 338.

The `nisping` Command

The `nisping` command is used to:

- Display when a replica was last pinged as described in "Displaying When Replicas Were Last Updated" on page 337.
- Force the master server to ping a replica if the automatic ping cycle has not been successful as described in "Forcing a Ping" on page 338.
- Checkpoint servers as described in "Checkpointing a Directory" on page 338.

Displaying When Replicas Were Last Updated

When used with the `-u` option, the `nisping` command displays the update times for the master and replicas of the local domain.

```
/usr/lib/nis/nisping -u [domain]
```

To display the last updates in some other domain, specify the domain name in the command line. Note that when used with the `-u` option, the `nisping` command does not actually ping any replicas.

For example, to display the most recent replica update times for the local `doc.com.` domain, you would enter:

```
rootmaster# /usr/lib/nisping -u
Last updates for directory doc.com.:
Master server is rootmaster.doc.com.
 Last update occurred at Wed Nov 25 10:53:37 1992
Replica server is rootreplica1.doc.com.
 Last update seen was Wed Nov 25 10:53:37 1992
```

Forcing a Ping

If the `nisping -u` command reveals that a replica has not been properly updated, you can use the `nisping` command to force the master server to ping all the replicas in a domain, or one replica in particular.

To ping all the replicas, use the `nisping` command without options:

```
/usr/lib/nis/nisping
```

This forces the master server to ping all the replicas in the domain. Here is an example that pings all the replicas of the local `doc.com.` domain:

```
rootmaster# /usr/lib/nis/nisping
Pinging replicas serving directory doc.com.:
Master server is rootmaster.doc.com.
 Last update occurred at Wed Nov 25 10:53:37 1992
Replica server is rootreplica1.doc.com.
 Last update seen was Wed Nov 18 11:24:32 1992
 Pinging ... rootreplica1.doc.com.
```

To ping all the replicas in a domain other than the local domain, append a domain name:

```
/usr/lib/nis/nisping domainname
```

You can also ping all the tables in all the directories on a single specified host. To ping all the tables in all the directories of a particular host, us the `-a` option:

```
/usr/lib/nis/nisping -a hostname
```

Checkpointing a Directory

Each domain and subdomain should be checkpointed at least once every 24 hour, or more often if the transaction log grows too large in relationship to swap space or total disk space.

Note – Checkpointing large domains, or any domain with a large transaction log, is a time-consuming process which ties up NIS+ servers and slows NIS+ service. While a server is checkpointing, it will still answer requests for service, but it will be unavailable for updates. If possible, checkpoint operations should be scheduled for times when system use is low. You can use the `cron` file to schedule checkpoint operations.

To perform a checkpoint operation, run `nisping -C` on the domain's master server. It is good practice to first ping all replicas before checkpointing. This ensures that the replicas are checkpointing data that is current and up to date.

- To checkpoint a particular directory, run the `nisping` command with the `-C` *directoryname* option. For example,

  ```
  rootmaster# /usr/lib/nis/nisping
  rootmaster# /usr/lib/nis/nisping -C org_dir
  ```

- To checkpoint *all* the directories in the local domain, run the `nisping` command with the `-C -a` options. For example,

  ```
  rootmaster# /usr/lib/nis/nisping
  rootmaster# /usr/lib/nis/nisping -C -a
  ```

Once a server has transferred information from the server's transaction log to the appropriate NIS+ tables, the transactions in the log file are erased to conserve disk space.

For example, to checkpoint all of the directories in the `doc.com.` domain, you would enter:

```
rootmaster# /usr/lib/nis/nisping -C -a
Checkpointing replicas serving directory doc.com. :
Master server is rootmaster.doc.com.
 Last update occurred at Wed May 25 10:53:37 1995
Master server is rootmaster.doc.com.
checkpoint has been scheduled with rootmaster.doc.com.
Replica server is rootreplica1.doc.com.
 Last update seen was Wed May 25 10:53:37 1995
Replica server is rootreplica1.doc.com.
checkpoint has been scheduled with rootmaster.doc.com.
```

The `nislog` Command

The `nislog` command displays the contents of the transaction log.

```
/usr/sbin/nislog
/usr/sbin/nislog -h [number]
/usr/sbin/nislog -t [number]
```

TABLE 18–5 Options for the `nislog` Command

Option	Purpose
`-h [`*num*`]`	Display transactions starting with the head (beginning) of the log. If the number is omitted, the display begins with the first transaction. If the number 0 is entered, only the log header is displayed.

TABLE 18–5 Options for the `nislog` Command *(Continued)*

Option	Purpose
`-t [`*num*`]`	Display transactions starting backward from the end (tail) of the log. If the number is omitted, the display begins with the last transaction. If the number 0 is entered, only the log header is displayed.
`-v`	Verbose mode.

Displaying the Contents of the Transaction Log

Each transaction consists of two parts: the particulars of the transaction and a copy of an object definition.

Here is an example that shows the transaction log entry that was made when the `doc.com.` directory was first created. "XID" refers to the transaction ID.

```
rootmaster# /usr/sbin/nislog -h 1
NIS Log printing facility.
NIS Log dump:
 Log state : STABLE
Number of updates : 48
Current XID : 39
Size of log in bytes : 18432
***UPDATES***
@@@@@@@@@@@@@@TRANSACTION@@@@@@@@@@@@@@
#00000, XID : 1
Time : Wed Nov 25 10:50:59 1992
Directory : doc.com.
Entry type : ADD Name
Entry timestamp : Wed Nov 25 10:50:59 1992
Principal : rootmaster.doc.com.
Object name : org_dir.doc.com.
...................Object.....................
Object Name : org_dir
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : doc.com.
Access Rights : r---rmcdr---r---
Time to Live : 24:0:0
Object Type : DIRECTORY
Name : 'org_dir.doc.com.'
Type: NIS
Master Server : rootmaster.doc.com.
.
.
................................................
@@@@@@@@@@@@@@TRANSACTION@@@@@@@@@@@@@@
#00000, XID : 2
```

The `nischttl` Command

The `nischttl` command changes the time-to-live value of objects or entries in the namespace. This time-to-live value is used by the cache manager to determine when to expire a cache entry. You can specify the time-to-live in total number of seconds or in a combination of days, hours, minutes, and seconds.

The time-to-live values you assign objects or entries should depend on the stability of the object. If an object is prone to frequent change, give it a low time-to-live value. If it is steady, give it a high one. A high time-to-live is a week; a low one is less than a minute. Password entries should have time-to-live values of about 12 hours to accommodate one password change per day. Entries in tables that don't change much, such as those in the RPC table, can have values of several weeks.

To change the time-to-live of an object, you must have modify rights to that object. To change the time-to-live of a table entry, you must have modify rights to the table, entry, or columns you wish to modify.

To display the current time-to-live value of an object or table entry, use the `nisdefaults -t` command, described in Chapter 15.

To change the time-to-live value of objects, use:

```
nischttl time-to-live object-name
```

or

```
nischttl [-L] time-to-live object-name
```

To change the time-to-live value of entries, use:

```
nischttl time-to-live \
 [column=value,...], \
 table-name
```

or

```
nischttl [-ALP] time-to-live \
 [column=value,...], \
 table-name
```

Where *time-to-live* is expressed as:

- *Number of seconds.* A number with no letter is interpreted as a number of seconds. Thus, `1234` for TTL would be interpreted as 1,234 seconds. A number followed by the letter `s` is also interpreted as a number of seconds. Thus, `987s` for TTL would be interpreted as 987 seconds. When seconds are specified in combination with days, hours, or minutes, you *must* use the letter `s` to identify the seconds value.
- *Number of minutes.* A number followed by the letter `m` is interpreted as a number of minutes. Thus, `90m` for TTL would be interpreted as 90 minutes.

- *Number of hours.* A number followed by the letter `h` is interpreted as a number of hours. Thus, `9h` for TTL would be interpreted as 9 hours.
- *Number of days.* A number followed by the letter `d` is interpreted as a number of days. Thus, `7d` for TTL would be interpreted as 7 days.

These values may be used in combination. For example, a TTL value of `4d3h2m1s` would specify a time to live of four days, three hours, two minutes, and one second.

The following flags may also be used with the `nischttl` command:

TABLE 18–6 `nischttl` Syntax Options

Option	Purpose
`A`	All. Apply the change to all the entries that match the *column=value* specifications that you supply.
`L`	Links. Follow links and apply the change to the linked object rather than the link itself.
`P`	Path. Follow the path until there is one entry that satisfies the condition.

Changing the Time-to-Live of an Object

To change the time-to-live of an object, type the `nischttl` command with the *time-to-live* value and the *object-name*. You can add the `-L` command to extend the change to linked objects.

```
nischttl -L time-to-live object-name
```

You can specify the *time-to-live* in seconds by typing the number of seconds. Or you can specify a combination of days, hours, minutes, and seconds by using the suffixes `s`, `m`, `h`, and `d` to indicate the number of seconds, minutes, days, and hours. For example:

```
client% nischttl 86400 sales.doc.com.
client% nischttl 24h sales.doc.com.
client% nischttl 2d1h1m1s sales.doc.com.
```

The first two commands change the time-to-live of the `sales.doc.com.` directory to 86,400 seconds, or 24 hours. The third command changes the time-to-live of all the entries in a hosts table to 2 days, 1 hour, 1 minute, and 1 second.

Changing the Time-to-Live of a Table Entry

To change the time-to-live of entries, use the indexed entry format. You can use any of the options, `-A`, `-L`, or `-P`.

```
nischttl [-ALP] time-to-live \
 [column=value,...], \
```

table-name

Note – C-shell users should use quotes to prevent the shell from interpreting the square brackets ([]) around the column value as a meta character.

These examples are similar to those above, but they change the value of table entries instead of objects:

```
client% nischttl 86400 '[uid=99],passwd.org_dir.doc.com.'
client% nischttl 24h '[uid=99],passwd.org_dir.doc.com.'
client% nischttl 2d1h1m1s '[name=fred],hosts.org_dir.doc.com.'
```

CHAPTER **19**

Administering NIS+ Tables

This chapter describes NIS+ tables and how to administer them. (See Table 10–1, for detailed descriptions of the default NIS+ tables.)

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see*System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Tables

Information used by NIS+ is stored in NIS+ tables. (See Chapter 23 for a description of each default NIS+ system tables supplied in the Solaris software.)

For a complete description of NIS+ table-related commands and their syntax and options, see the NIS+ man pages.

Using the `nistbladm` Command With Tables

Note – Some NIS+ table administration tasks can be performed more easily with Solaris Management Console tools if you have them available.

The `nistbladm` command is the primary NIS+ table administration command. The `nistbladm` command is for use on NIS+ tables stored in an NIS+ directory object. With it, you can create, modify, and delete NIS+ tables and entries. To create a table, its directory must already exist. To add entries to the table, the table and columns must already be defined.

To create a table, you must have create rights to the directory under which you will create it. To delete a table, you must have destroy rights to the directory. To modify the contents of a table, whether to add, change, or delete entries, you must have modify rights to the table or the entries.

`nistbladm` Syntax Summary

The general syntax of the `nistbladm` command is:

```
nistbladm options \
  [columspec | columnvalue] \
  [tablename | indexedname]
```

Where:

- *columnspec* is a specification defining a column to be created in a table as described in "Specifying Table Columns" on page 351.
- *columnvalue* identifies a particular cell value in the table identified by *tablename* as described in "`nistbladm` and Column Values" on page 347.
- *tablename* is the name of the table. For example, `hosts.org_dir.doc.com.`
- *indexedname* identifies a particular cell value in a certain table as described in "`nistbladm` and Column Values" on page 347. In essence *indexedname* is the equivalent of *columnvalue* plus *tablename*.

TABLE 19–1 `nistbladm` Options

Option	Description
`-a | -A`	Add an entry to an existing NIS+ table. The `-a` option returns an error if execution of the command would result in overwritting any existing entry. The `-A` option forces execution of the command even if it results in overwriting an existing entry. (See "Adding Entries to a Table" on page 353.)
`-D` *defaults*	Specify a different set of default properties when creating an object. (See the `nistbladm` man page for details.)
`-d`	Destroy a table. (See "Deleting a Table" on page 353.)
`-c`	Create a table. (See "Creating a New Table" on page 350.)
`-r | -R`	Remove one or more entries from an existing NIS+ table. The `-r` option returns an error if execution of the command would result in removal of more than one entry. The `-R` option forces execution of the command even if it results in removing multiple entries. (See "Removing Table Entries" on page 358.)
`-m`	An obsoleted option for modifying table entries that is still supported for backwards compatibility. The `-e` and `-E` options are the preferred method for editing entries.
`-e | -E`	Edit an entry in an existing NIS+ table. The `-e` option returns an error if execution of the command would affect more than one entry. The `-A` option forces execution of the command even if it results in changing an existing entry in such a way as to overwrite a different entry. (See "Modifying Table Entries" on page 356.)

`nistbladm` and Column Values

Column values are used to identify individual entries in tables using the format:

```
columname="value", \
 columnname="value", ...
```

Where:

- *columname* is the name of a table column.
- *value* is the contents of a particular cell within a column. That value is what identifies a table row. (When using *column=value* to create or modify table data, always enclose the *value* element in quotes.)

For example, suppose you had a `hosts` table that listed machine names and IP addresses:

TABLE 19–2 Example Hosts Table

IP address	name	aliases
`172.22.168.4`	`altair`	
`172.22.168.119`	`deneb`	mail
`172.22.168.120`	`regulus`	`dnsmaster`
`172.22.168.121`	`regulus`	`dnsmaster`
`172.22.168.11`	`sirius`	

In this example, your could identify the `altair` entry (row) in three different ways using the *column=value* of:

- `name=altair`
- `address=172.22.168.4`
- `name=altair,address=172.22.168.4`

But notice in the table above that the machine `regulus` is multi-homed and has *two* IP addresses. In that case, the *column=value* of `host=regulus` identifies two rows. To identify just the first `regulus` row, you would enter either:

- `address=172.22.168.120` or
- `address=172.22.168.120.,name=regulus,dnsmaster`

Note – Some `nistbladm` operations require that you enter a *column=value* pair for *every* column in the table.

`nistbladm`, Searchable Columns, Keys, and Column Values

When an NIS+ table is created, one or more columns are designated *searchable* with either the `S` or the `I` flags as described in "Specifying Table Columns" on page 351. You can use the `niscat -o` *tablename* command to display a list of a table's columns and their characteristics.

A table is *keyed* on its searchable columns. This means that each row in the table must have a unique combination of values in the searchable columns. For example, if a table has one searchable column, each table row must have a unique value in that column, no two rows can contain the same value.

For example, suppose you had a table containing one searchable column named `city` and a non-searchable column named `country`. The following rows would all be permitted:

City	Country
San Francisco	United States
Santa Fe	United States
Santiago	Chile

But you could not have two rows like:

City	Country
London	Canada
London	England

If a table has multiple searchable columns, it is the *combination* of values that must be unique. For example, suppose you had a table containing two searchable columns, `Lastname`, `Firstname` and a non-searchable column named `city`. The following rows would all be permitted:

Lastname	Firstname	City
Kuznetsov	Sergei	Odessa
Kuznetsov	Rima	Odessa
Sergei	Alex	Odessa

But you could not have two rows like this:

Lastname	Firstname	City
Kuznetsov	Rima	Odessa
Kuznetsov	Rima	Chelm

NIS+ commands use the values in the searchable columns to identify specific table rows.

`nistbladm` and Indexed Names

In the context of table administration, an NIS+ *indexed name* is a name that combines a table name with column value search criteria to identify and select particular entries in a table. Indexed names use the format:

[*search_criteria*] , *tablename* . *directory*

Note that search_criteria must be enclosed in square brackets `[ ]`. The *search_criteria* use the format:

```
columnname=value, \
 columnname=value,...
```

Where *columnname=value* pairs are column values from the table's searchable columns as described in "`nistbladm` and Column Values" on page 347.

For example, to identify the `altair` entry in Table 19–2 you could use the indexed name:

```
[addr=172.22.168.4,cname=altair],hosts.org_dir.doc.com.
```

The `nistbladm -R` command allows you to remove all the entries in a table by using the two square brackets with nothing between them `[ ]` as a wildcard specifying all table rows.

`nistbladm` and Groups

In a Solaris-NIS+ environment, there are three types of groups:

- UNIX groups. Information about UNIX groups is stored in the `groups.org_dir` table. Use `nistbladm` to administer UNIX group information.
- Netgroups. Information about net groups is stored in the `netgroups.org_dir` table. Use `nistbladm` to administer net group information.
- NIS+ groups. Information about NIS+ groups is stored in one or more tables in the `groups_dir` directory object. Use `nisgrpadm` to administer NIS+ group information.

Note – Do not use `nistbladm` to administer NIS+ groups.

(See "Solaris Groups" on page 313 for more information on the different types of groups and how to work with them.)

Creating a New Table

An NIS+ table must have at least one column and at least one of its columns must be searchable. To create an NIS+ table, use the `nistbladm` command with the `-c` option:

```
nistbladm -c tabletype columnspec \
 ... tablename
```

Where:

- *Tabletype* is simply a name that identifies a class of tables to which this table belongs. You can use any name you choose.
- A *columnspec* specifies the name and characteristics of each column in a new table. Enter one *columnspec* for each column you want in your new table. Separate the *columnspecs* with spaces, as shown in the following example.

```
nistbladm -c tabletype columnspec columnspec \
columnspec tablename
```

Columnspec formats are described in "Specifying Table Columns" on page 351, below.

Specifying Table Columns

Each *columnspec* entry has two to four components in the format:

name=type,rights :

TABLE 19–3 Table Column Components

Component	Description
name	Name of the column
=	An equal sign which is required.
type	[Optional] The type of column specified by the letters `S`, `I` or `C` (see Table 19–4). This component is optional. If no *type* is specified, the column becomes the default type.
rights	[Optional] Access rights. These access rights are over and above those granted to the table as a whole or to specific entries. If no *access* is specified, the column's access rights are those granted to the table as a whole, or to the entry. The syntax for access rights is described in "Specifying Access Rights in Commands" on page 270.

A column can be one of the following types:

TABLE 19–4 Table Column Types

Type	Description
	No column type specified after the = sign. The column is neither searchable nor encrypted.
`S`	Searchable.
`I`	Searchable, but case-insensitive. When NIS+ commands search through the column, they will ignore case.

TABLE 19–4 Table Column Types *(Continued)*

Type	Description
`C`	Encrypted.

NIS+ commands search through the column and identify individual table rows based on the contents of the searchable columns. Searchable columns are designated with either the `S` or the `I` option. In database terminology, a searchable column is a key. The first column in each table must be searchable. The remaining columns do not have to be searchable. Because the table is keyed on the searchable columns, if you have more than one searchable column, they must be the first and subsequent columns and not skip any columns. For example, if only one column in a table is searchable, it has to be the first column. If two columns are searchable, they must be the first two columns. (See "`nistbladm`, Searchable Columns, Keys, and Column Values" on page 348 for more information on searchable columns.)

If you specify only access rights, you don't need to use a comma. If you include one or more of the `-S`, `-I`, or `-C` flags, add a comma before the access rights.

In the example below, a table is created with the addition of column-specific access rights applied to the first two columns:

```
master% nistbladm -c depts Name=I,w+m Site=w+m Name=C \
 divs.mydir.doc.com.
```

For more information about specifying column access rights when creating a table, see "Setting Column Rights When Creating a Table" on page 279.

Note – NIS+ assumes that all column entries are null terminated. Applications and routines that write information to NIS+ tables must be configured to null terminate each column entry.

Creating Additional Automount Tables

If you are creating an automount table, the table can have only two columns. The first column must be named `key` and the second column must be named `value`. For example, to create an automount table named `auto1`, you would enter:

```
master% nistbladm -c key-value key=S value= auto1.org_dir.doc.com.
```

Deleting a Table

To delete a table, use the `-d` option and enter the table name:

```
nistbladm -d tablename
```

The table must be empty before you can delete it (see "Removing Table Entries" on page 358). This example deletes the divs table from the `doc.com.` directory:

```
rootmaster% nistbladm -d divs.doc.com.
```

Adding Entries to a Table

To add new entries (rows) to a table, use `nistbladm` with either the `-a` or `-A` options followed by either one or more column=value pairs and the table name or an indexed name as described in "`nistbladm` and Indexed Names" on page 349.

```
nistbladm [-a | -A] indexedname
nistbladm [-a | -A] column="value" \
column="value" \
... tablename
```

When adding new entry rows to a table with either `-a` or `-A`:

- Always enclose the *value* element in quotes. For example, to add an entry where the value of the `cname` column is `deneb`, the *column=value* pair would look like: `cname="deneb"`.
- You *must* specify a value for *every* column in the table.
- To specify that a column in the entry row you are adding is empty use *column*`=" "`. In other words, for the *value*, enclose a space between the quote marks.

Note – NIS+ is a naming service and its tables are designed to store references to objects, not the objects themselves. NIS+ is optimized to support 10,000 objects with a combined total size of all tables not more than 10M bytes. NIS+ does not support individual tables where the sum of field sizes in a single column are greater than approximately 7k. If a table is too large, `rpc.nisd` may fail.

Adding a Table Entry With the `-a` Option

The `-a` option adds an entry to a table unless the entry already exists, in which case it returns an error. An entry is defined as *existing* if its values in the searchable columns exactly match the values in the new entry's searchable columns. (The values in non-searchable columns are not taken into account.)

To use the `-a` option, you must specify a value for every column in the table:

```
nistbladm -a column="value" \
 column="value" \
 ... tablename
nistbladm -a indexedname
```

(To list the names and characteristics of table columns, use the `niscat -o` *tablename* command.)

For example, to add a new row to a table named `depts` using column=value pairs, you would enter:

```
rootmaster% nistbladm -a Name='R&D' Site='SanFran' \
 Name='vattel' depts.doc.com.
```

To add the same entry using an indexed name, you would enter:

```
rootmaster% nistbladm -a [Name='R&D',Site='SanFran',\
 Name='vattel'],depts.doc.com.
```

Both examples would produce a table row that looked like this:

Dept	Site	Name
R&D	SanFran	vattel

C-shell users should also use quotes to set off expressions using square brackets.

You can only add one entry with each instance of the `nistbladm` command. You must run `nistbladm` once for each entry row you want to add.

If a table row already exists with values in each column that are identical to the entry you are trying to create, `nistbladm -a` will return an error. You cannot have two identical entry rows in a table. In this context, rows are considered *identical* if the values in the *searchable* columns are identical, the values in none search able columns are not considered.

For example, if the `Dept` and `Site` columns are searchable, and the `Name` column is not searchable, `nistbladm` considers the following two rows to be identical:

Dept (searchable)	Site (searchable)	Name (not searchable)
Sales	Vancouver	Hosteen
Sales	Vancouver	Lincoln

In this example, nistbladm -a would not allow you to create the `Sales Vancouver Lincoln` row.

However if just *some* of the searchable columns have values identical to the entry you are trying to create, `nistbladm -a` will create a new entry as specified. For example, you could run the following commands to create two similar, but not identical, rows in a `depts` table:

```
rootmaster% nistbladm -a Dept='Sales' \
   Site='Vancouver' Name='hosteen' staff.doc.com.
rootmaster% nistbladm -a Dept='Sales' \
   Site='SanFran' Name='lincoln' staff.doc.com.
```

Which would produce rows that had some, but not all identical values in the searchable columns:

Dept	Site	Name
Sales	Vancouver	hosteen
Sales	SanFran	lincoln

Adding a Table Entry With the -A Option

The `-A` option is designed for applications where you need to force `nistbladm` to overwrite an existing entry. Like the `-a` option, `-A` adds a new entry to a table. However, if the entry already exists, instead of exiting with an error, it overwrites the existing entry row.

When using the `-A` option, you must specify all columns in the entry.

For example, suppose the following table exists and the `Dept` and `Site` columns are searchable:

Dept (searchable)	Site (searchable)	Name
Sales	SanFran	Lincoln

Now you run the following command:

```
rootmaster% nistbladm -A Name=Sales Site=SanFran \
 Name=Tsosulu depts.doc.com.
```

The `-a` option would have returned an error, since the entry specified by `Name=Sales Site=SanFran` already exists. But the `-A` option allows you to overwrite the existing row.

Dept	**Site**	**Name**
Sales	`SanFran`	`Tsosulu`

Modifying Table Entries

Existing table entries are edited (modified) using either the `-e` or `-E` options. The Solaris release also supports use of the `-m` option for backwards compatibility with earlier releases. (All new applications and command line operations should use either the `-e` or `-E` options.)

To edit an existing entry (row) in a table, use `nistbladm` with either the `-e` or `-E` options followed by one or more column=value pairs that specify the new values and ending with an indexed name that identifies a particular row in a table as described in "`nistbladm` and Indexed Names" on page 349.

```
nistbladm [-e | -E] column="value" \
 column="value" \
 . . . indexedname
```

When adding new entry rows to a table with either `-e` or `-E`:

- Always enclose the *value* element in quotes. For example, to change the value of the `cname` column to `deneb`, the *column=value* pair would look like: `cname="deneb"`.
- You can only edit values in searchable columns one entry (row) at a time.
- To specify that a column in the entry row that you are editing be empty, use *column*`=" "`. In other words, for the *value*, enclose a space between the quote marks.

Editing a Table Entry With the `-e` Option

The `-e` option edits an entry in a table unless doing so would result in changing values in searchable columns in more than one entry row, in which case it returns an error. (The values in non-searchable columns are not taken into account.)

```
nistbladm column="value" \
 column="value" \
 . . . indexedname
```

To use the `-e` option, you only need to specify the column values you are changing.

For example, suppose you had the table:

Dept	Site	Name
Sales	SanFran	Tsosulu

To change the value of the `Name` column to `Chandar`, you would enter:

```
master% nistbladm -e Name="Chandar" [Dept='Sales',Site='SanFran'],\
 depts.doc.com.
```

Now the table looks like this:

Dept	Site	Name
Sales	SanFran	Chandar

(Note that in the example above, the indexed name did not need to include the `Name` column because in these examples that column is not searchable.)

C-shell users should also use quotes to set off expressions using square brackets.

You can use the `-e` option to edit the values in searchable columns so long as the new values you specify affect only the single row identified by the indexed name. For example, to change the department to `Manf`, you would enter:

```
master% nistbladm -e Dept="Manf" [Dept='Sales',Site='SanFran'],\
 depts.doc.com.
```

Dept (searchable)	Site (searchable)	Name
Manf	SanFran	Chandar

However, if an entry row already existed with `Manf` and `SanFran` in the searchable columns, the `-e` option would return an error.

You can specify changes to multiple columns so long as they all apply to a single entry row. For example, to change both the Dept and Name values, you would enter:

```
master% nistbladm -e Dept="Manf" Name="Thi" \
 [Dept='Sales',Site='SanFran'],depts.doc.com.
```

Dept (searchable)	Site (searchable)	Name
Manf	SanFran	Thi

Editing a Table Entry With the `-E` Option

The `-E` option is designed for applications where you need to force `nistbladm` to overwrite an existing entry even if doing so will affect more than one entry.

For example, suppose your table had the following rows:

Dept (searchable)	Site (searchable)	Name
Sales	SanFran	Chandar
Sales	Alameda	Achmed

Now you run the following command:

```
master% nistbladm -E Site="Alameda" Mgr="Chu" \
[Div='Sales',Site='SanFran'],depts.doc.com.
```

Which would change the `Sales SanFran Chandar` row to `Sales Alameda Chu`. But `Sales Alameda` are the key values identifying the `Sales Alameda Achmed` row, so that row would also be changed. The result would be a single row where once there had been two rows:

Dept (searchable)	Site (searchable)	Name
Sales	Alameda	Chu

The `-e` option would have returned an error, since the edit would affect more than one row. But the `-E` option allows you to affect more than one entry row.

Removing Table Entries

- To remove a single entry from a table, use the `-r` option as described in "Removing Single Table Entries" on page 359.
- To remove multiple entries from a table, use the `-R` option as described in "Removing Multiple Entries From a Table" on page 359.

Removing Single Table Entries

To remove a single entry from a table, use the `-r` option:

```
nistbladm -r indexed-name
```

This example removes the `Manf-1` entry from the `depts` table:

```
rootmaster% nistbladm -r [Dept=Manf-1,Site=Emeryville,Name=hosteen],\
depts.doc.com.
```

You can specify as few column values as you wish. If NIS+ finds duplicates, it does not remove any entry and returns an error message instead. Thus, you could have removed the `Manf-1` by specifying only the `Site` column value, as in this example:

```
rootmaster% nistbladm -r [Site=Emeryville],depts.doc.com.
```

However, you could *not* have removed the `Sales` entry by specifying only the `Site` column value (`SanFran`), because two entries have that same value (`R&D` and `Sales`):

Dept	Site	Name
R&D	SanFran	kuznetsov
Sales	SanFran	jhill
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

Removing Multiple Entries From a Table

To remove multiple entries from a table, use the `-R` option:

```
nistbladm -R indexedname
```

As with the `-r` option, you can specify as few column values as you wish. Unlike the `-r` option, however, if NIS+ finds duplicates, it removes all of them. You can find the name of a table's column by using the `niscat -o` command. This example removes all entries in which the `Site` is `SanFran`:

```
rootmaster% nistbladm -R [Site=SanFran],depts.doc.com.
```

Dept	Site	Name
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

You can use the `-R` option to remove all the entries from a table. Simply do not specify any column values between the square brackets, as in this example:

```
rootmaster% nistbladm -R [],depts.doc.com.
```

When used with the `nistbladm -R` command, an empty set of square brackets is interpreted as a wildcard specifying all table rows.

The `niscat` Command

The `niscat` command displays the contents of an NIS+ table. However, you can also use it to display the object properties of the table. You must have read rights to the table, entries, or columns that you wish to display.

`niscat` Command Syntax

To display the contents of a table, use:

```
niscat [-hM] tablename
```

To display the object properties of a table, use:

```
niscat -o tablename
niscat -o entry
```

TABLE 19–5 `niscat` Options

Option	Description
`-h`	Header. Displays a header line above the table entries, listing the name of each column.
`-M`	Master. Displays only the entries of the table stored on the Master server. This ensures you get the most up-to-date information and should be used only for debugging.
`-o`	Object. Displays object information about the table, such as column names, properties, and servers.

Displaying the Contents of a Table

To display the contents of a table, use `niscat` with a table name:

```
niscat tablename
```

This example displays the contents of the table named `depts`.

```
rootmaster% niscat -h depts.doc.com.
#Name:Site:Name
R&D:SanFran:kuznetsov
Sales:SanFran:jhill
Manf-1:Emeryville:hosteen
Manf-2:Sausalito:lincoln
```

Note – The symbol `*NP*` indicates that you do not have permission to view that entry. Permissions are granted on a table, column, or entry (row) basis. For more on access permissions, see Chapter 15.

Displaying the Object Properties of a Table or Entry

To list the object properties of a table, use `niscat -o` and the table's name:

```
niscat -o tablename.org_dir
```

To display the object properties of a table entry, use `niscat -o` and specify the entry with an indexed name:

```
entry ::=column=value \
 ... tablename | \
 [column=value,...],\
 tablename
```

Here are two examples, one for a table and one for a table entry:

Table

```
rootmaster# niscat -o hosts.org_dir.doc.com.
Object Name : hosts
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : org_dir.doc.com.
Access Rights : ----rmcdr---r---
Time to Live : 12:0:0
Object Type : TABLE
Table Type : hosts_tbl
Number of Columns : 4
Character Separator :
Search Path :
Columns :
 [0] Name : cname
 Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INS
 Access Rights: ----------------
 [1] Name : name
 Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INS
 Access Rights: ----------------
 [2] Name : addr
 Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INS
 Access Rights: ----------------
 [3] Name : comment
 Attributes : (TEXTUAL DATA)
 Access Rights: ----------------
```

Table entry

```
rootmaster# niscat -o [name=rootmaster],hosts.org_dir.doc.com.
Object Name : hosts
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : org_dir.doc.com.
Access Rights : ----rmcdr---r---
Time to Live : 12:0:0
Object Type : ENTRY
 Entry data of type hosts_tbl
 Entry has 4 columns.
 .
#
```

The `nismatch` and `nisgrep` Commands

The `nismatch` and `nisgrep` commands search through NIS+ tables for entries that match a particular string or regular expression, respectively. They display either the entries themselves or a count of how many entries matched. The differences between the `nismatch` and `nisgrep` commands are highlighted in Table 19–6 below.

TABLE 19–6 Characteristics of `nismatch` and `nisgrep`

Characteristics	nismatch	nisgrep
Search criteria	Accepts text only	Accepts regular expressions
Speed	Faster	Slower
Searches through	Searchable columns only	All columns, whether searchable or not
Syntax of search criteria	*column=string* `...` *tablename* `[` *column=* *string*`, ...],` *tablename*	*column=exp* `...` *tablename*

The tasks and examples in this section describe the syntax for both commands.

To use either command, you must have read access to the table you are searching through.

The examples in this section are based on the values in the following table, named `depts.doc.com.` Only the first two columns are searchable.

Name (S)	Site (S)	Name
R&D	SanFran	kuznetsov
Sales	SanFran	jhill
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln
Shipping-1	Emeryville	tsosulu
Shipping-2	Sausalito	katabami
Service	Sparks	franklin

About Regular Expressions

Regular expressions are combinations of text and symbols that you can use to search for special configurations of column values. For example, the regular expression `'Hello'` searches for a value that begins with `Hello`. When using a regular expression in the command line, be sure to enclose it in quotes, since many of the regular expression symbols have special meaning to the Bourne and C shells. For example:

```
rootmaster% nisgrep -h greeting='Hello' phrases.doc.com.
```

The regular expression symbols are summarized in Table 19–7, below.

TABLE 19–7 Regular Expression Symbols

Symbol	Description
`^`*string*	Find a value that begins with *string.*
string `$`	Find a value that ends with *string*.
`.`	Find a value that has a number characters equal to the number of periods.
`[`*chars*`]`	Find a value that contains any of the characters in the brackets.
`*`*expr*	Find a value that has zero or more matches of the *expr.*
`+`	Find something that appears one or more times.

TABLE 19–7 Regular Expression Symbols *(Continued)*

Symbol	Description
`?`	Find any value.
`\'`*s-char*`'`	Find a special character, such as ? or $.
`x | y`	Find a character that is either `x` or `y`.

`nismatch` and `nisgrep` Command Syntax

To search through the first column, use:

```
nismatch string tablename
nisgrep reg-exp tablename
```

To search through a particular column, use:

```
nismatch column=string tablename
nisgrep column=reg-exp tablename
```

To search through multiple columns, use:

```
nismatch column=string tablename ...\
nismatch [column=string,...],tablename
nisgrep column=reg-exp ... \
    tablename
```

TABLE 19–8 `nismatch` and `nisgrep` Options

Option	Description
`-c`	Count. Instead of the entries themselves, displays a count of the entries that matched the search criteria.
`-h`	Header. Displays a header line above the entries, listing the name of each column.
`-M`	Master. Displays only the entries of the table stored on the master server. This ensures you get the most up-to-date information and should be used only for debugging.

Searching the First Column

To search for a particular value in the first column of a table, simply enter the first column value and a *tablename*. In `nismatch`, the value must be a string. In `nisgrep`, the value must be a regular expression.

```
nismatch [-h] string tablename
nisgrep [-h] reg-expression tablename
```

This example searches through the `depts` table for all the entries whose first column has a value of `R&D`:

```
rootmaster% nismatch -h ‘R&D’ depts.doc.com.
rootmaster% nisgrep -h ‘R&D’ depts.doc.com.
```

Note – Quotes are used in the ’R&D’ expression above to prevent the shell from interpreting the ampersand (`&`) as a metacharacter.

Searching a Particular Column

To search through a particular column other than the first, use the following syntax:

```
nismatch column=string tablename
nisgrep column=reg-expression tablename
```

This example searches through the depts table for all the entries whose second column has a value of `SanFran`:

```
rootmaster% nismatch -h Site=SanFran depts.doc.com.
rootmaster% nisgrep -h Site=SanFran depts.doc.com.
```

Searching Multiple Columns

To search for entries with matches in two or more columns, use the following syntax:

```
nismatch [-h] [column=string, ... \
 column=string,...],tablename
nisgrep [-h] column=reg-exp ... \
 tablename
```

This example searches for entries whose second column has a value of `SanFran` and whose third column has a value of `jhill`:

```
rootmaster% nismatch -h [Site=SanFran,Name=jhill], depts.doc.com.
rootmaster% nisgrep -h Site=SanFran Name=jhill depts.doc.com.
```

The `nisln` Command

The `nisln` command creates symbolic links between NIS+ objects such as tables and directories. All NIS+ administration commands accept the `-L` flag, which directs them to follow links between NIS+ objects.

Note – Do not link table entries. Tables may be linked to other tables, but do not link an entry in one table to an entry in another table.

To create a link to another object (table or directory), you must have modify rights to the source object; that is, the one that will point to the other object or entry.

Caution – Never link a `cred` table. Each `org_dir` directory should have its own `cred` table. Do not use a link to some other `org_dir` `cred` table.

`nisln` Command Syntax

To create a link, use:

`nisln` *source target*

TABLE 19–9 `nisln` Options

Option	Description
`-L`	Follow links. If the *source* is itself a link, the new link will not be linked to it, but to that link's original source.
`-D`	Defaults. Specify a different set of defaults for the linked object. Defaults are described in "Specifying Nondefault Security Values at Creation Time" on page 277.

Creating a Link

To create a link between objects such as tables and directories, specify both object names: first the *source*, and then the *target*. Do not link table entries.

`nisln` *source-object target-object*

The `nissetup` Command

The `nissetup` command expands an existing NIS+ directory object into a domain by creating the `org_dir` and `groups_dir` directories, and a full set of NIS+ tables. It does not, however, populate the tables with data. For that, you will need the `nisaddent` command, described in "The `nisaddent` Command" on page 368. Expanding a directory into a domain is part of the process of setting up a domain.

Note – When setting up a new NIS+ domain, the nisserverscript is easier to use than the nissetup command. See "Setting Up NIS+ Root Servers" on page 90 for a full description of using nisserver.

The nissetup command can expand a directory into a domain that supports NIS clients as well.

To use nissetup, you must have modify rights to the directory under which you'll store the tables.

Expanding a Directory Into an NIS+ Domain

You can use the nissetup command with or without a directory name. If you don't supply the directory name, it uses the default directory. Each object that is added is listed in the output.

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

Expanding a Directory Into an NIS-Compatible Domain

To expand a directory into a domain that supports NIS+ and NIS client requests, use the -Y flag. The tables are created with read rights for the nobody class so that NIS clients requests can access them.

```
rootmaster# /usr/lib/nis/nissetup -Y Test.doc.com.
```

The `nisaddent` Command

The `nisaddent` command loads information from text files or NIS maps into NIS+ tables. It can also dump the contents of NIS+ tables back into text files. If you are populating NIS+ tables for the first time, see the instructions in "Populating NIS+ Tables" on page 96. It describes all the prerequisites and related tasks.

You can use `nisaddent` to transfer information from one NIS+ table to another (for example, to the same type of table in another domain), but not directly. First, you need to dump the contents of the table into a file, and then load the file into the other table. Be sure, though, that the information in the file is formatted properly. Chapter 10 describes the format required for each table.

When you load information into a table, you can use any of three options: replace, append, or merge. The append option simply adds the source entries to the NIS+ table. With the replace option, NIS+ first deletes all existing entries in the table and then adds the entries from the source. In a large table, this adds a large set of entries into the table's `.log` file (one set for removing the existing entries, another for adding the new ones), taking up space in `/var/nis` and making propagation to replicas time consuming.

The merge option produces the same result as the replace option but uses a different process, one that can greatly reduce the number of operations that must be sent to the replicas. With the merge option, NIS+ handles three types of entries differently:

- Entries that exist only in the source are *added* to the table
- Entries that exist in both the source and the table are *updated* in the table
- Entries that exist only in the NIS+ table are *deleted* from the table

When updating a large table with a file or map whose contents are not greatly different from those of the table, the merge option can spare the server a great many operations. Because the merge option deletes only the entries that are not duplicated in the source (the replace option deletes *all* entries, indiscriminately), it saves one delete and one add operation for every duplicate entry.

If you are loading information into the tables for the first time, you must have create rights to the table object. If you are overwriting information in the tables, you must have modify rights to the tables.

`nisaddent` Command Syntax

To load information from text files, use:

```
/usr/lib/nis/nisaddent -f filename table-type\ [domain]
/usr/lib/nis/nisaddent -f filename \
 -t tablename table-type  [domain]
```

To load information from NIS maps, use:

```
/usr/lib/nis/nisaddent -y NISdomain table-type\
  [domain]
/usr/lib/nis/nisaddent -y NISdomain -t tablename table-type [domain]
/usr/lib/nis/nisaddent -Y map table-type [domain]
/usr/lib/nis/nisaddent -Y map -t tablename table-type [domain]
```

To dump information from an NIS+ table to a file, use:

```
/usr/lib/nis/nisaddent -d [-t tablename tabletype] \
 > filename
```

Loading Information From a File

You can transfer the contents of a file into an NIS+ table in several different ways:

- The `-f` option with no other option *replaces* the contents of *table-type* in the local domain with the contents of *filename*.

  ```
  nisaddent -f filename table-type
  ```
- With the `-a` option, `-f` *appends* the contents of *filename* to *table-type*.

  ```
  nisaddent -a -f filename table-type
  ```
- With the `-m` option, `-f` *merges* the contents of *filename* into the contents of *table-type*.

  ```
  nisaddent -m -f filename table-type
  ```

The following two examples load the contents of a text file named `/etc/passwd.xfr` into the NIS+ Passwd table. The first is into a table in the local domain, the second into a table in another domain:

```
rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow
rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd sales.doc.com.
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow sales.doc.com.
```

Note – When creating an NIS+ passwd table from `/etc` files, you must run `nisaddent` twice; once on the `/etc/passwd` file and once on the `/etc/shadow` file.

To merge entries from the `/etc/inet/ipnodes` file (IPv6 addresses) into the `ipnodes.org_dir` table, use the `-v` and `-f` options.

```
rootmaster# /usr/lib/nis/nisaddent -mv -f  /etc/inet/ipnodes ipnodes
```

Another way is to use `stdin` as the source. However, you cannot use the `-m` option with `stdin`. You can use redirect (`->`) or pipe (`-|`), but you cannot pipe into another domain.

Task	Command
Redirect	`cat` *filename* `> nisaddent` *table-type*
Redirect with append option	`cat` *filename* `> nisaddent -a` *table-type*
Redirect with append into another domain	`cat` *filename* `> nisaddent -a` *table-type NIS+ domain*
Pipe	`cat` *filename* `| nisaddent` *table-type*
Pipe with append option	`cat` *filename* `| nisaddent -a` *table-type*

If the NIS+ table is an automounter table or a nonstandard table, add the `-t` option and the complete name of the NIS+ table.

```
master# nisaddent -f /etc/auto_home.xfr \
  -t auto_home.org_dir.doc.com.key-value
master# nisaddent -f /etc/auto_home.xfr \
  -t auto_home.org_dir.doc.com. key-value sales.doc.com.
```

Loading Data From an NIS Map

You can transfer information from an NIS map in two different ways; either by specifying the NIS domain or by specifying the actual NIS map. If you specify the domain, NIS+ will figure out which map file in `/var/yp/`*nisdomain* to use as the source, based on the *table-type*. Note that `/var/yp/`*nisdomain* must be *local* files.

NIS+ Table Type	NIS Map Name
Hosts	hosts.byaddr
Nodes	ipnodes.byaddr
Passwd	passwd.byname
Group	group.byaddr
Ethers	ethers.byname
Netmasks	netmasks.byaddr
Networks	networks.byname
Protocols	protocols.byname
RPC	rpc.bynumber
Services	services.byname

To transfer by specifying the NIS domain, use the `-y` (lowercase) option and provide the NIS domain in addition to the NIS+ table type.

Table replacement

```
nisaddent -y nisdomain table-type
```

Table append

```
nisaddent -a -y nisdomain table-type
```

Table merge

```
nisaddent -m -y nisdomain table-type
```

By default, `nisaddent` replaces the contents of the NIS+ table with the contents of the NIS map. Use the `-a` and `-m` options to append or merge. Here is an example that loads the NIS+ passwd table from its corresponding NIS map (`passwd.byname`) in the old-doc domain:

```
rootmaster# /usr/lib/nis/nisaddent -y old-doc passwd
```

This example does the same thing, but for the `sales.doc.com.` domain instead of the local domain, `doc.com.`

```
rootmaster# /usr/lib/nis/nisaddent -y old-doc passwd sales.doc.com.
```

If the NIS+ table is an automounter table or a nonstandard table, add the `-t` option and the complete name of the NIS table, just as you would if the source were a file.

```
rootmaster# nisaddent -y old-doc \
  -t auto_home.org_dir.doc.com. key-value
rootmaster# nisaddent -y old-doc \
  -t auto_home.org_dir.doc.com. key-value sales.doc.com.
```

If instead of using the map files for a domain, you prefer to specify a particular NIS map, use the `-Y` (uppercase) option and specify the map name.

```
rootmaster# nisaddent -Y hosts.byname hosts
rootmaster# nisaddent -Y hosts.byname hosts sales.doc.com.
```

If the NIS map is an automounter map or a non standard map, combine the `-Y` option with the `-t` option:

```
rootmaster# nisaddent -Y auto_home
 -t auto_home.org_dir.doc.com. key-value
rootmaster# nisaddent -Y auto_home
 -t auto_home.org_dir.doc.com. key-value sales.doc.com.
```

Dumping the Contents of an NIS+ Table to a File

To dump the contents of an NIS+ table into a file, use the `-d` and `-t` options. The `-d` options tells the command to dump, and the `-t` option specifies the NIS+ table:

```
rootmaster# nisaddent -d auto_home
 -t auto_home.org_dir.doc.com. key-value
rootmaster# nisaddent -d auto_home
 -t auto_home.org_dir.doc.com. key-value sales.doc.com.
```

CHAPTER 20

Server-Use Customization

This chapter describes how to customize and control which servers NIS+ clients use.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Servers and Clients

When client machines, users, applications, or processes need NIS+ information, they seek an active NIS+ server (master or replica) from which to get the needed data. On large networks, networks with many subnets, and networks that span wide-area links, you may be able to improve NIS+ performance by customizing server usage.

Default Client Search Behavior

By default, if no server preferences have been set with the `nisprefadm` command, a client will first try to obtain the information it needs from an NIS+ server on the client's local subnet. If the client finds an active server on the local subnet, it obtains the information it needs from the first local server that responds. If no server is available on the local subnet, the client searches outside the local subnet, and obtains the NIS+ information it needs from the first remote server that responds.

On large, busy networks, this default search behavior may reduce NIS+ performance for one of two reasons:

- When multiple servers on a subnet are serving a large number of clients, the random nature of the client's default search pattern may result in some servers being over worked while others are under used.
- When a client has to seek an NIS+ server beyond the local subnet, it will obtain its information from the first server that responds even if that server is overworked, or linked to the client's subnet by a slower Wide Area Network connection such as a modem or a dedicated line that is already carrying heavy traffic.

Designating Preferred Servers

The Solaris release contains a new feature—server-use customization—that allows you to control the order in which clients search for NIS+ servers. With this new feature you can balance and customize server usage by:

- Specifying that clients prefer (search for) certain servers over others.
- Specify whether or not clients are permitted to use remote servers if no local servers are available.

The search criteria that you specify can be applied to all clients within a domain, all clients on a subnet, or to individual clients on a machine-by-machine basis.

Note – When server-use preferences are set for a particular machine, those preferences apply to all users, applications, processes, or other clients running on that machine. You cannot set different server-use patterns for different clients on the *same* machine.

NIS+ Over Wide Area Networks

Server-use customization is particularly valuable for large networks with many subnets and networks that span multiple geographic sites connected by modems or leased lines. To maximize network performance, you want to minimize network traffic between subnets, and between sites linked by slower connections. You can do that by specifying which NIS+ servers the clients can use, and their order of server preference. In this way you confine as much NIS+ network traffic as possible to the local subnet.

Optimizing Server-Use—Overview

This section provides an overview of server-use customization.

`nis_cachemgr` is Required

Server-use customization requires that a client be running `nis_cachemgr`. If a client machine is not running `nis_cachemgr`, it cannot make use of server-use customization. If there is no `nis_cachemgr` running on a client machine, that client will use the first server it identifies as described in "Default Client Search Behavior" on page 373.

Global Table or Local File

Depending on how you use the `nisprefadm` command, it creates either a local `client_info` file or a domain `client_info` table:

- *File*. You can use `nisprefadm` to create a local, machine-specific `client_info` file that is stored in the machine's `/var/nis` directory. A local file specifies server preferences for that machine only. When a machine has a local `/var/nis/client_info` file, it ignores any server preferences contained in a domain `client_info.org_dir` table. To create a local `client_info` file, you run `nisprefadm` with the `-L` option.
- *Table*. You can use `nisprefadm` to create an NIS+ `client_info` table which is stored in each domain's `org_dir` NIS+ directory object. This table can specify server preferences for:
  - Individual machines. (If a machine has a local `/var/nis/client_info` file, any preferences for that machine that happen to be in the domain `client_info` table are ignored.)
  - All the machines on a particular subnet. (If a machine on the subnet has a local `/var/nis/client_info` file or individual preferences set for it in the table it ignores subnet preferences.)

  To create a global `client_info` table that applies to all machine on a subnet, you run `nisprefadm` with the `-G` and `-C` options as described in "Specifying Global Server Preferences" on page 382.

  Note that if a machine has its own local `client_info` file as described below, it will ignore all server preferences set for it in a global `client_info` table. If a machine has either a local `client_info` file or a machine-specific entry for it in the global `client_info` table, it will ignore preferences set for its subnet.

Caution – Use only the `nisprefadm` command to make changes to `client_info` files and tables. Never use other NIS+ commands such as `nistbladm`.

When working with `client_info` tables or files, you *must* use either the `-G` or the `-L` option to specify that your command apply to either the global table (`-G`) or local file (`-L`) of the machine you are running the command on.

Preference Rank Numbers

Server preferences are controlled by giving each server a *preference rank number*. Clients search for NIS+ servers in order of numeric preference, querying servers with lower preference rank numbers before seeking servers with higher numbers.

Thus, a client will first try to obtain namespace information from NIS+ servers with a preference of zero. If there are no preference=0 servers available, then the client will query servers whose preference=1. If no 1's are available, it will try to find a 2, and then a 3, and so on until it either gets the information it needs or runs out of servers.

Preference rank numbers are assigned to servers with the `nisprefadm` command as described in "Specifying Global Server Preferences" on page 382.

Server preference numbers are stored in `client_info` tables and files. If a machine has its own `/var/nis/client_info` file, it uses the preference numbers stored in that file. If a machine does not have its own `client_info` file, it uses the preference numbers stored in the domain's `client_info.org_dir` table. These `client_info` tables and files are called "preferred server lists" or simply *server lists*.

You customize server usage by controlling the server preferences of each client. For example, suppose a domain has a client machine named `mailer` that makes heavy use of namespace information and the domain has both a master server (`nismaster`) and a replica server (`replica1`). You could assign a preference number of `1` to `nismaster` and a number of `0` to `replica1` for the `mailer` machine. The `mailer` machine would then always try to obtain namespace information from `replica1` before trying `nismaster`. You could then specify that for all the other machines on the subnet the `nismaster` server had a preference number of zero and `replica1` the number `1`. This would cause the other machine to always try `nismaster` first.

You can give the same preference number to more than one server in a domain. For example, you could assign both `nismaster1` and `replica2` a preference number of `0`, and assign `replica3`, `replica4`, and `replica5` a preference number of `1`.

Default Server Preferences

If there is no `client_info` file or table, the cache manager automatically assigns all servers on the local subnet a default preference number of zero (0) and all servers outside the local subnet a preference of infinite. The purpose of `nisprefadm` is to change these default preference numbers to what you want them to be.

Efficiency and Server Preference Numbers

A client must seek all servers with a given preference number before searching for servers with the next higher number. It requires 5 or more seconds for a client to search for all the servers with a given preference number. This means that if you have a master server and 4 replicas in a domain, and you give each one a *different* preference number from 0 to 4, it could take a client more than 25 seconds to run through all of those preference levels.

To maximize performance, you should not use more than two or three levels of server preference. For example, in the case described above, it is better to give one of those five servers a preference=`0` and all the others a preference of `1`, or give two of them a preference of `1` and the remaining three a preference of `2`.

Preferred Only Servers Versus All Servers

Server lists also specify what a client does if it cannot find *any* preferred servers. A *preferred server* is any server with a preference of zero, or any server that you have assigned a preference number with `nisprefadm`.

By default, if a client fails to reach a preferred server, it will then seek out any server it can find anywhere on the network using the search mode described in "Default Client Search Behavior" on page 373. You can change this default behavior with the `nisprefadm -o` option to specify that a client can only use preferred servers and if no servers are available it cannot go to non-preferred servers. See "Specifying Preferred-Only Servers" on page 387 for details.

Note – This option is ignored when the machine's domain is not served by *any* preferred servers.

Viewing Preferences

To view the server preferences currently in effect for a particular client machine, you run `nisprefadm` with the `-l` option as described in "Viewing Current Server Preferences" on page 380.

Server and Client Names

When specifying server or client machines, keep in mind the following points:

- Server and client names do *not* need to be fully qualified so long as they are in the same NIS+ domain and uniquely identify the object. You can simply use the machine name by itself.
- If a server or subnet is in another NIS+ domain, you need to include enough of the domain name to uniquely identify that machine. For example, if you are in the `sales.doc.com` domain and you need to specify the `nismaster2` machine in the `manf.doc.com` domain, you need only enter `nismaster2.manf`.

Server Preferences

To specify a server preference for:

- *Individual client machine*, use the `-L` option to create a local `client_info` file for the machine you are running the `nisprefadm` on. Use the `-G -C` *machine* options to create machine-specific preferences in the global `client_info` table.
- *All machines on a subnet*, use the `-G -C` *subnetnumber* option.
- *All machines in the current domain that do not have machine-specific or subnet-specific preferences,* use the `-G` option.

When Server Preferences Take Effect

Changes you make to a machine or subnet's server preferences normally do not take effect on a given machine until that machine updates it `nis_cachemgr` data. When the `nis_cachemgr` of a machine updates its server-use information depends on whether the machine is obtaining its server preferences from a global `client_info` table or a local `/var/nis/client_info` file (see "Global Table or Local File" on page 375).

- *Global table*. The cache managers of machines obtaining their server preferences from global tables update their server preferences whenever the machine is booted or whenever the Time-to-live (TTL) value expires for the `client_info` table. By default, this TTL value is 12 hours, but you can change that as described in "Changing the Time-to-Live of an Object" on page 342.
- *Local file*. The cache managers of machines obtaining their server preferences from local files update their server preferences every 12 hours or whenever you run `nisprefadm` to change a server preference. (Rebooting the machine does not update the cache manager's server preference information.)

However, you can force server preference changes to take effect immediately by running `nisprefadm` with the `-F` option. The `-F` option forces `nis_cachemgr` to immediately update its information. See "How to Immediately Implement Preference Changes" on page 391 for details.

Using the `nisprefadm` Command

The following sections describe how to use the `nisprefadm` command to set, modify, and delete server preferences.

The `nisprefadm` command is used to specify the servers that clients are to prefer.

The `nisprefadm` command has the following syntax:

```
nisprefadm -a|-m|-r|-u|-x|-l -L|-G [-o type] \
 [-d domain] \
 [-C machine] \
 servers
nisprefadm -F
```

TABLE 20–1 `nisprefadm` Command Options

Option	Description
`-G`	Create a global client_info table stored in the domain's `org_dir` directory. In other words, create a global preferred server list. This option must be used with either `-C` *subnet* to specify preferences for all the machines on a given subnet, or `-C` *machine* to specify preferences for an individual machine.
`-L`	Create a local `client_info` file stored in the local machine's `/var/nis` directory. In other words, create a preferred server list that applies only to the machine you are running the command on.
`-o` *type*	Specify an option. The valid options are: `pref_type=all`, which specifies that clients can use non-preferred servers if no preferred servers can be contacted, and `pref_type=pref_only`, which specifies that clients may only use the designated preferred servers.
`-d` *domain*	Create a global preferred server client_info table for the specified domain or subdomain.
`-C` *subnet*	The number of a subnet to which the preferences will apply.
`-C` *machine*	The name of a client machine.
servers	One or more NIS+ servers. These are the servers that are to be preferred.
`-a`	Add the specified servers to the server list.
`-m`	Modify the server list. For example, you can use the `-m` option to change the preference number given to one or more servers.
`-r`	Remove the specified servers from the server list.

TABLE 20–1 `nisprefadm` Command Options *(Continued)*

Option	Description
`-u`	Clear the server list, and then add the specified servers. (In other words, replace the current server list with a new list of preferred servers.)
`-x`	Remove the server list completely.
`-l`	List (display) the current preferred server information.
`-F`	Force changes to a preferred server list to take effect immediately.

Note – The `-C` machine option should not be used with the `-L` (local) flag because it has no effect. For example, suppose you are running `nisprefadm` on the `altair` machine. You use the `-L` flag to specify that the preferences you are specifying be written into `altair`'s local `client_info` file. You also use a `-C` vega option to specify that the preferences you are creating be applied to the `vega` machine. The `nisprefadm`command then write your preferences for `vega` into `altair`'s file. But `vega` will never see them because `vega` will always get its server preferences from either its own local `client_info` file or the domain's global `client_info` table. Thus, it only makes sense to use the `-C` option when running `nisprefadm` with the `-G` (global) flag.

Viewing Current Server Preferences

To view current server preferences, run `nisprefadm` with the `-l` option.

How to View Preferences for a Machine

● **Run `nisprefadm` with the `-L` and `-l` options on the machine.**

```
sirius# nisprefadm -L -l
```

This displays any server preferences defined in the machine's local `/var/nis/client_info` file. If there is no local file, no information is displayed and you are returned to your shell prompt.

How to View Global Preferences for Single Machine

- **Run `nisprefadm` with the `-l`, `-G` and `-C` *machinename* options.**

```
sirius# nisprefadm -G -l -C machinename
```

Where *machinename* is the IP address (number) of the machine.

This displays the preferences set in the domain's global `client_info` table for that machine.

How to View Global Preferences for a Subnet

- **Run `nisprefadm` with the `-l`, `-G` and `-C` *subnet* options.**

```
sirius# nisprefadm -G -l -C subnet
```

Where *subnet* is the IP address (number) of the subnet.

This displays the preferences set in the domain's global `client_info` table for that machine.

How to Specify Preference Rank Numbers

By default, all servers listed after the `-a` option are given a preference number of zero. To specify a different preference number, enclose the number in parentheses immediately after the server name like this: `-a` *name*`(`*n*`)`. Where *name* is the name of the server and *n* is the preference number.

For example, assign the `replica2` server a preference number of 3:

```
# nisprefadm -G -a replica2(3)
```

Note – With some shells you may have to enclose the element in quotes like this: `"name(n)"`.

See "Preference Rank Numbers" on page 376 for background information on the server preference rank numbers.

Specifying Global Server Preferences

You can set global server preferences for a local or remote domain. Preferences may be set for individual machines and all the machines on a subnet.

The procedures in this section describe how to specify server preferences in a global `client_info` table residing on the NIS+ domain's master server. Once the table exists on the master server, NIS+ replicates it on to any existing replica servers for the domain.

- See "Specifying Local Server Preference" on page 383 for information on how to create a local `client_info` file on an individual machine.
- See "Global Table or Local File" on page 375 for an explanation of the difference between a global `client_info` table and a local `client_info` file.

To assign server preference numbers, run `nisprefadm` with either the:

- `-a` option to add new or additional preferred servers.
- `-u` option to delete existing server preferences and create new ones.

How to Set Global Preferences for a Subnet

To assign server preferences in the global table for all the machines on a subnet:

● **Run `nisprefadm` with the `-G` and `-C` *subnet* options.**

```
#nisprefadm -G -a -C subnet servers (preferences)
```

Where:

- `-C` *subnet* identifies the IP number of the subnet the preferences will apply to.
- *servers(preferences)* are one or more servers with optional preference ranking numbers.

  For example, to specify that the subnet `172.22.123.123` uses the `nismaster` and `replica3` servers with a default preference number of zero and the `manf.replica6` server with a preference number of 1, type the following.

  ```
  polaris# nisprefadm -a -G -C 172.22.123.123 nismaster1 \
  replica3 "manf.replica6(1)"
  ```

How to Set Global Preferences for an Individual Machine

● **Run `nisprefadm` with the `-G`, and `-C` *machine* options.**

```
#nisprefadm -G -a -C machine servers (preferences)
```

Where:

- `-C` *machine* identifies the machine the preferences will apply to. (Depending on the shell you are using, you may need to enclose *machine* in quotes.)
- *servers(preferences)* are one or more servers with optional preference ranking numbers.

  For example, to replace the current preferences for the machine `cygnus` with `replica7` and `replica9` both with a default preference number of zero, type the following.

```
polaris# nisprefadm -u -G -C cygnus replica7 replica9
```

How to Set Global Preferences for a Remote Domain

To assign server preferences for an individual machine in a remote domain or all the machines on a subnet in a remote domain:

● **Run `nisprefadm` with the `-C`, `-G`, and `-d` options.**

```
#nisprefadm -a -G -C name \
 -d  domain  servers(preferences)
```

Where:

- *name* is the IP number of a subnet or the name of a machine. The modifications you make with this command apply to the subnet or machine that you name.
- *domainname* is the name of the remote domain.
- *servers(preferences)* are one or more servers with optional preference ranking numbers.

For example, to add the `nismaster2` server with a default preference number of zero to the preferred server list of the `10.10.111.11` subnet in the remote `sales.doc.com` domain:

```
polaris# nisprefadm -a -G -C 10.10.111.11 -d sales.doc.com. nismaster2
```

Specifying Local Server Preference

These procedures explain how to create or change a local `client_info` file that specifies server preferences for the machine on which it resides.

If a machine has a local `/var/nis/client_info` file, that machine takes its server preferences from its local file rather than the global `client_info` tables on NIS+ servers. In other words, a local file overrides any global table.

- See “Specifying Global Server Preferences” on page 382 for information on how to create a global `client_info` tables for NIS+ servers.
- See “Global Table or Local File” on page 375 for an explanation of the difference between a global `client_info` table and a local `client_info` file.

To assign server preferences, run `nisprefadm` with either the:

- `-a` option to add new or additional preferred servers.
- `-u` option to delete existing server preferences and create new ones.

How to Set Preferences on a Local Machine

To assign server preferences for the local machine that you are running the `nisprefadm` command on:

- **Run `nisprefadm` with the `-L` option and either the `-a` or `-u` options.**

```
#nisprefadm -a -L servers(preferences)
```

Where *servers(preferences)* are one or more servers with optional preference ranking numbers.

For example, to specify that the `deneb` machine first seek NIS+ information from the `replica3` server with a default preference number of zero and then from the `replica6` server (with a preference number of 1) in the `manf.doc.com` domain:

```
deneb# nisprefadm -a -L replica3 replica6.manf(1)
```

Modifying Server Preferences

You can change a server’s preference number and switch (replace) the preference numbers assigned to different servers.

To change preferred servers or the preference number assigned to a server, run `nisprefadm` with the `-m` *oldserver*`-=`*newserver*`(`*n*`)` option.

How to Change a Server’s Preference Number

- **Run `nisprefadm` with the `-m` *server=server(new)* option.**

```
#nisprefadm -L|-G -C name -m oldserver=newserver(n)
```

Where:

- `-L|-G` determines whether you are modifying a local file or a global table.
- `-C` *name* is the IP number of a subnet or the name of a machine. This option is only used when you are also using the `-G` option. The modifications you make with this command apply to the subnet or machine that you name.
- `-m` is the modify server list option.
- *old server* is the name of the server whose preference number you want to change.
- *new server(n)* is the server name and its new preference number.

For example, on the `deneb` machine, to change the number given to the `replica6.manf` server to 2 in `deneb`'s local `client_info` file:

```
deneb# nisprefadm -L -m replica6.manf=replica6.manf(2)
```

How to Replace One Server With Another in a Preference List

To change one server for another in a preference list:

● **Run `nisprefadm` with the `-m` *oldserver=newserver* option.**

```
#nisprefadm -L|-G -C name -m \
 oldserver=newserver (prefnumber)
```

Where:

- `-L|-G` determine whether you are modifying a local or a domain-wide server list.
- `-C` *name* is the IP number of a subnet or the name of a machine. This option is only used in when you are also using the `-G` option. The modifications you make with this command apply to the subnet or machine that you name.
- `-m` is the modify-server-list option.
- *oldserver* is the old server you are replacing.
- *newserver(prefnumber)* is the new server (with an optional preference number) that is taking the old server's place in the preferred server list.

Keep in mind that when you replace a server in a global `client_info` table using the `-G` option, the replacement only applies to the subnet or machine identified by the `-C` option. Other listings of the replaced server are not affected.

For example, suppose you have a domain with three subnets, and the `replica1` server is listed as a preferred server for two of those subnets. If `replica1` is obsolete and you take it out of service, you then run `nisprefadm -m` to replace it with the new server for the first subnet. Until you do the same for the second subnet, `replica1` is still listed as a preferred server for that subnet. The same principle applies to preferred servers for individual machines.

For example, to replace the `replica3` server with the `replica6` server for subnet `172.21.123.12` in the domain's global `client_info` table and assign `replica6` a preference number of 1:

```
nismaster# nisprefadm -G -C 172.21.123.12 -m replica3 replica6(1)
```

How to Remove Servers From Preference Lists

To remove one or more servers from a preference list:

● **Run `nisprefadm` with the `-r` option.**

```
#nisprefadm -L|-G -C name -r servers
```

Where:

- `-L|-G` determines whether you are modifying a local or a domain-wide server list.
- `-C` *name* is the IP number of a subnet or the name of a machine. This option is only used when you are also using the `-G` option. The preferred servers you remove with this command apply to the subnet or machine that you name.
- `-r` removes the named *servers* from the list.

For example, in the domain's global `client_info` table, to remove the `replica3` and `replica6.manf` servers for the machine `polaris`:

```
polaris# nisprefadm -G -C polaris -r replica3 replica6.manf
```

How to Replace an Entire Preferred Server List

To replace an entire list of preferred servers for a subnet or machine in either a global `client_info` table or a machine in its local `client_info` file, run `nisprefadm` with the `-u` option.

The `-u` option operates the same way as the `-a` option, except that `-u` first deletes any existing server preferences for the machine or subnet before adding the new ones that you specify. (If there are existing preferences, the `-a` option adds the new ones to the old list.)

See "How to Set Global Preferences for an Individual Machine" on page 382 for an example using the `-u` option.

Specifying Preferred-Only Servers

You can specify what clients do when no preferred servers are available.

By default, if a client cannot reach a preferred server, it uses whatever other server it can find. You can specify that clients may only use preferred servers by setting the preferred-only option. See "Preferred Only Servers Versus All Servers" on page 377 for background information on the preferred-only and all servers options.

To specify what clients do when no preferred servers are available, run `nisprefadm` with the `-o` *value* option.

How to Specify Preferred-Only Servers

To specify that clients using a server list may only obtain NIS+ information from servers named in the list:

● **Run `nisprefadm` with the `-o pref_only` option.**

```
#nisprefadm -L|-G -o pref_only
```

Where:

- `-L|-G` determines whether you are modifying a local or a domain-wide server list.
- `-o -pref_only` specifies that clients can only obtain NIS+ information from servers on the list.

Note – This option is ignored for domains that are not served by *any* preferred servers.

For example, to specify in `altair`'s local `client_info` file that `altair` must always use preferred servers and cannot use any server not on `altair`'s preferred server list:

```
altair# nisprefadm -L -o pref_only
```

How to Revert to Using Non-Preferred Servers

To specify that clients using a server list may obtain NIS+ information from servers not named in the list if no preferred servers are available:

- **Run `nisprefadm` with the `-o all` option.**

```
#nisprefadm -L|-G -o all
```

Where:

- `-L|-G` determines whether you are modifying a local or a domain-wide server list.
- `-o -all` specifies that clients may obtain NIS+ information from servers not on the list if no preferred servers are available.

Note – This is the default behavior. You only need to use the `-o all` option if you have previously specified preferred-only servers with the `-o pref_only` option.

For example, to specify in `altair`'s local `client_info` file that `altair` can now use non-preferred servers if no preferred servers can be reached:

```
altair# nisprefadm -L -o all
```

Ending Use of Server Preferences

You can stop using server-use customization and revert to the obtaining NIS+ information as described in "Default Client Search Behavior" on page 373.

To end server preferences, run `nisprefadm` with the `-x` option.

Note – When you end server preferences, clients do not stop using server preferences until the normal course of events as described "When Server Preferences Take Effect" on page 378. You can force an immediate end to server preferences as described in "Putting Server Preferences Into Immediate Effect" on page 390.

How to Eliminate Global Server Preferences

- **Run `nisprefadm` with the `-G` and `-x` options.**

```
#nisprefadm -G -x
```

This eliminates global server preferences.

- Client machines that do not have local server preferences will obtain NIS+ information as described in "Default Client Search Behavior" on page 373.

- Client machines that do have local server preferences set by a local /var/nis/client_info file will continue to use servers as specified in that file.

How to Eliminate Local Server Preferences

Ending local preferences can mean one of three different things:

- That you want the machine to stop using its local client_info file for its server preferences and start using the preferences set for its subnet in the domain's global client_info table.
- That you want this machine to stop using its local client_info file for its server preferences and start using the preferences set for it specifically in the domain's global client_info table.
- That you do not want the machine to use server preferences at all. When a machine does not use server preferences, it obtains NIS+ information as described in "Default Client Search Behavior" on page 373.

How to Switch From Local to Global Subnet Preferences

- **Remove the machine's /var/nis/client_info file.**

```
# rm /var/nis/client_info
```

This causes the machine to use the preferences specified for the machine's subnet in the domain's global client_info table.

How to Switch From Local to Machine-Specific Global Preferences

1. **Remove the machine's /var/nis/client_info file.**

   ```
   # rm /var/nis/client_info
   ```

2. **Specify preferences for the machine in the global table using the -G and -C options.**

   See "How to Set Global Preferences for an Individual Machine" on page 382.

How to Stop a Machine From Using Any Server Preferences

1. **Remove the machine's /var/nis/client_info file.**

   ```
   # rm /var/nis/client_info
   ```

If the machine's domain does not have a global `client_info` table, this step is all you have to do. If the domain does have a `client_info` table, continue on to the next step.

2. **Create an empty `/var/nis/client_info` file.**

```
# touch /var/nis/client_info
```

When a machine has its own `/var/nis/client_info` file, it does not use global preferences from any `client_info` table. If the machine has an empty `/var/nis/client_info` file, it will not use any preferences at all and will obtain NIS+ information, as described in "Default Client Search Behavior" on page 373.

Putting Server Preferences Into Immediate Effect

Server-use changes normally go into effect whenever the client machine is rebooted or updates its cache manager.

When you use `nisprefadm` to set or change server preferences on a local machine using a local `client_info` file (the `-L` option), your changes go into effect immediately.

For machines obtaining their server preferences from a global `client_info` table (the `-G` option) you can force server preference changes into immediate effect by running `nisprefadm` a particular with the `-F` option.

```
# nisprefadm -F
```

The `-F` option forces the machine's cache manager to immediately update its server preference information from the domain's global `client_info` table. (If the machine on which you run `nisprefadm -F` has its own local `client_info` file in `/var/nis`, running nisprefadm `-F` on it will have no effect.)

Note – You cannot use the `-F` option with any other `nisprefadm` options. The `nisprefadm -F` command must always be run by itself on the machine you want it to apply to. You cannot use the `-G` option to update the cache managers of all machines in a domain. The `nisprefadm -F` command must be run on each machine individually.

How to Immediately Implement Preference Changes

To force a newly created or modified server list into immediate effect on a given machine:

- **Run `nisprefadm` with the `-F` option on that machine.**

```
# nisprefadm -F
```

For example, to force immediate implementation of changes to `vega`'s preferred server list (whether local or global):

```
vega# nisprefadm -F
```

CHAPTER **21**

NIS+ Backup and Restore

This chapter describes how to back up and restore an NIS+ namespace.

The NIS+ backup and restore capabilities provide a quick and easy method of preserving and reinstalling your NIS+ namespace. These features can also be used to simplify creation of new replica servers and reduce the time it takes to bring them online. These tasks are performed by two commands:

- `nisbackup`. Backs up NIS+ directory objects.
- `nisrestore`. Restores NIS+ directory objects.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Backing Up Your Namespace With `nisbackup`

The `nisbackup` command backs up one or more NIS+ directory objects or an entire namespace to a specified UNIX file system directory.

Note – The `nisbackup` command is always run on a master server. Never run it on a replica server.

The `nisbackup` command copies the NIS+ namespace data set as of the time the backup command is run. This recording includes all current NIS+ data *and also* any changes entered into the NIS+ namespace by an authorized network administrator but not yet checkpointed (posted) to the NIS+ tables. The backup operation does not check or correct NIS+ data. If data in a table is corrupt, the corrupt data is backed up as if it were valid data.

The `nisbackup` command only backs up those directory objects that the machine is master server for. In other words, you can only use `nisbackup` on a master server. You cannot use `nisbackup` on a replica server.

- If a machine is a master server for both a domain and a subdomain's NIS+ directory objects, you can run `nisbackup` on that machine to back up both domain and subdomain directory objects.
- However, if a machine is a master server for one directory object, and a replica server for a different directory object, you can run `nisbackup` to back up the directory object that the machine is master server for, but it will not back up any objects that the machine is only a replica server for.

If the backup process is interrupted or unable to successfully complete its operation, it halts and restores all previous backup files that were stored in the target directory.

`nisbackup` Syntax

The `nisbackup` command uses the following syntax:

`nisbackup [-v] [-a]` *backupdir objects*

Where:

- *Backupdir* is the target directory where the backup files are to be stored. For example, `/var/master1_bakup`.
- *Objects* are the NIS+ directory objects that you want to back up. For example, `org_dir.doc.com`. Multiple NIS+ directory objects can be listed separated by spaces.

The `nisbackup` command takes the following options:

TABLE 21–1 Options for the `nisbackup` Command

Option	Purpose
`-v`	Verbose mode. This mode provides additional information
-a	All. Backs up all NIS+ directory objects that the server is master of. This includes any sub-domain directory objects that this server is the master for. Note that directory objects of subdomains that have their own master servers will not be backed up.

The `nisbackup` command must be run on the master server for the NIS+ directory objects you are backing up.

When specifying NIS+ directory objects to be backed up, you can use full or partially qualified directory names.

When you back up multi-level directories, the backup files for lower level directories are automatically placed in subdirectories of the target backup directory.

What `nisbackup` Backs Up

When using nisbackup, keep in mind that `nisbackup` is server specific. Regardless of whether or not you use the `-a` option, `nisbackup` only backs up those directories that the server you are running it on is master of. NIS+ directory objects that have some other master server will not be backed up.

For example, suppose the `submaster1` server is master server for the `sales.doc.com.` directory objects and also a replica for the `west.sales.doc.com.` directory objects. When you run `nisbackup` on `submaster1`, only the `sales.doc.com.` directory objects will be backed up.

Some of the implications of this server specific principle are:

- *Entire NIS+ namespace*. If you want to perform an NIS+ back up for an entire multi-domain namespace, *and* your root master server is *also* the master server of all subdomains, you can run `nisbackup` on the root master with the `-a` option. However, if the root master server is *not* the master server of all subdomains, you must also run `nisbackup` on each of the other master servers in order to obtain a complete back up of your entire namespace.
- *Sub-domains*. If you are performing an NIS+ back up for one or more sub-domains, you must run `nisbackup` on the subdomain's master server. If one machine, such as the root master, is also master of one or more subdomains, you can run `nisbackup` on that machine with the `-a` option.

The Backup Target Directory

While the backup target directory must be available to the server being backed up, it is good practice to use a target directory that is not physically mounted on the server. That way you ensure that if the server is damaged the backup directory is still available.

A separate target directory must be used for each master server being backed up. It is good practice to avoid confusion by including the master server's machine name in the target directory name. For example, the target directory for a `nisbackup` run on the `master1` machine might be named `/var/master1_bakup`.

Caution – Never back up more than one master server to a given target directory. Always use different target directories for different master servers. This is because each time you backup one or more NIS+ directory objects to a given target directory, previous backup files for those NIS+ directory objects in that directory are overwritten.

Maintaining a Chronological Sequence of NIS+ Backups

There are at least two ways to maintain an historic sequence of backup files:

- *Different target directories*s. You can maintain separate target directories for each date's backup. For example, `/var/master1_bakup/July14`, and `/var/master1_bakup/July15`, and so on. While this method is simple it wastes disk storage space.
- *File system backup*. The most common method of maintaining an historical sequence of NIS+ backups is to simply include the backup target directory in whatever regular file system backup method that you use. To facilitate this, the `nisbackup` command can be run from a `crontab` file, or from within the Solstice backup routine. See your Solstice documentation for information on how to specify that commands like `nisbackup` be automatically run as part of the system backup procedure.

Backing Up Specific NIS Directories

To back up specific NIS+ directory objects, you list those directories after the target backup directory.

For example, to backup the three `org_dir` directory objects for the root, sales, and manf domains to a `/master1_bakup` directory, you would run `nisbackup` on the `master1` machine as follows:

```
master1# nisbackup /var/master1_bakup org_dir org_dir.sales org_dir.manf
```

Backing Up an Entire NIS+ Namespace

To back up an entire NIS+ namespace you run the `nisbackup` command on the root master server with the `-a` option.

When you use the `-a` option, you do not specify the NIS+ directory objects to be backed up. All NIS+ directory objects on the server and all those of subdomains below it will be automatically backed up.

For example, to backup the `doc.com.` namespace to a `/master1_bakup` directory, you would run `nisbackup` on the root master as follows:

```
rootmaster# nisbackup -a /var/master1_bakup
```

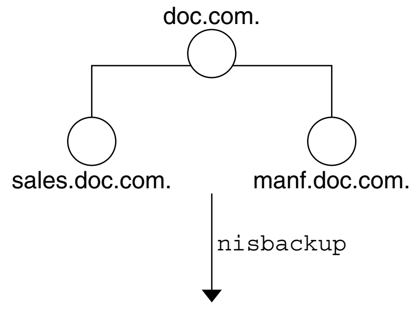
Backup Directory Structure

When you perform a back up on a domain, a subdirectory for each NIS+ directory object is created in the backup target directory. The names of these subdirectories match the fully qualified NIS+ directory object name including the trailing period.

If you perform a full backup of an entire NIS+ object using the `-a` option, then all three of the associated directory objects (*domain.* `org_dir`*.domain.*, and `groups_dir`*.domain.*) are backed up and three target subdirectories are created. If you are backing up multiple objects, subdirectories are created for every object that you are backing up.

Note that the backup subdirectories for multiple NIS+ directory object are all subdirectories of the parent target backup directory regardless of whether or not they are subdomains. In other words, `nisbackup` does not reproduce a domain hierarchy under the parent backup target directory, instead all of the back up subdirectories are simple subddirectories of the target directory.

For example, if you backed up the root, sales, and manf directory objects of `doc.com.` to a `/var/master1_bakup` directory, nine subdirectories would be created in the `/var/master1_bakup` directory as shown in Figure 21–1:



```
/var/master1_bakup
    /master1_bakup/doc.com./
    /master1_bakup/groups_dir.doc.com./
    /master1_bakup/org_dir.doc.com.
    /master1_bakup/sales.doc.com./
    /master1_bakup/groups_dir.sales.doc.com./
    /master1_bakup/org_dir.sales.doc.com./
    /master1_bakup/manf.doc.com./
    /master1_bakup/groups_dir.manf.doc.com./
    /master1_bakup/org_dir.manf.doc.com./
```

FIGURE 21–1 Example directories created by `nisbackup`

Backup Files

The backup target directory contains a `backup_list` file that lists the NIS+ directory objects most recently backed up to this target directory.

Each of the subdirectories contain two files and a `/data` subdirectory. The three files are:

- `data.dict`. An XDR encoded file containing an NIS+ data dictionary for the NIS+ directory objects backed up to this directory.
- `last.upd`. A binary file containing time-stamp information about the NIS+ directory object backed up to this directory.

Each of the `/data` subdirectories contain one or more of the following files:

- `root.object`. An XDR encoded data file containing a description of the NIS+ root directory object. For example, `/master1_bakup/doc.com/data/root.object.`
- `root_dir.` An XDR encoded file containing a description of NIS+ objects contained in the root directory and server information for those objects. For example, `/master1_bakup/doc.com/data/root_dir`.
- *table.directory*. An XDR encoded file containing the data that was present in an NIS+ table at the time the backup was performed *and also* any data contained in any associated NIS+ log files. If there is an NIS+ table in the NIS+ directory object

being backed up, a corresponding *table.directory* backup file will be created in the `/data` subdirectory for that directory object.

For example, every NIS+ `org_dir` directory contains a `hosts` table so there will be a `hosts.org_dir` file in each *target*`/org_dir.`*domain*`/data` subdirectory. For example, `/master1_bakup/org_dir.doc.com./data/hosts.org_dir`

User-created NIS+ tables present in a given directory object are backed up in the same way as the standard NIS+ tables.

- `groups_dir`. An XDR encoded file containing NIS+ groups information. This file is stored in the corresponding NIS+ `groups_dir` target directory.

Restoring Your NIS+ Namespace With `nisrestore`

The `nisrestore` command recreates NIS+ directory objects to match the data stored in backup files created with the `nisbackup` command. This command can be used to restore NIS+ servers, replace directory objects that have become corrupted, or down load NIS+ data on to a new NIS+ server.

Prerequisites to Running `nisrestore`

In order to use `nisrestore` the target machine that will be receiving the NIS+ data from `nisrestore` must have already been set up as an NIS+ server. (See Chapter 4 for a detailed description of setting up NIS+ servers.) This means that:

- The machine must have already been initialized as an NIS+ client.
- If the machine will be running in NIS-compatibility mode and support DNS forwarding, it must have a properly configured `/etc/resolv.conf` file.
- If you are using `nisrestore` on a server while other servers in the namespace are up and running, `nisrestore` will verify with those other servers that this server is configured to serve the backed up NIS+ objects that you are restoring to it. If no other servers are up and running in your namespace, then you must run `nisrestore` with the `-f` option. In other words, if there are other servers that `nisrestore` can check with, you do not need to use the `-f` option. If no other servers are available, for example if you are restoring a single master server and there are no functioning replica servers, then you must use the `-f` option.

Caution – In addition to the three pre-requisites listed above, the `rpc.nisd` daemon must *not* be running on the machine. If necessary, you must kill `rpc.nisd`, by stopping the NIS+ service, before running `nisrestore`.

`nisrestore` Syntax

The `nisrestore` command uses the following syntax:

```
nisrestore [-fv][-a][-t] backupdir [directory_objects]
```

Where:

- *Backupdir* is the directory containing the backup files to be used to restore the NIS+ objects. For example, `/var/master1_bakup`.
- *Directory_objects* are the NIS+ directory objects that you want to restore. For example, `org_dir.doc.com`. Multiple NIS+ directory objects can be listed separated by spaces. (If you run `nisrestore` with the `-a` option, you do not specify specific directory objects.)

The `nisrestore` command takes the following options:

TABLE 21–2 Options for the `nisbackup` Command

Option	Purpose
`-a`	All. Restores all of the NIS+ directory objects contained in the backup directory.
`-f`	Forces the restoration without validating that the server is listed in the directory object's serving list. This option must be used when restoring a root master server or if you get an "unable to lookup object" type of error.
`-v`	Verbose mode. This mode provides additional information
`-t`	This option lists all of the NIS+ directory objects stored in the backup directory. No restoration of objects takes place.

Using `nisrestore`

To restore NIS+ data from NIS+ backup files, use the `nisrestore` command.

For example, to restore the `org_dir.doc.com.` directory object on the `replica1` server, you would log in as root on `replica1`, make sure that the prerequisites described in "Prerequisites to Running `nisrestore`" on page 399 have been met and then run `nisrestore` as shown below:

```
replica1# nisrestore /var/master1_bakup org_dir.doc.com.
```

The following points apply to `nisrestore`:

- *Damaged namespace*. To restore a damaged or corrupted NIS+ namespace, the `nisrestore` command must be run on all of the servers for the NIS+ directory objects you are restoring.
- *Lookup error*. If you get an error message telling you that `nisrestore` cannot verify or look up needed data, then you must use the `-f` option.

  For example, to reload NIS+ data on a root master server named `master1`, you would enter the following.

  ```
  master1# nisrestore -f -a /var/master1_bakup
  ```
- *Directory names*. When specifying the NIS+ directory objects to be restored, you can use full or partially qualified directory names.

Using Backup/Restore to Set Up Replicas

The NIS+ backup and restore features can be used to quickly down load NIS+ data on to a new replica server. For large namespaces this is much faster than `nisping` to obtain data from the master server.

To use `nisbackup` and `nisrestore` to set up a new replica, perform the following:

1. **Run `nisserver` on the master to create the new replica.**
2. **Stop the NIS+ service on the new replica server.**

   Use `svcadm disable`. This interrupts the automatic transfer for namespace data from the master to the replica using the `nisping` command.
3. **Run `nisbackup` on the master server.**
4. **Run `nisrestore` on the new replica to down load the NIS+ data.**
5. **Restart the NIS+ service on the new replica.**

   Use `svcadm restart`.

Replacing Server Machines

You can use `nisbackup` and `nisrestore` to quickly replace a machine that you are using as a server with another machine. For example, you want to improve network performance by replacing an older server with a newer, faster machine.

Machine Replacement Requirements

To replace a machine being used as an NIS+ server with another machine, you *must*:

- Assign the new machine the same IP address as the older machine it is replacing.
- Assign the new machine the same machine name as the older machine it is replacing.
- Connect the new machine to the same subnet as the older machine it is replacing.

How to Replace Server Machines

To replace a server machine, follow these steps:

1. **Run `nisbackup` on the master server for the domain that the old server serves.**

   See "Backing Up an Entire NIS+ Namespace" on page 396. (Note that the old server you are replacing could be the master server for the domain, in which case you would run `nisbackup` on this old master server.)

2. **Copy the old server's `/var/nis/NIS_COLD_START` file to the backup directory.**

3. **Copy the old server's `/etc/.rootkey` file to the backup directory.**

4. **Disconnect the old server from the network.**

5. **Connect the new server to the network.**

6. **Assign the new server the same IP address (number) as the old server.**

7. **Assign the new server the same machine name as the old server.**

8. **If necessary, stop the NIS+ service on the new server.**

9. **Run `nisrestore` on the new server to down load the NIS+ data.**

   See "Restoring Your NIS+ Namespace With `nisrestore`" on page 399.

10. **Copy the `.rootkey` file from the backup directory to `/etc` on the new server.**

11. **Copy the `NIS_COLD_START` file from the backup directory to `/var/nis` on the new server.**

12. Reboot the new server.

CHAPTER 22

Removing NIS+

This chapter describes how to use the NIS+ directory administration commands to remove NIS+ from clients, servers, and the namespace as a whole.

For information on disassociating an NIS+ replica server from a directory so that it no longer acts as a replica for that domain, see "The `nisrmdir` Command" on page 330.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

Note – The NIS+ service is managed by the Service Management Facility. Administrative actions on this service, such as enabling, disabling, or restarting, can be performed by using the `svcadm` command. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+. For an overview of SMF, refer to "Managing Services (Overview)" in *System Administration Guide: Basic Administration*. Also refer to the `svcadm`(1M) and `svcs`(1) man pages for more details.

Removing NIS+ From a Client Machine

This section described how to remove NIS+ from a client machine. Keep in mind that removing NIS+ from a client machine does not remove the NIS+ name service from your network. See "Removing the NIS+ Namespace" on page 407 for information on removing the NIS+ name service from a network and returning to either NIS or `/etc` files for name purposes.

Removing NIS+ That Was Installed Using `nisclient`

To remove NIS+ from a client machine that was set up as an NIS+ client using the `nisclient -i` script as described in Chapter 4, run `nisclient` with the `-r` option:

```
client# nisclient -r
```

`nisclient -r` simply undoes the most recent iteration of `nisclient -i`; it restores the previous naming system used by the client, such as NIS or `/etc` files.

Removing NIS+ That Was Installed Using NIS+ Commands

To remove NIS+ from a client machine that was set up as an NIS+ client using the `nisaddcred`, `domainname`, and `nisinit` commands as described in Chapter 4, perform the following steps:

1. **Remove the `.rootkey` file.**

   ```
   client# rm -f /etc/.rootkey
   ```

2. **Stop the keyserver.**

   ```
   client# svcadm disable /network/rpc/keyserv
   ```

3. **Stop the NIS+ service.**

   This stops the `rpc.nisd` daemon and the `nis_cachemgr`.

   ```
   client# svcadm disable /network/rpc/nisplus:default
   ```

4. **Stop the name service cache (`nscd`).**

   ```
   client# svcadm disable /system/name-service-cache:default
   ```

5. **Remove the `/var/nis` directory and files.**

   ```
   clientmachine# rm -rf /var/nis/*
   ```

Removing NIS+ From a Server

This section describes how to remove NIS+ from an NIS+ server.

Keep in mind that removing NIS+ from a server does not remove the NIS+ name service from your network. See "Removing the NIS+ Namespace" on page 407 for information on removing the NIS+ name service from a network and returning to either NIS or `/etc` files for naming purposes.

Note – You can replace a machine that you are using as an NIS+ server with another machine. See "Replacing Server Machines" on page 402.

To remove NIS+ from a server, follow these steps:

1. **Perform the steps necessary to remove NIS+ from a client.**

   An NIS+ server is also an NIS+ client. This means that you must first remove the client-related part of NIS+. You can use `nisclient -r` as described in "Removing NIS+ That Was Installed Using `nisclient`" on page 406 or the NIS+ command set as described in "Removing NIS+ That Was Installed Using NIS+ Commands" on page 406.

2. **Remove the server's `groups_dir` and `org_dir` directories.**

   ```
   server# nisrmdir -f groups_dir.domainname
   server# nisrmdir -f org_dir.domainname
   ```

3. **Stop the keyserver.**

   ```
   client# svcadm disable /network/rpc/keyserv
   ```

4. **Stop the NIS+ service.**

   This kills the `rpc.nisd` daemon and the `nis_cachemgr`.

   ```
   server# svcadm disable /network/rpc/nisplus:default
   ```

5. **Stop the name service cache (`nscd`).**

   ```
   server# svcadm disable /system/name-service-cache:default
   ```

6. **Remove the `/var/nis` directory and files.**

   ```
   rootmaster# rm -rf /var/nis/*
   ```

Removing the NIS+ Namespace

To remove the NIS+ namespace and return to using either NIS or `/etc` files for name services, follow these steps:

1. **Remove the `.rootkey` file from the root master.**

   ```
   rootmaster# rm -f /etc/.rootkey
   ```

2. **Remove the `groups_dir` and `org_dir` subdirectories from the root master root domain.**

   ```
   rootmaster# nisrmdir -f groups_dir.domainname
   rootmaster# nisrmdir -f org_dir.domainname
   ```

Where *domainname* is the name of the root domain, for example, doc.com.

3. **Remove the root domain.**

   ```
   rootmaster# nisrmdir -f domainname
   ```

   Where *domainname* is the name of the root domain, for example, doc.com.

4. **Stop the keyserver.**

   ```
   client# svcadm disable /network/rpc/keyserv
   ```

5. **Stop the NIS+ service.**

   This kills the `rpc.nisd` daemon and the `nis_cachemgr`.

   ```
   rootmaster# svcadm disable -t /network/rpc/nisplus:default
   ```

6. **Stop the name service cache (`nscd`).**

   ```
   rootmaster# svcadm disable -t /system/name-service-cache:default
   ```

7. **Create a new domain.**

   ```
   rootmaster# domainname name
   ```

   Where *name* is the name of the new domain; for example, the name of the domain before you installed NIS+.

8. **Remove the existing `/etc/defaultdomain` file.**

   ```
   rootmaster# rm /etc/defaultdomain
   ```

9. **Recreate the `/etc/defaultdomain` file with the new domain name.**

   ```
   rootmaster# domainname > /etc/defaultdomain
   ```

10. **Replace the original `nsswitch.conf` file.**

    If you set up this server with `nisserver -r`, you can use:

    ```
    rootmaster# cp /etc/nsswitch.conf.no_nisplus /etc/nsswitch.conf
    ```

    Alternatively, you can copy over one of the default switch template files. To use the default NIS switch file template, you would type:

    ```
    rootmaster# cp /etc/nsswitch.nis etc/nsswitch.conf
    ```

    To use the default `/etc` files switch file template, you would type:

    ```
    rootmaster# cp /etc/nsswitch.files etc/nsswitch.conf
    ```

11. **Remove the `/var/nis` directory and files.**

    ```
    rootmaster# rm -rf /var/nis/*
    ```

12. **Start the NIS+ service.**

    ```
    rootmaster# svcadm enable /network/rpc/nisplus:default
    ```

CHAPTER **23**

Information in NIS+ Tables

This chapter summarizes the information stored in the default NIS+ tables supplied in the Solaris software. This information is also documented in the corresponding manpages.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release. (See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.) For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Tables

In an NIS+ environment, most namespace information is stored in NIS+ tables.

Without a name service, most network information would be stored in `/etc` files and almost all NIS+ tables have corresponding `/etc` files. With the NIS service, you stored network information in NIS maps that also mostly corresponded with `/etc` files.

Note – This chapter describes only those that are distributed as part of NIS+. Users and application developers frequently create NIS+ compatible tables for their own purposes. For information about tables created by users and developers, you must consult the documentation that they provide.

All NIS+ tables are stored in the domain's `org_dir` NIS+ directory object except the admin and groups tables that are stored in the `groups_dir` directory object.

Note – Do not link table entries. Tables can be linked to other tables, but do not link an entry in one table to an entry in another table.

NIS+ Tables and Other Name Services

In the Solaris system the name service switch file (`nsswitch.conf`) allows you to specify one or more sources for different types of namespace information. In addition to NIS+ tables, sources can be NIS maps, DNS zone files, or `/etc` tables. The order in which you specify them in the switch file determines how the information from different sources is combined. (See Chapter 1 for more information on the switch file.)

NIS+ Table Input File Format

If you are creating input files for any of these tables, most tables share two formatting requirements:

- You must use one line per entry.
- You must separate columns with one or more spaces or Tabs.

If a particular table has different or additional format requirements, they are described under the heading, "Input File Format."

auto_home Table

The auto_home table is an indirect automounter map that enables an NIS+ client to mount the home directory of any user in the domain. It does this by specifying a mount point for each user's home directory, the location of each home directory, and mount options, if any. Because it is an indirect map, the first part of the mount point is specified in the auto_master table, which is, by default, /home. The second part of the mount point (that is, the subdirectory under /home) is specified by the entries in the auto_home map, and is different for each user.

The auto_home table has two columns:

TABLE 23–1 auto_home Table

Column	Content	Description
Key	Mount point	The login name of every user in the domain
Value	Options & location	The mount options for every user, if any, and the location of the user's home directory

For example:

```
costas barcelona:/export/partition2/costas
```

The home directory of the user costas, which is located on the server barcelona, in the directory /export/partition2/costas, would be mounted under a client's /home/costas directory. No mount options were provided in the entry.

auto_master Table

The auto_master table lists all the automounter maps in a domain. For direct maps, the auto_master table provides a map name. For indirect maps, it provides both a map name and the top directory of its mount point. The auto_master table has two columns:

TABLE 23–2 `auto_master` Table

Column	Content	Description
Key	Mount point	The top directory into which the map will be mounted. If the map is a direct map, this is a dummy directory, represented with /–.
Value	Map name	The name of the automounter map

For example, assume these entries in the `auto_master` table:

```
/home auto_home
 /-auto_man
 /programs auto_programs
```

The first entry names the `auto_home` map. It specifies the top directory of the mount point for all entries in the `auto_home` map: `/home`. (The `auto_home` map is an indirect map.) The second entry names the`auto_man` map. Because that map is a direct map, the entry provides only the map name. The `auto_man` map will itself provide the topmost directory, as well as the full path name, of the mount points for each of its entries. The third entry names the `auto_programs` map and, since it provides the top directory of the mount point, the `auto_programs` map is an indirect map.

All automounter maps are stored as NIS+ tables. By default, the Solaris system provides the `auto_master` map, which is mandatory, and the `auto_home` map, which is a great convenience.

You can create more automounter maps for a domain, but be sure to store them as NIS+ tables and list them in the `auto_master` table. When creating additional automount maps to supplement `auto_master` (which is created for you), the column names must be `key` and `value`. For more information about the automounter consult your automounter or NFS file system documentation.

`bootparams` Table

The `bootparams` table stores configuration information about every diskless machine in a domain. A diskless machine is a machine that is connected to a network, but has no hard disk. Since it has no internal storage capacity, a diskless machine stores its files and programs in the file system of a server on the network. It also stores its configuration information—or *boot parameters*—on a server.

Because of this arrangement, every diskless machine has an initialization program that knows where this information is stored. If the network has no name service, the program looks for this information in the server's `/etc/bootparams` file. If the network uses the NIS+ name service, the program looks for it in the bootparams table, instead.

The `bootparams` table can store any configuration information about diskless machines. It has two columns: one for the configuration key, another for its value. By default, it is set up to store the location of each machine's root, swap, and dump partitions.

The default `bootparams` table has only two columns that provide the following items of information:

TABLE 23–3 `bootparams` Table

Column	Content	Description
Key	Hostname	The diskless machine's official host name, as specified in the hosts table
Value	Configuration	Root partition: the location (server name and path) of the machine's root partition
		Swap partition: the location (server name and path) of the machine's swap partition
		Dump partition: the location (server name and path) of the machine's dump partition
		Install partition.
		Domain.

Input File Format

The columns are separated with a TAB character. Backslashes (\) are used to break a line within an entry. The entries for root, swap, and dump partitions have the following format:

```
client-name  root=server:path  \
swap=server:path \
dump=server:path \
install=server:path \
domain=domainname
```

Here is an example:

```
buckarooroot=bigriver:/export/root1/buckaroo \
 swap=bigriver:/export/swap1/buckaroo \
 dump=bigriver:/export/dump/buckaroo \
 install=bigriver:/export/install/buckaroo \
 domain=sales.doc.com
```

Additional parameters are available for x86-based machines. See the `bootparams` man page for additional information.

`client_info` Table

The `client_info` table is an optional internal NIS+ table used to store server preferences for the domain in which it resides. This table is created and maintained with the `nisprefadm` command.

Caution – Only use `nisprefadm` to work with this table. Do not use any other NIS+ commands on this table.

`cred` Table

The `cred` table stores credential information about NIS+ principals. Each domain has one `cred` table, which stores the credential information of client machines that belong to that domain and client users who are allowed to log into them. (In other words, the principals of that domain.) The `cred` tables are located in their domains' `org_dir` subdirectory.

Note – Do not link a `cred` table. Each `org_dir` directory should have its own `cred` table. Do not use a link to some other `org_dir` `cred` table.

The `cred` table has five columns:

TABLE 23–4 `cred` Table

			Public Data	**Private Data**
NIS+ Principal Name	**Authentication Type**	**Authentication Name**		
Principal name of a principal user	LOCAL	UID	GID list	
Principal name of a principal user or machine	DES	Secure RPC netname	Public key	Encrypted private key

The second column, authentication type, determines the types of values found in the other four columns.

- *LOCAL*. If the authentication type is LOCAL, the other columns contain a principal user's name, UID, and GID; the last column is empty.
- *DES*. If the authentication type is DES, the other columns contain a principal's name, Secure RPC netname, public key, and encrypted private key. These keys are used in conjunction with other information to encrypt and decrypt a DES credential.

See Chapter 12 for additional information on credentials and the `cred` table.

`ethers` Table

The `ethers` table stores information about the 48-bit Ethernet addresses of machines on the Internet. It has three columns:

TABLE 23–5 `ethers` Table

Column	Content	Description
Addr	Ethernet-address	The 48-bit Ethernet address of the machine
Name	Official-host-name	The name of the machine, as specified in the hosts table
Comment	Comment	An optional comment about the entry

An Ethernet address has the form:

n:*n*:*n*:*n*:*n*:*n hostname*

where *n* is a hexadecimal number between 0 and FF, representing one byte. The address bytes are always in network order (most significant byte first).

`group` Table

The `group` table stores information about UNIX user groups. The `group` table has four columns:

TABLE 23–6 `group` Table

Column	Description
Name	The group's name
Passwd	The group's password
GID	The group's numerical ID
Members	The names of the group members, separated by commas

Earlier Solaris releases used a `+/-` syntax in local `/etc/group` files to incorporate or overwrite entries in the NIS group maps. Since the Solaris system uses the name service switch file to specify a machine's sources of information, this is no longer necessary. All you have to do in Solaris Release 2x systems is edit a client's `/etc/nsswitch.conf` file to specify `files`, followed by `nisplus` as the sources for the `group` information. This effectively adds the contents of the `group` table to the contents of the client's `/etc/group` file.

`hosts` Table

The `hosts` table associates the names of all the machines in a domain with their IP addresses. The machines are usually also NIS+ clients, but they don't have to be. Other tables, such as `bootparams`, `group`, and `netgroup`, rely on the network names stored in this table. They use them to assign other attributes, such as home directories and group memberships, to individual machines. The `hosts` table has four columns:

TABLE 23–7 `hosts` Table

Column	Description
Addr	The machine's IP address (network number plus machine ID number)
Cname	The machine's official name
Name	A name used in place of the host name to identify the machine
Comment	An optional comment about the entry

`mail_aliases` Table

The `mail_aliases` table lists the domain's mail aliases recognized by `sendmail`. It has four columns:

TABLE 23–8 `mail_aliases` Table

Column	Description
Alias	The name of the alias
Expansion	A list containing the members that receive mail sent to this alias; members can be users, machines, or other aliases
Comment	An optional comment about the entry
Options	(See man page for options)

Input File Format

Each entry has the following format:

`alias-name:`*member*`[,`*member*`]...`

To extend an entry over several lines, use a backslash.

`netgroup` Table

The `netgroup` table defines network wide groups used to check permissions for remote mounts, logins, and shells. The members of net groups used for remote mounts are machines; for remote logins and shells, they are users.

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

The `netgroup` table has six columns:

TABLE 23–9 `netgroup` Table

Column	Content	Description
Name	groupname	The name of the network group
Group	groupname	Another group that is part of this group
Host	hostname	The name of a host
User	username	A user's login name
Domain	domainname	A domain name
Comment	Comment	An optional comment about the entry

Input File Format

The input file consists of a group name and any number of members:

groupname member-list. . .

The member list can contain the names of other net groups or an ordered member list with three fields or both:

member-list::=*groupname* | (*hostname*, *username*, *domainname*)

The first field of the member list specifies the name of a machine that belongs to the group. The second field specifies the name of a user that belongs to the group. The third field specifies the domain in which the member specification is valid.

A missing field indicates a wildcard. For example, the `netgroup` specification shown below includes all machines and users in all domains:

```
everybody ( , , )
```

A dash in a field is the opposite of a wildcard; it indicates that no machines or users belong to the group. Here are two examples:

```
(host1, -,doc.com.) (-,joe,doc.com.)
```

The first specification includes one machine, `host1`, in the `doc.com.` domain, but excludes all users. The second specification includes one user in the `doc.com.` domain, but excludes all machines.

`netmasks` Table

The `netmasks` table contains the network masks used to implement standard Internet subnetting. The table has three columns:

TABLE 23–10 `netmasks` Table

Column	Description
Addr	The IP number of the network
Mask	The network mask to use on the network
Comment	An optional comment about the entry

For network numbers, you can use the conventional IP dot notation used by machine addresses, but leave zeros in place of the machine addresses. For example, this entry

```
172.31.0.0 255.255.255.0
```

means that class B network 172.31.0.0 should have 24 bits in its subnet field, and 8 bits in its host field.

`networks` Table

The `networks` table lists the networks of the Internet. This table is normally created from the official network table maintained at the Network Information Control Center (NIC), though you might need to add your local networks to it. It has four columns:

TABLE 23–11 `networks` Table

Column	Description
Cname	The official name of the network, supplied by the Internet
Addr	The official IP number of the network
Name	An unofficial name for the network
Comment	An optional comment about the entry

`passwd` Table

The `passwd` table contains information about the accounts of users in a domain. These users generally are, but do not have to be, NIS+ principals. Remember though, that if they are NIS+ principals, their credentials are not stored here, but in the domain's `cred` table. The `passwd` table usually grants read permission to the world (or to nobody).

Note – The `passwd` table should not have an entry for the user root (user ID 0). Root's password information should be stored and maintained in the machine's `/etc` files.

The information in the `passwd` table is added when users' accounts are created.

The `passwd` table contains the following columns:

TABLE 23–12 `passwd` Table

Column	Description
Name	The user's login name, which is assigned when the user's account is created; the name can contain no uppercase characters and can have a maximum of eight characters
Passwd	The user's encrypted password
UID	The user's numerical ID, assigned when the user's account is created
GID	The numerical ID of the user's default group
GCOS	The user's real name plus information that the user wishes to include in the From: field of a mail-message heading; an "&" in this column simply uses the user's login name
Home	The path name of the user's home directory.
Shell	The user's initial shell program; the default is the Bourne shell: `/usr/bin/sh.`
Shadow	(See Table 23–13.)

The `passwd` table shadow column stores restricted information about user accounts. It includes the following information:

TABLE 23–13 `passwd` Table Shadow Column

Item	Description
Lastchg	The number of days between January 1, 1970, and the date the password was last modified
Min	The minimum number of days recommended between password changes
Max	The maximum number of days that the password is valid
Warn	The number of days' warning a user receives before being notified that his or her password has expired

TABLE 23–13 `passwd` Table Shadow Column *(Continued)*

Item	Description
Inactive	The number of days of inactivity allowed for the user
Expire	An absolute date past which the user's account is no longer valid
Flag	Reserved for future use: currently set to 0.

Earlier Solaris releases used a `+/-` syntax in local `/etc/passwd` files to incorporate or overwrite entries in the NIS password maps. Since the Solaris 2x release uses the name service switch file to specify a machine's sources of information, this is no longer necessary. All you have to do in Solaris Release 2x systems is edit a client's `/etc/nsswitch.conf` file to specify `files`, followed by `nisplus` as the sources for the passwd information. This effectively adds the contents of the passwd table to the contents of the `/etc/passwd` file.

However, if you still want to use the `+/-` method, edit the client's `nsswitch.conf` file to add `compat` as the `passwd` *source* if you are using NIS. If you are using NIS+, add `passwd_compat: nisplus`.

`protocols` Table

The `protocols` table lists the protocols used by the Internet. It has four columns:

TABLE 23–14 `protocols` Table

Column	Description
Cname	The protocol name
Name	An unofficial alias used to identify the protocol
Number	The number of the protocol
Comments	Comments about the protocol

`rpc` Table

The `rpc` table lists the names of RPC programs. It has four columns:

TABLE 23–15 `rpc` Table

Column	Description
Cname	The name of the program
Name	Other names that can be used to invoke the program
Number	The program number
Comments	Comments about the RPC program

Here is an example of an input file for the `rpc` table:

```
#
# rpc file
#
rpcbind    00000    portmap    sunrpc    portmapper
rusersd    100002    rusers
nfs    100003    nfsprog
mountd    100005    mount    showmount
walld    100008    rwall    shutdown
sprayd    100012    spray
llockmgr    100020
nlockmgr    100021
status    100024
bootparam    100026
keyserv    100029    keyserver
nisd    100300    rpc.nisd
#
```

`services` Table

The `services` table stores information about the Internet services available on the Internet. It has five columns:

TABLE 23–16 `services` Table

Column	Description
Cname	The official Internet name of the service
Name	The list of alternate names by which the service can be requested
Proto	The protocol through which the service is provided (for instance, 512/tcp)
Port	The port number

TABLE 23–16 `services` Table *(Continued)*

Column	Description
Comment	Comments about the service

`timezone` Table

The `timezone` table lists the default timezone of every machine in the domain. The default time zone is used during installation but can be overridden by the installer. The table has three columns:

TABLE 23–17 `timezone` Table

Field	Description
Name	The name of the domain
Tzone	The name of the time zone (for example, US/Pacific)
Comment	Comments about the time zone

Additional Default Tables

For information the other default tables:

- audit_user
- auth_attr
- exec_attr
- prof_attr
- user_attr

Refer to the appropriate section (4) man pages.

CHAPTER **24**

NIS+ Troubleshooting

In this chapter, problems are grouped according to type. For each problem there is a list of common symptoms, a description of the problem, and one or more suggested solutions.

In addition, Appendix A contains an alphabetic listing of the more common NIS+ error messages.

Note – NIS+ might not be supported in a future release. Tools to aid the migration from NIS+ to LDAP are available as of the Solaris 9 release (see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*). For more information, visit `http://www.sun.com/directory/nisplus/transition.html`.

NIS+ Debugging Options

The `NIS_OPTIONS` environment variable can be set to control various NIS+ debugging options.

Options are specified after the `NIS_OPTIONS` command separated by spaces with the option set enclosed in double quotes. Each option has the format *name=value*. Values can be integers, character strings, or filenames depending on the particular option. If a value is not specified for an integer value option, the value defaults to 1.

`NIS_OPTIONS` recognizes the following options:

TABLE 24–1 `NIS_OPTIONS` Options and Values

Option	Values	Actions
`debug_file`	*filename*	Directs debug output to specified file. If this option is not specified, debug output goes to `stdout`.
`debug_bind`	*Number*	Displays information about the server selection process.
`debug_rpc`	*1* or *2*	If the value is 1, displays RPC calls made to the NIS+ server and the RPC result code. If the value is 2, displays both the RPC calls and the contents of the RPC and arguments and results.
`debug_calls`	*Number*	Displays calls to the NIS+ API and the results that are returned to the application.
`pref_srvr`	*String*	Specifies preferred servers in the same format as that generated by the `nisprefadm` command (see Table 20–1). This will over-ride the preferred server list specified in `nis_cachemgr`.
`server`	*String*	Bind to a particular server.
`pref_type`	*String*	Not currently implemented.

For example, (assuming that you are using a C-Shell):

- To display many debugging messages you would enter:

```
setenv NIS_OPTIONS "debug_calls=2 debug_bind debug_rpc"
```

- To obtain a simple list of API calls and store them in the file `/tmp/CALLS` you would enter:

```
setenv NIS_OPTIONS "debug_calls debug_file=/tmp/CALLS"
```

- To obtain a simple list of API calls sent to a particular server you would enter:

```
setenv NIS_OPTIONS "debug_calls server=sirius"
```

NIS+ Administration Problems

This section describes problems that may be encountered in the course of routine NIS+ namespace administration work. Common symptoms include:

- `"Illegal object type"` for operation message
- Other "object problem" error messages
- Initialization failure
- Checkpoint failures
- Difficulty adding a user to a group
- Logs too large/lack of disk space/difficulty truncating logs
- Cannot delete `groups_dir` or `org_dir`

Illegal Object Problems

Symptoms

- `"Illegal object type"` for operation message
- Other "`object problem`" error messages

There are a number of possible causes for this error message:

- You have attempted to create a table without any searchable columns.
- A database operation has returned the status of `DB_BADOBJECT` (see the `nis_db` man page for information on the `db` error codes).
- You are trying to add or modify a database object with a length of zero.
- You attempted to add an object without an owner.
- The operation expected a directory object, and the object you named was not a directory object.
- You attempted to link a directory to a LINK object.
- You attempted to link a table entry.
- An object that was not a group object was passed to the `nisgrpadm` command.
- An operation on a group object was expected, but the type of object specified was not a group object.
- An operation on a table object was expected, but the object specified was not a table object.

`nisinit` Fails

Make sure that:

- You can `ping` the NIS+ server to check that it is up and running as a machine.
- The NIS+ server that you specified with the `-H` option is a valid server and that it is running the NIS+ software.
- `rpc.nisd` is running on the server.
- The nobody class has read permission for this domain.
- The netmask is properly set up on this machine.

Checkpoint Keeps Failing

If checkpoint operations with a `nisping -C` command consistently fail, make sure you have sufficient swap and disk space. Check for error messages in `syslog`. Check for `core` files filling up space.

Cannot Add User to a Group

A user must first be an NIS+ principal client with a LOCAL credential in the domain's `cred` table before the user can be added as a member of a group in that domain. A DES credential alone is not sufficient.

Logs Grow too Large

Failure to regularly checkpoint your system with `nisping -C` causes your log files to grow too large. Logs are not cleared on a master until *all* replicas for that master are updated. If a replica is down or otherwise out of service or unreachable, the master's logs for that replica cannot be cleared. Thus, if a replica is going to be down or out of service for a period of time, you should remove it as a replica from the master as described in "Removing a Directory" on page 330. Keep in mind that you must first remove the directory's `org_dir` and `groups_dir` subdirectories before you remove the directory itself.

Lack of Disk Space

Lack of sufficient disk space will cause a variety of different error messages. (See "Insufficient Disk Space" on page 451 for additional information.)

Cannot Truncate Transaction Log File

First, check to make sure that the file in question exists and is readable and that you have permission to write to it.

- You can use `ls /var/nis/trans.log` to display the transaction log.
- You can use `nisls -l` and `niscat` to check for existence, permissions, and readability.
- You can use `syslog` to check for relevant messages.

The most likely cause of inability to truncate an existing log file for which you have the proper permissions is lack of disk space. (The checkpoint process first creates a duplicate temporary file of the log before truncating the log and then removing the temporary file. If there is not enough disk space for the temporary file, the checkpoint process cannot proceed.) Check your available disk space and free up additional space if necessary.

Domain Name Confusion

Domain names play a key role in many NIS+ commands and operations. To avoid confusion, you must remember that, except for root servers, all NIS+ masters and replicas are clients of the domain *above* the domain that they serve. If you make the mistake of treating a server or replica as if it were a client of the domain that it serves, you may get `Generic system error` or `Possible loop detected in namespace` *directoryname*:*domainname* error messages.

For example, the machine `altair` might be a client of the `subdoc.doc.com.` domain. If the master server of the `subdoc.doc.com.` subdomain is the machine `sirius`, then `sirius` is a client of the `doc.com`. domain. When using, specifying, or changing domains, remember these rules to avoid confusion:

1. Client machines belong to a given domain or subdomain.
2. Servers and replicas that serve a given subdomain are clients of the domain above the domain they are serving.
3. The only exception to Rule 2 is that the root master server and root replica servers are clients of the same domain that they serve. In other words, the root master and root replicas are all clients of the root domain.

Thus, in the example above, the fully qualified name of the `altair` machine is `alladin.subdoc.doc.com.` The fully qualified name of the `sirius` machine is `sirius.doc.com.` The name `sirius.subdoc.doc.com.` is wrong and will cause an error because `sirius` is a client of `doc.com.`, not `subdoc.doc.com.`

Cannot Delete `org_dir` or `groups_dir`

Always delete `org_dir` and `groups_dir` *before* deleting their parent directory. If you use `nisrmdir` to delete the domain before deleting the domain's `groups_dir` and `org_dir`, you will not be able to delete either of those two subdirectories.

Removal or Disassociation of NIS+ Directory from Replica Fails

When removing or disassociating a directory from a replica server you must first remove the directory's `org_dir` and `groups_dir` subdirectories before removing the directory itself. After each subdirectory is removed, you must run `nisping` on the parent directory of the directory you intend to remove. (See "Removing a Directory" on page 330.)

If you fail to perform the `nisping` operation, the directory will not be completely removed or disassociated.

If this occurs, you need to perform the following steps to correct the problem:

1. Remove `/var/nis/rep/org_dir` on the replica.
2. Make sure that `org_dir.`*domain* does not appear in `/var/nis/rep/serving_list` on the replica.
3. Perform a `nisping` on *domain*.

4. From the master server, run `nisrmdir -f` *replica_directory*.

If the replica server you are trying to dissociate is down or out of communication, the `nisrmdir -s` command will return a `Cannot remove replica` *name*`: attempt to remove a non-empty table` error message.

In such cases, you can run `nisrmdir -f -s` *replicaname* on the master to force the dissociation. Note, however, that if you use `nisrmdir -f -s` to dissociate an out-of-communication replica, you *must* run `nisrmdir -f -s`*again* as soon as the replica is back on line in order to clean up the replica's `/var/nis` file system. If you fail to rerun `nisrmdir -f -s` *replicaname* when the replica is back in service, the old out-of-date information left on the replica could cause problems.

NIS+ Database Problems

This section covers problems related to the namespace database and tables. Common symptoms include error messages with operative clauses such as:

- `Abort_transaction: Internal database error`
- `Abort_transaction: Internal Error, log entry corrupt`
- `Callback: select failed`
- `CALLBACK_SVC: bad argument`

as well as when `rpc.nisd` fails.

See also "NIS+ Ownership and Permission Problems" on page 436.

Multiple `rpc.nisd` Parent Processes

Symptoms:

Various Database and transaction log corruption error messages containing the terms:

- `Log corrupted`
- `Log entry corrupt`
- `Corrupt database`
- `Database corrupted`

Possible Causes:

You have multiple *independent* `rpc.nisd` daemons running. In normal operation, `rpc.nisd` can spawn other child `rpc.nisd` daemons. This causes no problem. However, if two parent`rpc.nisd` daemons are running at the same time on the same machine, they will overwrite each other's data and corrupt logs and databases. (Normally, this could only occur if someone started running `rpc.nisd` by hand.)

Diagnosis:

Run `ps -e | grep rpc.nisd`. Make sure that you have no more than one parent `rpc.nisd` process.

Solution:

If you have more than one “parent” `rpc.nisd` entry, you must kill all but one of them. Use `svcadm disable` to stop the unwanted services, then run the `ps` command again to make sure the unwanted daemons have died.

If an NIS+ database is corrupt, you will also have to restore it from your most recent backup that contains an uncorrupted version of the database. You can then use the logs to update changes made to your namespace since the backup was recorded. However, if your logs are also corrupted, you will have to recreate by hand any namespace modifications made since the backup was taken.

`rpc.nisd` Fails

If an NIS+ table is too large, `rpc.nisd` may fail.

Diagnosis:

Use `nisls` to check your NIS+ table sizes. Tables larger than 7k may cause rpc.nisd to fail.

Solution:

Reduce the size of large NIS+ tables. Keep in mind that as a naming service NIS+ is designed to store references to objects, not the objects themselves.

NIS+ and NIS Compatibility Problems

This section describes problems having to do with NIS compatibility with NIS+ and earlier systems and the switch configuration file. Common symptoms include:

- The `nsswitch.conf` file fails to perform correctly.
- Error messages with operative clauses.

Error messages with operative clauses include:

- `Unknown user`
- Permission denied
- Invalid principal name

User Cannot Log In After Password Change

Symptoms:

New users, or users who recently changed their password are unable to log in at all, or able to log in on one or more machines but not on others. The user may see error messages with operative clauses such as:

- Unknown user *username*
- `Permission denied`
- `Invalid principal name`

First Possible Cause:

Password was changed on NIS machine.

If a user or system administrator uses the `passwd` command to change a password on a Solaris system machine running NIS in a domain served by NIS+ namespace servers, the user's password is changed only in that machine's `/etc/passwd` file. If the user then goes to some other machine on the network, the user's new password will not be recognized by that machine. The user will have to use the old password stored in the NIS+ passwd table.

Diagnosis:

Check to see if the user's old password is still valid on another NIS+ machine.

Solution:

Use `passwd` on a machine running NIS+ to change the user's password.

Second Possible Cause:

Password changes take time to propagate through the domain.

Diagnosis:

Namespace changes take a measurable amount of time to propagate through a domain and an entire system. This time might be as short as a few seconds or as long as many minutes, depending on the size of your domain and the number of replica servers.

Solution:

You can simply wait the normal amount of time for a change to propagate through your domain(s). Or you can use the `nisping org_dir` command to resynchronize your system.

`nsswitch.conf` File Fails to Perform Correctly

A modified (or newly installed) `nsswitch.conf` file fails to work properly.

Symptoms:

You install a new `nsswitch.conf` file or make changes to the existing file, but your system does not implement the changes.

Possible Cause:

Each time an `nsswitch.conf` file is installed or changed, you must reboot the machine for your changes to take effect. This is because nscd caches the `nsswitch.conf` file.

Solution:

Check your `nsswitch.conf` file against the information contained in the `nsswitch.conf` man page. Correct the file if necessary, and then reboot the machine.

NIS+ Object Not Found Problems

This section describes problem in which NIS+ was unable to find some object or principal. Common symptoms include:

Error messages with operative clauses such as:

- `Not found`
- `Not exist`
- `Can't find suitable transport for`*name*
- `Cannot find`
- `Unable to find`
- `Unable to stat`

Syntax or Spelling Error

The most likely cause of some NIS+ object not being found is that you mistyped or misspelled its name. Check the syntax and make sure that you are using the correct name.

Incorrect Path

A likely cause of an "*object* not found" problem is specifying an incorrect path. Make sure that the path you specified is correct. Also make sure that the `NIS_PATH` environment variable is set correctly.

Domain Levels Not Correctly Specified

Remember that all servers are clients of the domain above them, not the domain they serve. There are two exceptions to this rule:

- The root masters and root replicas are clients of the root domain.
- *NIS+ domain names end with a period.* When using a fully qualified name you must end the domain name with a period. If you do not end the domain name with a period, NIS+ assumes it is a partially qualified name. However, the domain name of a machine should not end with a dot in the `/etc/defaultdomain` file. If you

add a dot to a machine's domain name in the `/etc/defaultdomain` file, you will get `Could not bind to server serving domain` *name* error messages and encounter difficulty in connecting to the net on boot up.

Object Does Not Exist

The NIS+ object may not have been found because it does not exist, either because it has been erased or not yet created. Use `nisls -l` in the appropriate domain to check that the object exists.

Lagging or Out-of-Sync Replica

When you create or modify an NIS+ object, there is a time lag between the completion of your action and the arrival of the new updated information at a given replica. In ordinary operation, namespace information may be queried from a master or any of its replicas. A client automatically distributes queries among the various servers (master and replicas) to balance system load. This means that at any given moment you do not know which machine is supplying you with namespace information. If a command relating to a newly created or modified object is sent to a replica that has not yet received the updated information from the master, you will get an "object not found" type of error or the old out-of-date information. Similarly, a general command such as `nisls` may not list a newly created object if the system sends the `nisls` query to a replica that has not yet been updated.

You can use `nisping` to resync a lagging or out of sync replica server.

Alternatively, you can use the `-M` option with most NIS+ commands to specify that the command must obtain namespace information from the domain's master server. In this way you can be sure that you are obtaining and using the most up-to-date information. (However, you should use the `-M` option only when necessary because a main point of having and using replicas to serve the namespace is to distribute the load and thus increase network efficiency.)

Files Missing or Corrupt

One or more of the files in `/var/nis/data` directory has become corrupted or erased. Restore these files from your most recent backup.

Old `/var/nis` Filenames

In Solaris Release 2.4 and earlier, the `/var/nis` directory contained two files named *hostname*`.dict` and *hostname*`.log`. It also contained a subdirectory named `/var/nis/`*hostname*. Starting with Solaris Release 2.5, the two files were renamed `trans.log` and `data.dict`, and the subdirectory is named `/var/nis/data`.

Do not rename the `/var/nis` or `/var/nis/data` directories or any of the files in these directories that were created by `nisinit` or any of the other NIS+ setup procedures.

In Solaris Release 2.5, the *content* of the files were also changed and they are not backward compatible with Solaris Release 2.4 or earlier. Thus, if you rename either the directories or the files to match the Solaris Release 2.4 patterns, the files will not work with *either* the Solaris Release 2.4 or the Solaris Release 2.5 or later versions of `rpc.nisd`. Therefore, you should not rename either the directories or the files.

Blanks in Name

Symptoms:

Sometimes an object is there, sometimes it is not. Some NIS+ or UNIX commands report that an NIS+ object does not exist or cannot be found, while other NIS+ or UNIX commands do find that same object.

Diagnoses:

Use `nisls` to display the object's name. Look carefully at the object's name to see if the name actually begins with a blank space. (If you accidentally enter two spaces after the flag when creating NIS+ objects from the command line with NIS+ commands, some NIS+ commands will interpret the second space as the beginning of the object's name.)

Solution:

If an NIS+ object name begins with a blank space, you must either rename it without the space or remove it and then recreate it from scratch.

Cannot Use Automounter

Symptoms:

You cannot change to a directory on another host.

Possible Cause:

Under NIS+, automounter names must be renamed to meet NIS+ requirements. NIS+ cannot access `/etc/auto*` tables that contain a period in the name. For example, NIS+ cannot access a file named `auto.direct`.

Diagnosis:

Use `nisls` and `niscat` to determine if the automounter tables are properly constructed.

Solution:

Change the periods to underscores. For example, change `auto.direct` to `auto_direct`. (Be sure to change other maps that might reference these.)

Links To or From Table Entries Do Not Work

You cannot use the `nisln` command (or any other command) to create links between entries in tables. NIS+ commands do not follow links at the entry level.

NIS+ Ownership and Permission Problems

This section describes problems related to user ownership and permissions. Common symptoms include:

Error messages with operative clauses such as:

- `Unable to stat name`
- `Unable to stat NIS+ directory name`
- `Security exception on LOCAL system`
- `Unable to make request`
- `Insufficient permission to . . .`
- `You` *name* `do not have secure RPC credentials`

Another Symptom:

- User or root unable to perform any namespace task.

No Permission

The most common permission problem is the simplest: you have not been granted permission to perform some task that you try to do. Use `niscat -o` on the object in question to determine what permissions you have. If you need additional permission, you, the owner of the object, or the system administrator can either change the permission requirements of the object (as described in Chapter 15) or add you to a group that does have the required permissions (as described in Chapter 17).

No Credentials

Without proper credentials for you and your machine, many operations will fail. Use `nismatch` on your home domain's cred table to make sure you have the right credentials. See "Corrupted Credentials" on page 443 for more on credentials-related problems.

Server Running at Security Level 0

A server running at security level 0 does not create or maintain credentials for NIS+ principals.

If you try to use `passwd` on a server that is running at security level 0, you will get the error message: `You` *name* `do not have secure RPC credentials in NIS+ domain` *domainname*.

Security level 0 is only to be used by administrators for initial namespace setup and testing purposes. Level 0 should not be used in any environment where ordinary users are active.

User Login Same as Machine Name

A user cannot have the same login ID as a machine name. When a machine is given the same name as a user (or vice versa), the first principal can no longer perform operations requiring secure permissions because the second principal's key has overwritten the first principal's key in the cred table. In addition, the second principal now has whatever permissions were granted to the first principal.

For example, suppose a user with the login name of `saladin` is granted namespace read-only permissions. Then a machine named `saladin` is added to the domain. The user `saladin` will no longer be able to perform any namespace operations requiring any sort of permission, and the root user of the machine `saladin` will only have read-only permission in the namespace.

Symptoms:

- The user or machine gets "`permission denied`" type error messages.
- Either the user or root for that machine cannot successfully run `keylogin`.
- `Security exception on LOCAL system. UNABLE TO MAKE REQUEST.` error message.
- If the first principal did not have read access, the second principal might not be able to view otherwise visible objects.

Note – When running `nisclient` or `nisaddcred`, if the message `Changing Key` is displayed rather than `Adding Key`, there is a duplicate user or host name already in existence in that domain.

Diagnosis:

Run `nismatch` to find the host and user in the hosts and passwd tables to see if there are identical host names and user names in the respective tables:

```
nismatch username passwd.org_dir
```

Then run `nismatch` on the domain's cred table to see what type of credentials are provided for the duplicate host or user name. If there are both LOCAL and DES credentials, the cred table entry is for the user; if there is only a DES credential, the entry is for the machine.

Solution:

Change the machine name. (It is better to change the machine name than to change the user name.) Then delete the machine's entry from the `cred` table and use `nisclient` to reinitialize the machine as an NIS+ client. (If you wish, you can use `nistbladm` to create an alias for the machine's old name in the hosts tables.) If necessary, replace the user's credentials in the cred table.

Bad Credentials

See "Corrupted Credentials" on page 443.

NIS+ Security Problems

This section describes common password, credential, encryption, and other security-related problems.

Security Problem Symptoms

Error messages with operative clauses such as:

- `Authentication error`
- `Authentication denied`
- `Cannot get public key`
- `Chkey failed`
- `Insufficient permission to`
- `Login incorrect`
- `Keyserv fails to encrypt`
- `No public key`
- `Permission denied`
- `Password [problems]`

User or root unable to perform any namespace operations or tasks. (See also "NIS+ Ownership and Permission Problems" on page 436.)

`Login Incorrect` Message

The most common cause of a "login incorrect" message is the user mistyping the password. Have the user try it again. Make sure the user knows the correct password and understands that passwords are case-sensitive and also that the letter "o" is not interchangeable with the numeral "0," nor is the letter "l" the same as the numeral "1."

Other possible causes of the "login incorrect" message are:

- The password has been locked by an administrator. See "Locking a Password" on page 300 and "Unlocking a Password" on page 301.
- The password has been locked because the user has exceeded an inactivity maximum. See "Specifying Maximum Number of Inactive Days" on page 307.
- The password has expired. See "Password Privilege Expiration" on page 306.

See Chapter 16 for general information on passwords.

Password Locked, Expired, or Terminated

A common cause of a `Permission denied, password expired`, type message is that the user's password has passed its age limit or the user's password privileges have expired. See Chapter 16 for general information on passwords.

- See "Setting a Password Age Limit" on page 302.
- See "Password Privilege Expiration" on page 306.

Stale and Outdated Credential Information

Occasionally, you may find that even though you have created the proper credentials and assigned the proper access rights, some client requests still get denied. This may be due to out-of-date information residing somewhere in the namespace.

Storing and Updating Credential Information

Credential-related information, such as public keys, is stored in many locations throughout the namespace. NIS+ updates this information periodically, depending on the time-to-live values of the objects that store it, but sometimes, between updates, it gets out of sync. As a result, you may find that operations that should work, don't work. Table 24–2 lists all the objects, tables, and files that store credential-related information and how to reset it.

TABLE 24–2 Where Credential-Related Information is Stored

Item	Stores	To Reset or Change
cred table	NIS+ principal's secret key and public key. These are the master copies of these keys.	Use `nisaddcred` to create new credentials; it updates existing credentials. An alternative is `chkey`.
Directory object	A copy of the public key of each server that supports it.	Run the `/usr/lib/nis/nisupdkeys` command on the directory object.
Keyserver	The secret key of the NIS+ principal that is currently logged in.	Run `keylogin` for a principal user or `keylogin -r` for a principal machine.

TABLE 24–2 Where Credential-Related Information is Stored *(Continued)*

Item	Stores	To Reset or Change
NIS+ daemon	Copies of directory objects, which in turn contain copies of their servers' public keys.	Stop the `rpc.nisd` daemon and the cache manager by disabling the NIS+ service, and then remove `NIS_SHARED_DIRCACHE` from `/var/nis`. Then restart the NIS+ service.
Directory cache	A copy of directory objects, which in turn contain copies of their servers' public keys.	Restart the NIS+ cache manager with the `-i` option.
Cold-start file	A copy of a directory object, which in turn contains copies of its servers' public keys.	Stop the NIS+ service. Remove the `NIS_COLD_START` and `NIS_SHARED_DIRCACHE` files from `/var/nis`. Restart the NIS+ service.
passwd table	A user's password or a machine's superuser password.	Use the `passwd -r nisplus` command. It changes the password in the NIS+ passwd table and updates it in the cred table.
`passwd` file	A user's password or a machine's superuser password.	Use the `passwd -r nisplus` command, whether logged in as superuser or as yourself, whichever is appropriate.
`passwd` map (NIS)	A user's password or a machine's superuser password.	Use `passwd -r nisplus`.

Updating Stale Cached Keys

The most commonly encountered out-of-date information is the existence of stale objects with old versions of a server's public key. You can usually correct this problem by running `nisupdkeys` on the domain you are trying to access. (See Chapter 12 for information on using the `nisupdkeys` command.)

Because some keys are stored in files or caches, `nisupdkeys` cannot always correct the problem. At times you might need to update the keys manually. To do that, you must understand how a server's public key, once created, is propagated through namespace objects. The process usually has five stages of propagation. Each stage is described below.

Stage 1: Server's Public Key Is Generated

An NIS+ server is first an NIS+ client. So, its public key is generated in the same way as any other NIS+ client's public key: with the `nisaddcred` command. The public key is then stored in the cred table of the server's home domain, not of the domain the server will eventually support.

Stage 2: Public Key Is Propagated to Directory Objects

Once you have set up an NIS+ domain and an NIS+ server, you can associate the server with a domain. This association is performed by the `nismkdir` command. When the `nismkdir` command associates the server with the directory, it also copies the server's public key from the cred table to the domain's directory object. For example, assume the server is a client of the `doc.com.` root domain, and is made the master server of the `sales.doc.com.` domain.

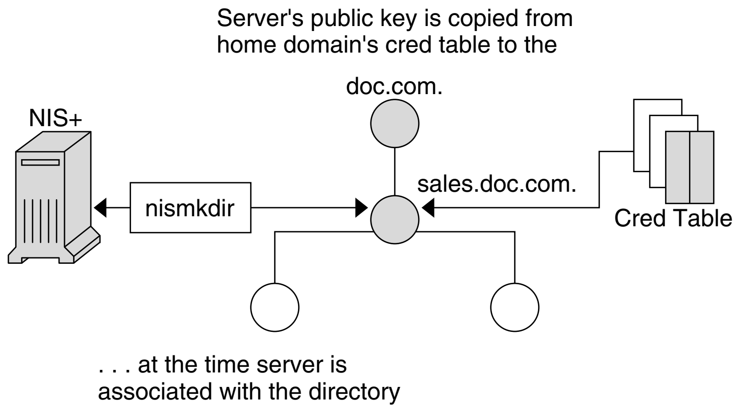

Server's public key is copied from home domain's cred table to the
doc.com.
NIS+
nismkdir
sales.doc.com.
Cred Table
. . . at the time server is associated with the directory

FIGURE 24–1 Public Key is Propagated to Directory Objects

Its public key is copied from the `cred.org_dir.doc.com.` domain and placed in the `sales.doc.com.` directory object. This can be verified with the `niscat -o sales.doc.com.` command.

Stage 3: Directory Objects Are Propagated Into Client Files

All NIS+ clients are initialized with the `nisinit` utility or with the `nisclient` script.

Among other things, `nisinit` (or `nisclient`) creates a cold-start file `/var/nis/NIS_COLDSTART`. The cold-start file is used to initialize the client's directory cache `/var/nis/NIS_SHARED_DIRCACHE`. The cold-start file contains a copy of the directory object of the client's domain. Since the directory object already contains a copy of the server's public key, the key is now propagated into the cold-start file of the client.

In addition when a client makes a request to a server outside its home domain, a copy of the remote domains directory object is stored in the client's `NIS_SHARED_DIRCACHE` file. You can examine the contents of the client's cache by using the `nisshowcache` command, described on page 184.

This is the extent of the propagation until a replica is added to the domain or the server's key changes.

Stage 4: When a Replica is Added to the Domain

When a replica server is added to a domain, the `nisping` command (described in "The `nisping` Command" on page 337) is used to download the NIS+ tables, including the cred table, to the new replica. Therefore, the original server's public key is now also stored in the replica server's cred table.

Stage 5: When the Server's Public Key Is Changed

If you decide to change DES credentials for the server (that is, for the root identity on the server), its public key will change. As a result, the public key stored for that server in the cred table will be different from those stored in:

- The cred table of replica servers (for a few minutes only)
- The main directory object of the domain supported by the server (until its time-to-live expires)
- The `NIS_COLDSTART` and `NIS_SHARED_DIRCACHE` files of every client of the domain supported by server (until their time-to-live expires, usually 12 hours)
- The `NIS_SHARED_DIRCACHE` file of clients who have made requests to the domain supported by the server (until their time-to-live expires)

Most of these locations will be updated automatically within a time ranging from a few minutes to 12 hours. To update the server's keys in these locations immediately, use the commands:

TABLE 24–3 Updating a Server's Keys

Location	Command	See
Cred table of replica servers (instead of using `nisping`, you can wait a few minutes until the table is updated automatically)	`nisping`	"The `nisping` Command" on page 337
Directory object of domain supported by server	`nisupdkeys`	"The `nisupdkeys` Command" on page 250
`NIS_COLDSTART` file of clients	`nisinit -c`	"The `nisinit` Command" on page 333
`NIS_SHARED_DIRCACHE` file of clients	`nis_cachemgr`	"The `nis_cachemgr` Daemon" on page 335

Note – You must stop the existing `nis_cachemgr` before restarting `nis_cachemgr`. To stop `nis_cachemgr`, disable the NIS+ service by using `svcadm disable /network/rpc/nisplus:default`.

Corrupted Credentials

When a principal (user or machine) has a corrupt credential, that principal is unable to perform any namespace operations or tasks, not even `nisls`. This is because a corrupt credential provides no permissions at all, not even the permissions granted to the nobody class.

Symptoms:

User or root cannot perform any namespace tasks or operations. All namespace operations fail with a "permission denied" type of error message. The user or root cannot even perform a `nisls` operation.

Possible Cause:

Corrupted keys or a corrupt, out-of-date, or otherwise incorrect `/etc/.rootkey` file.

Diagnosis:

Use `snoop` to identify the bad credential.

Or, if the principal is listed, log in as the principal and try to run an NIS+ command on an object for which you are sure that the principal has proper authorization. For example, in most cases an object grants read authorization to the nobody class. Thus, the `nisls` *object* command should work for any principal listed in the cred table. If the command fails with a "permission denied" error, then the principal's credential is likely corrupted.

Solution

- *Ordinary user*. Perform a `keylogout` and then a `keylogin` for that principal.
- *Root* or *superuser*. Run `keylogout -f` followed by `keylogin -r`.

`Keyserv` Failure

The `keyserv` is unable to encrypt a session. There are several possible causes for this type of problem:

Possible Causes and Solutions:

- The client has not keylogged in. Make sure that the client is keylogged in. To determine if a client is properly keylogged in, have the client run `nisdefaults -v` (or run it yourself as the client). If `(not authenticated)` is returned on the `Principal Name` line, the client is not properly keylogged in.
- The client (host) does not have appropriate LOCAL or DES credentials. Run `niscat` on the client's cred table to verify that the client has appropriate credentials. If necessary, add credentials as explained in "Creating Credential Information for NIS+ Principals" on page 237.
- The `keyserv` daemon is not running. Use the `ps` command to see if `keyserv` is running. If it is not running, restart `keyserv` and then do a `keylogin`.

- While `keyserv` is running, other long running processes that make secure RPC or NIS+ calls are not. For example, `automountd`, `rpc.nisd`, and `sendmail`. Verify that these processes are running correctly. If they are not, restart them.

Machine Previously Was an NIS+ Client

If this machine has been initialized before as an NIS+ client of the same domain, try `keylogin -r` and enter the root login password at the Secure RPC password prompt.

No Entry in the cred Table

To make sure that an NIS+ password for the principal (user or host) exists in the cred table, run the following command in the principal's home domain

```
nisgrep -A cname=principal cred.org_dir.domainname
```

If you are running `nisgrep` from another domain, the *domainname* must be fully qualified.

Changed Domain Name

Do not change a domain name.

If you change the name of an existing domain you will create authentication problems because the fully qualified original domain name is embedded in objects throughout your network.

If you have *already* changed a domain name and are experiencing authentication problems, or error messages containing terms like "malformed" or "illegal" in relation to a domain name, change the domain name back to its original name. The recommended procedure for renaming your domains is to create a *new* domain with the *new* name, set up your machines as servers and clients of the new domain, make sure they are performing correctly, and then remove the old domain.

When Changing a Machine to a Different Domain

If this machine is an NIS+ client and you are trying to change it to a client of a different domain, remove the `/etc/.rootkey` file, and then rerun the `nisclient` script using the network password supplied by your network administrator or taken from the `nispopulate` script.

NIS+ and Login Passwords in `/etc/passwd` File

Your NIS+ password is stored in the NIS+ passwd table. Your user login password may be stored in NIS+ passwd table or in your `/etc/passwd` file. (Your user password and NIS+ password can be the same or different.) To change a password in an `/etc/passwd` file, you must run the `passwd` command with the `nsswitch.conf` file set to files or with the `-r files` flag.

The `nsswitch.conf` file specifies which password is used for which purpose. If the `nsswitch.conf` file is directing system queries to the wrong location, you will get password and permission errors.

Secure RPC Password and Login Passwords Are Different

When a principal's login password is different from his or her Secure RPC password, `keylogin` cannot decrypt it at login time because `keylogin` defaults to using the principal's login password, and the private key was encrypted using the principal's Secure RPC password.

When this occurs the principal can log in to the system, but for NIS+ purposes is placed in the authorization class of nobody because the keyserver does not have a decrypted private key for that user. Since most NIS+ environments are set up to deny the nobody class create, destroy, and modify rights to most NIS+ objects this results in "permission denied" types errors when the user tries to access NIS+ objects.

Note – In this context network password is sometimes used as a synonym for Secure RPC password. When prompted for your "network password," enter your Secure RPC password.

To be placed in one of the other authorization classes, a user in this situation must explicitly run the `keylogin` program and give the principal's Secure RPC password when `keylogin` prompts for password. (See "Keylogin With NIS+" on page 244.)

But an explicit `keylogin` provides only a temporary solution that is good only for the current login session. The keyserver now has a decrypted private key for the user, but the private key in the user's cred table is still encrypted using the user's Secure RPC password, which is different than the user's login password. The next time the user logs in, the same problem reoccurs. To permanently solve the problem the user needs to change the private key in the cred table to one based on the user's login ID rather than the user's Secure RPC password. To do this, the user need to run the `chkey` program as described in "Changing Keys for an NIS+ Principal" on page 245.

Thus, to permanently solve a Secure RPC password different than login password problems, the user (or an administrator acting for the user) must perform the following steps:

1. Log in using the login password.
2. Run the `keylogin` program to temporarily get a decrypted private key stored in the keyserver and thus gain temporary NIS+ access privileges.
3. Run `chkey -p`to permanently change the encrypted private key in the cred table to one based on the user's login password.

Preexisting `/etc/.rootkey` File

Symptoms:

Various `insufficient permission to` and `permission denied` error messages.

Possible Cause:

An `/etc/.rootkey` file already existed when you set up or initialized a server or client. This could occur if NIS+ had been previously installed on the machine and the `.rootkey` file was not erased when NIS+ was removed or the machine returned to using NIS or `/etc` files.

Diagnosis:

Run `ls -l` on the `/etc` directory and `nisls -l org_dir` and compare the date of the `/etc/.rootkey` to the date of the cred table. If the `/etc/.rootkey` date is clearly earlier than that of the cred table, it may be a preexisting file.

Solution:

Run `keylogin -r` as root on the problem machine and then set up the machine as a client again.

Root Password Change Causes Problem

Symptoms:

You change the root password on a machine, and the change either fails to take effect or you are unable to log in as superuser.

Possible Cause:

Note – For security reasons, you should not have User ID 0 listed in the `passwd` table.

You changed the root password, but root's key was not properly updated. Either because you forgot to run `chkey -p` for root or some problem came up.

Solution

Log in as a user with administration privileges (that is, a user who is a member of a group with administration privileges) and use `passwd` to restore the old password. Make sure that old password works. Now use `passwd` to change root's password to the new one, and then run `chkey -p`.

Caution – Once your NIS+ namespace is set up and running, you can change the root password on the root master machine. But do not change the root master keys, as these are embedded in all directory objects on all clients, replicas, and servers of subdomains. To avoid changing the root master keys, always use the `-p` option when running `chkey` as root.

NIS+ Performance and System Hang Problems

This section describes common slow performance and system hang problems.

Performance Problem Symptoms

Error messages with operative clauses such as:

- `Busy try again later`
- `Not responding`

Other common symptoms:

- You issue a command and nothing seems to happen for far too long.
- Your system, or shell, no longer responds to keyboard or mouse commands.
- NIS+ operations seem to run slower than they should or slower than they did before.

Checkpointing Operations

Someone has issued an `nisping` or `nisping -C` command. Or the `rpc.nisd` daemon is performing a checkpoint operation.

Caution – Do not reboot! Do not issue any more `nisping` commands.

When performing a `nisping` or checkpoint, the server will be sluggish or may not immediately respond to other commands. Depending on the size of your namespace, these commands may take a noticeable amount of time to complete. Delays caused by checkpoint or ping commands are multiplied if you, or someone else, enter several such commands at one time. Do not reboot. This kind of problem will solve itself. Just wait until the server finishes performing the `nisping` or checkpoint command.

During a full master-replica resync, the involved replica server will be taken out of service until the resync is complete. Do not reboot—just wait.

Variable NIS_PATH

Make sure that your NIS_PATH variable is set to something clean and simple. For example, the default: org_dir.$:$. A complex NIS_PATH, particularly one that itself contains a variable, will slow your system and may cause some operations to fail. (See "NIS_PATH Environment Variable" on page 74 for more information.)

Do not use nistbladm to set nondefault table paths. Nondefault table paths will slow performance.

Table Paths

Do not use table paths because they will slow performance.

Too Many Replicas

Too many replicas for a domain degrade system performance during replication. There should be no more than 10 replicas in a given domain or subdomain. If you have more than five replicas in a domain, try removing some of them to see if that improves performance.

Recursive Groups

A recursive group is a group that contains the name of some other group. While including other groups in a group reduces your work as system administrator, doing so slows down the system. You should not use recursive groups.

Large NIS+ Database Logs at Start-up

When rpc.nisd starts up it goes through each log. If the logs are long, this process could take a long time. If your logs are long, you may want to checkpoint them using nisping -C before starting rpc.nisd.

The Master rpc.nisd Daemon Died

Symptoms:

If you used the -M option to specify that your request be sent to the master server, and the rpc.nisd daemon has died on that machine, you will get a "server not responding" type error message and no updates will be permitted. (If you did not use the -M option, your request will be automatically routed to a functioning replica server.)

Possible Cause:

Using uppercase letters in the name of a home directory or host can sometimes cause `rpc.nisd` to die.

Diagnosis:

First make sure that the server itself is up and running. If it is, run `ps -ef | grep rpc.nisd` to see if the daemon is still running.

Solution:

If the daemon has died, restart the NIS+ service by using `svcadm enable /network/rpc/nisplus:default`. If `rpc.nisd` frequently dies, contact your service provider.

No `nis_cachemgr`

Symptoms:

It takes too long for a machine to locate namespace objects in other domains.

Possible Cause:

You do not have `nis_cachemgr` running.

Diagnosis:

Run `ps -ef | grep nis_cachemgr` to see if it is still running.

Solution

Start `nis_cachemgr` on that machine by enabling the NIS+ service.

Server Very Slow at Start-up After NIS+ Installation

Symptoms:

A server performs slowly and sluggishly after using the NIS+ scripts to install NIS+ on it.

Possible Cause:

You forgot to run `nisping -C -a` after running the `nispopulate` script.

Solution:

Run `nisping -C -a` to checkpoint the system as soon as you are able to do so.

`niscat` Returns: `Server busy. Try Again`

Symptoms:

You run `niscat` and get an error message indicating that the server is busy.

Possible Cause:

- The server is busy with a heavy load, such as when doing a resync.
- The server is out of swap space.

Diagnosis:

Run `swap -s` to check your server's swap space.

Solution:

You must have adequate swap and disk space to run NIS+. If necessary, increase your space.

NIS+ Queries Hang After Changing Host Name

Symptoms:

Setting the host name for an NIS+ server to be fully qualified is not recommended. If you do so, and NIS+ queries then just hang with no error messages, check the following possibilities:

Possible Cause:

Fully qualified host names must meet the following criteria:

- The domain part of the host name must be the same as the name returned by the `domainname` command.
- After the setting the host name to be fully qualified, you must also update all the necessary `/etc` and `/etc/inet` files with the new host name information.
- The host name must end in a period.

Solution:

Stop the NIS+ service and any associated processes that are hanging on that host or server. Rename the host to match the two requirements listed above. Then reinitialize the server with `nisinit`. (If queries still hang after you are sure that the host is correctly named, check other problem possibilities in this section.)

NIS+ System Resource Problems

This section describes problems having to do with lack of system resources such as memory, disk space, and so forth.

Resource Problem Symptoms

Error messages with operative clauses such as:

- `No memory`
- `Out of disk space`
- "Cannot [do something] with log" type messages
- `Unable to fork`

Insufficient Memory

Lack of sufficient memory or swap space on the system you are working with will cause a wide variety of NIS+ problems and error messages. As a short-term, temporary solution, try to free additional memory by killing unneeded windows and processes. If necessary, exit your windowing system and work from the terminal command line. If you still get messages indicating inadequate memory, you will have to install additional swap space or memory, or switch to a different system that has enough swap space or memory.

Under some circumstances, applications and processes may develop memory leaks and grow too large. you can check the current size of an application or process by running:

```
ps -el
```

The `sz` (size) column shows the current memory size of each process. If necessary, compare the sizes with comparable processes and applications on a machine that is not having memory problems to see if any have grown too large.

Insufficient Disk Space

Lack of disk space will cause a variety of error messages. A common cause of insufficient disk space is failure to regularly remove `tmp` files and truncate `log` files. `log` and `tmp` files grow steadily larger unless truncated. The speed at which these files grow varies from system to system and with the system state. `log` files on a system that is working inefficiently or having namespace problems will grow very fast.

Note – If you are doing a lot of troubleshooting, check your `log` and `tmp` files frequently. Truncate `log` files and remove `tmp` files before lack of disk space creates additional problems. Also check the root directory and home directories for core files and delete them.

The way to truncate `log` files is to regularly checkpoint your system (Keep in mind that a checkpoint process may take some time and will slow down your system while it is being performed, checkpointing also requires enough disk space to create a complete copy of the files before they are truncated.)

To checkpoint a system, run `nisping -C`.

Insufficient Processes

On a heavily loaded machine it is possible that you could reach the maximum number of simultaneous processes that the machine is configured to handle. This causes messages with clauses like "unable to fork". The recommended method of handling this problem is to kill any unnecessary processes. If the problem persists, you can reconfigure the machine to handle more processes as described in your system administration documentation.

NIS+ User Problems

This section describes NIS+ problems that a typical user might encounter.

User Problem Symptoms

- User cannot log in.
- User cannot `rlogin` to other domain.

User Cannot Log In

There are many possible reasons for a user being unable to log in:

- *User forgot password*. To set up a new password for a user who has forgotten the previous one, run `passwd` for that user on another machine (naturally, you have to be the NIS+ administrator to do this).
- *Mistyping password*. Make sure the user knows the correct password and understands that passwords are case-sensitive and that the letter "o" is not interchangeable with the numeral "0," nor is the letter "l" the same as the numeral "1."
- *"Login incorrect" type message*. For causes other than simply mistyping the password, see "`Login Incorrect` Message" on page 438.
- The user's password privileges have expired (see "Password Privilege Expiration" on page 306).
- An inactivity maximum has been set for this user, and the user has passed it (see "Specifying Maximum Number of Inactive Days" on page 307).
- The user's `nsswitch.conf` file is incorrect. The `passwd` entry in that file must be one of the following five permitted configurations:
  - `passwd: files`

- `passwd: files nis`
- `passwd: files nisplus`
- `passwd: compat`
- `passwd: compat passwd_compat: nisplus`

  Any other configuration will prevent a user from logging in.

(See "`nsswitch.conf` File Requirements" on page 290 for further details.)

User Cannot Log In Using New Password

Symptoms:

Users who recently changed their password are unable to log in at all, or are able to log in on some machines but not on others.

Possible Causes:

- It may take some time for the new password to propagate through the network. Have users try to log in with the old password.
- The password was changed on a machine that was not running NIS+ (see "User Cannot Log In Using New Password" on page 453).

User Cannot Remote Log In to Remote Domain

Symptoms:

User tries to `rlogin` to a machine in some other domain and is refused with a "Permission denied" type error message.

Possible Cause:

To `rlogin` to a machine in another domain, a user must have LOCAL credentials in that domain.

Diagnosis:

Run `nismatch` *username.domainname*. `cred.org_dir` in the other domain to see if the user has a LOCAL credential in that domain.

Solution:

Go to the remote domain and use `nisaddcred` to create a LOCAL credential for the user in the that domain.

User Cannot Change Password

The most common cause of a user being unable to change passwords is that the user is mistyping (or has forgotten) the old password.

Other possible causes:

- The password Min value has been set to be greater than the password Max value. See "Setting Minimum Password Life" on page 303.
- The password is locked or expired. See "`Login Incorrect` Message" on page 438 and "Password Locked, Expired, or Terminated" on page 439.

Other NIS+ Problems

This section describes problems that do not fit any of the previous categories.

How to Tell if NIS+ Is Running

You may need to know whether a given host is running NIS+. A script may also need to determine whether NIS+ is running.

There are two methods for checking if NIS+ is running.

- By using the `svcs \*nisplus\*` command to verify that NIS+ is online
- By verifying the following items
  - `nis_cachemgr` is running.
  - The host has a `/var/nis/NIS_COLD_START` file.
  - `nisls` succeeds.

Replica Update Failure

Symptoms:

Error messages indicating that the update was not successfully complete. (Note that the message: `replica_update:` *number* `updates` *number* `errors` indicates a successful update.)

Possible Causes:

Any of the following error messages indicate that the server was busy and that the update should be rescheduled:

- Master server busy, full dump rescheduled
- `replica_update` error result was Master server busy full dump rescheduled, full dump rescheduled
- `replica_update:` master server busy, rescheduling the resync
- `replica_update:` master server busy, will try later
- `replica_update:` nis dump result Master server busy, full dump rescheduled

- `nis_dump_svc:` one replica is already resyncing

(These messages are generated by, or in conjunction with, the NIS+ error code constant: `NIS_DUMPLATER` one replica is already resyncing.)

These messages indicate that there was some other problem:

- `replica_update:` error result was ...
- `replica_update:` nis dump result `nis_perror` error string
- `rootreplica_update:` update failednis dump result `nis_perror` string-variable: could not fetch object from master

(If `rpc.nisd` is being run with the `-C` (open diagnostic channel) option, additional information may be entered in either the master server or replica server's system log.

These messages indicate possible problems such as:

- The server is out of child processes that can be allocated.
- A read-only child process was requested to dump.
- Another replica is currently resynching.

Diagnosis:

Check both the replica and server's system log for additional information. How much, if any, additional information is recorded in the system logs depends on your system's error reporting level, and whether or not you are running `rpc.nisd` with the `-C` option (diagnostics).

Solution:

In most cases, these messages indicate minor software problems which the system is capable of correcting. If the message was the result of a command, simply wait for a while and then try the command again. If these messages appear often, you can change the threshold level in your `/etc/syslog.conf` file. See the `syslog.conf` man page for details.

APPENDIX **A**

Error Messages

This appendix alphabetically lists some common error messages for the DNS, NIS and NIS+ naming services in the Solaris software. For each message there is an explanation and, where appropriate, a solution or a cross-reference to some other portion of this manual.

About Error Messages

Some of the error messages documented in this chapter are documented more fully in the appropriate man pages.

Error Message Context

Error messages can appear in pop-up windows, shell tool command lines, user console window, or various log files. You can raise or lower the severity threshold level for reporting error conditions in your `/etc/syslog.conf` file.

In the most cases, the error messages that you see are generated by the commands you issued or the container object (file, map, table or directory) your command is addressing. However, in some cases an error message might be generated by a server invoked in response to your command (these messages usually show in `syslog`). For example, a "`permission denied`" message most likely refers to you, or the machine you are using, but it could also be caused by software on a server not having the correct permissions to carry out some function passed on to it by your command or your machine.

Similarly, some commands cause a number of different objects to be searched or queried. Some of these objects might not be obvious. Any one of these objects could return an error message regarding permissions, read-only state, unavailability, and so forth. In such cases the message might not be able to inform you of which object the problem occurred in.

In normal operation, the naming software and servers make routine function calls. Sometimes those calls fail and in doing so generate an error message. It occasionally happens that before a client or server processes your most recent command, then some other call fails and you see the resulting error message. Such a message might appear as if it were in response to your command, when in fact it is in response to some other operation.

Note – When working with a namespace you might encounter error messages generated by remote procedure calls. These RPC error messages are not documented here. Check your system documentation.

Context-Sensitive Meanings

A single error message might have slightly different meanings depending on which part of various naming software applications generated the message. For example, when a "`Not Found`" type message is generated by the `nisls` command, it means that there are no NIS+ objects that have the specified name, but when it is generated by the `nismatch` command it means that no table entries were found that meet the search criteria.

How Error Messages Are Alphabetized

The error messages in this appendix are sorted alphabetically according to the following rules:

- Capitalization is ignored. Thus, messages that begin with "A" and "a" are alphabetized together.
- Nonalphabetic symbols are ignored. Thus, a message that begins with `_svcauth_des` is listed with the other messages that begin with the letter "S".
- Error messages beginning with (or containing) the word *NIS+* are alphabetized after messages beginning with (or containing) the word *NIS*.
- Some error messages might be preceded by a date or the name of the host, application, program, or routine that generated the error message, followed by a colon. In these cases, the initial name of the command is used to alphabetize the message.
- Many messages contain variables such as user IDs, process numbers, domain names, host names, and so forth. In this appendix, these variables are indicated by an *italic typeface*. Because variables could be anything, they are not included in the

sorting of the messages listed in this appendix. For example, the actual message `sales: is not a table` (where `sales` is a variable) would be listed in this appendix as: *name*: is not a table and would be alphabetized as: `is not a table` among those messages beginning with the letter "I".

- Error messages that begin with asterisks, such as **ERROR: *domainname* does not exist, are generated by the NIS+ installation and setup scripts. Messages are alphabetized according to their first letter, ignoring the asterisks.

Numbers in Error Messages

- Many messages include an IP address. IP addresses are indicated by *n.n.n.n*.
- Some error messages include numbers such as process ID numbers, number of items, and so forth. Numbers in error messages are indicated: *nnnn*.

Common Namespace Error Messages

`abort_transaction: Failed to action NIS+` *objectname*

The `abort_transaction` routine failed to back out of an incomplete transaction due to a server crash or some other unrecoverable error. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for further information.

`abort_transaction: Internal database error abort_transaction: Internal error, log entry corrupt NIS+` *objectname*

These two messages indicate some form of corruption in a namespace database or log. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information.

`add_cleanup: Cant allocate more rags.`

This message indicates that your system is running low on available memory. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on insufficient memory problems.

`add_pingitem: Couldn't add` *directoryname* `to pinglist (no memory)`

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on low memory problems.

`add_update: Attempt add transaction from read only child.`
`add_update Warning: attempt add transaction from read only child`

An attempt by a read-only child `rpc.nisd` process to add an entry to a log. An occasional appearance of this message in a log is not serious. If this message appears frequently, contact the Sun Solutions Center.

`Attempting to free a free rag!`

This message indicates a software problem with `rpc.nisd`. The `rpc.nisd` should have aborted. Run `ps -ef | grep rpc.nisd` to see if `rpc.nisd` is still running. If it is, stop the NIS+ service and restart it with the same options as previously used. If the daemon is not running, start the NIS+ service with the same options as previously used. Check `/var/nis` to see if a `core` file has been dumped. If there is a `core` file, delete it.

Note – If you started the NIS+ service with the `-YB` options, you must also kill the `rpc.nisd_reply` daemon.

`Attempt to remove a non-empty table`

An attempt has been made by `nistbladm` to remove an NIS+ table that still contains entries. Or by `nisrmdir` to remove a directory that contains files or subdirectories.

- If you are trying to delete a table, use `niscat` to check the contents of the table and `nistbladm` to delete any existing contents.
- If you are trying to delete a directory, use `nisls -l -R` to check for existing files or subdirectories and delete them first.
- If you are trying to dissociate a replica from a domain with `nisrmdir -s`, and the replica is down or otherwise out of communication with the master, you will get this error message. In such cases, you can run `nisrmdir -f -s` *replicaname* on the master to force the dissociation. Note, however, that if you use `nisrmdir -f -s`to dissociate an out-of-communication replica, you *must* run `nisrmdir -f -s` *again* as soon as the replica is back on line in order to clean up the replica's `/var/nis` file system. If you fail to rerun `nisrmdir -f -s` *replicaname* when the replica is back in service, the old out-of-date information left on the replica could cause problems.

This message is generated by the NIS+ error code constant: `NIS_NOTEMPTY`. See the `nis_tables` man page for additional information.

`authdes_marshal: DES encryption failure`

DES encryption for some authentication data failed. Possible causes:

- Corruption of a library function or argument.
- A problem with a DES encryption chip, if you are using one.

Call the Sun Solutions Center for assistance.

`authdes_refresh: keyserv is unable to encrypt session key`

The `keyserv` process was unable to encrypt the session key with the public key that it was given. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information.

`authdes_refresh: unable to encrypt conversation key`

The `keyserv` process could not encrypt the session key with the public key that was given. This usually requires some action on your part. Possible causes are:

- The `keyserv` process is dead or not responding. Use `ps -ef` to check whether the `keyserv` process is running on the `keyserv` host. If it is not, then start it, and then run `keylogin`.
- The client has not performed a `keylogin`. Do a `keylogin` for the client and see if that corrects the problem.
- The client host does not have credentials. Run `nismatch` on the client's home domain cred table to see if the client host has the proper credentials. If it does not, create them.
- A DES encryption failure. See the authdes_marshal: DES encryption failure error message).

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information regarding security key problems.

`authdes_refresh: unable to synchronize clock`

This indicates a synchronization failure between client and server clocks. This will usually correct itself. However, if this message is followed by any time stamp related error, you should manually resynchronize the clocks. If the problem reoccurs, check that remote `rpcbind` is functioning correctly.

`authdes_refresh: unable to synch up w/server`

The client-server clock synchronization has failed. This could be caused by the `rpcbind` process on the server not responding. Use `ps -ef` on the server to see if `rpcbind` is running. If it is not, restart it. If this error message is followed by any time stamp-related message, then you need to use `rdate` *servername* to manually resync the client clock to the server clock.

`authdes_seccreate: keyserv is unable to generate session key`

This indicates that `keyserv` was unable to generate a random DES key for this session. This requires some action on your part:

- Check to make sure that `keyserv` is running properly. If it is not, restart it along with all other long-running processes that use Secure RPC or make NIS+ calls such as `automountd`, `rpc.nisd` and `sendmail`. Then do a `keylogin`.
- If `keyserv` is up and running properly, restart the process that logged this error.

`authdes_seccreate: no public key found for` *servername*

The client side cannot get a DES credential for the server named *servername*. This requires some action on your part:

- Check to make sure that *servername* has DES credentials. If it does not, create them.

- Check the switch configuration file to see which naming service is specified and then make sure that service is responding. If it is not responding, restart it.

`authdes_seccreate: out of memory`

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on insufficient memory problems.

`authdes_seccreate: unable to gen conversation key`

The `keyserv` process was unable to generate a random DES key. The most likely cause is that the `keyserv` process is down or otherwise not responding. Use `ps -ef` to check whether the `keyserv` process is running on the `keyserv` host. If it is not, then start it and run `keylogin`.

If restarting `keyserv` fails to correct the problem, it might be that other processes that use Secure RPC or make NIS+ calls are not running (for example, `automountd`, `rpc.nisd`, or `sendmail`). Check to see whether these processes are running; if they are not, restart them.

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information regarding security key problems.

`authdes_validate: DES decryption failure`

See authdes_marshal: DES decryption failure for authentication data failure.

`authdes_validate: verifier mismatch`

The time stamp that the client sent to the server does not match the one received from the server. (This is not recoverable within a Secure RPC session.) Possible causes:

- Corruption of the session key or time stamp data in the client or server cache.
- Server deleted from this cache a session key for a still active session.
- Network data corruption.

Try re-executing the command.

`CacheBind: xdr_directory_obj failed.`

The most likely causes for this message are:

- Bad or incorrect parameters being passed to the xdr_directory_obj routine. Check the syntax and accuracy of whatever command you most recently entered.
- An attempt to allocate system memory failed. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for a discussion of memory problems.
- If your command syntax is correct, and your system does not seem to be short of memory, contact the Sun Solutions Center.

`Cache expired`

The entry returned came from an object cache that has expired. This means that the time-to-live value has gone to zero and the entry might have changed. If the flag `-NO_CACHE` was passed to the lookup function, then the lookup function will retry the operation to get an unexpired copy of the object.

This message is generated by the NIS+ error code constant: `NIS_CACHEEXPIRED`. See the `nis_tables` and `nis_names` man pages for additional information.

`Callback: - select failed` *message nnnn*

An internal system call failed. In most cases this problem will correct itself. If it does not correct itself, make sure that `rpc.nisd` has not been aborted. If it has, restart it. If the problem reoccurs frequently, contact the Sun Solutions Center.

`CALLBACK_SVC: bad argument`

An internal system call failed. In most cases this problem will correct itself. If it does not correct itself, make sure that `rpc.nisd` has not been aborted. If it has, restart it. If the problem reoccurs frequently, contact the Sun Solutions Center.

`Cannot grow transaction log error` *string*

The system cannot add to the log file. The reason is indicated by the *string*. The most common cause of this message is lack of disk space. See “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

`Cannot truncate transaction log file`

An attempt has been made to checkpoint the log, and the `rpc.nisd` daemon is trying to shrink the log file after deleting the checkpointed entries from the log. See the `ftruncate` man pages for a description of various factors that might cause this routine to fail. See also “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

`Cannot write one character to transaction log, error`*message*

An attempt has been made by the `rpc.nisd` daemon to add an update from the current transaction into the transaction log, and the attempt has failed for the reason given in the *message* that has been returned by the function. Additional information can be obtained from the `write` routine’s man page.

`Can’t compile regular expression` *variable*

Returned by the `nisgrep` command when the expression in `keypat` was malformed.

`Can’t get any map parameter information.`

See “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

`Can’t find name service for passwd`

Either there is no `nsswitch.conf` file or there is no `passwd` entry in the file, or the `passwd` entry does not make sense or is not one of the allowed formats.

`Can't find` *name* `'s secret key`

Possible causes:

- You might have incorrectly typed the password.
- There might not be an entry for *name* in the `cred` table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt).
- The `nsswitch.conf` file might be directing the query to a local password in an `/etc/passwd` file that is different than the NIS+ password recorded in the cred table.

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on diagnosing and solving these type of problem.

`checkpoint_log: Called from read only child ignored.`

This is a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`checkpoint_log: Unable to checkpoint, log unstable.`

An attempt was made to checkpoint a log that was not in a stable state. (That is, the log was in a resync, update, or checkpoint state.) Wait until the log is stable, and then rerun the `nisping` command.

`check_updaters: Starting resync.`

This is a system status message. No action need be taken.

`Child process requested to checkpoint!`

This message indicates a minor software problem that the system is capable of correcting. If these messages appear often, you can change the threshold level in your `/etc/syslog.conf` file. See the `syslog.conf` man page for details.

`Column not found:` *columnname*

The specified column does not exist in the specified table.

`Could not find` *string* `'s secret key`

Possible causes:

- You might have incorrectly typed the password.
- There might not be an entry for `name` in the `cred` table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt)
- The `nsswitch.conf` file might have the wrong publickey policy. It might be directing the query to a local public key in an `/etc/publickey` file that is different from the NIS+ password recorded in the cred table.

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on diagnosing and solving these types of problem.

`Could not generate netname`

The Secure RPC software could not generate the Secure RPC netname for your UID when performing a `keylogin`. This could be due to the following causes:

- You do not have LOCAL credentials in the NIS+ cred table of the machine's home domain.
- You have a local entry in `/etc/passwd` with a UID that is different from the UID you have in the NIS+ passwd table.

string`: could not get secret key for '`*string*

Possible causes:

- You might have incorrectly typed the password.
- There might not be an entry for `name` in the cred table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt)
- The `nsswitch.conf` file might have the wrong publickey policy. It might be directing the query to a local publickey in an `/etc/publickey` file that is different from the NIS+ password recorded in the cred table.

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on diagnosing and solving these type of problem.

`Couldn't fork a process!`

The server could not fork a child process to satisfy a callback request. This is probably caused by your system reaching its maximum number of processes. You can kill some unneeded processes, or increase the number of processes your system can handle. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information.

`Couldn't parse access rights for column` *string*

This message is usually returned by the `nistbladm -u` command when something other than a `+` (plus sign), a `-` (minus sign), or an `=` (equal sign) is entered as the operator. Other possible causes are failure to separate different column rights with a comma, or the entry of something other than `r`, `d`, `c`, or `m` for the type of permission. Check the syntax for this type of entry error. If everything is entered correctly and you still get this error, the table might have been corrupted.

`Database for table does not exist`

At attempt to look up a table has failed. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for possible causes.

This message is generated by the NIS+ error code constant: `NIS_NOSUCHTABLE`. See the `nis_tables` and `nis_names` man pages for additional information.

`_db_add: child process attempting to add/modify _db_addib: non-parent process attempting an add`

These messages indicate that a read-only or nonparent process attempted to add or modify an object in the database. In most cases, these messages do not require any action on your part. If these messages are repeated frequently, call the Sun Solutions Center.

`db_checkpoint: Unable to checkpoint` *string*

This message indicates that for some reason NIS+ was unable to complete checkpointing of a directory. The most likely cause is that the disk is full See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information).

`_db_remib: non-parent process attempting an remove _db_remove: non-parent process attempting a remove`

These messages indicate that a read-only or non-parent process attempted to remove a table entry. In most cases, these messages do not require any action on your part. If these messages are repeated frequently, call the Sun Solutions Center.

`Do you want to see more information on this command?`

This indicates that there is a syntax or spelling error on your script command line.

`Entry/Table type mismatch`

This occurs when an attempt is made to add or modify an entry in a table, and the entry passed is of a different type from the table. For example, if the number of columns is not the same. Check that your update correctly matches the table type.

This message is generated by the NIS+ error code constant: `NIS_TYPEMISMATCH`. See the `nis_tables` man page for additional information.

`**ERROR: chkey failed again. Please contact your network administrator to verify your network password.`

This message indicates that you typed the wrong network password.

- If this is the first time you are initializing this machine, contact your network administrator to verify the network password.
- If this machine has been initialized before as an NIS+ client of the same domain, try typing the root login password at the Secure RPC password prompt.
- If this machine is currently an NIS+ client and you are trying to change it to a client of a different domain, remove the `/etc/.rootkey` file, and rerun the `nisclient` script, using the network password given to you by your network administrator (or the network password generated by the `nispopulate` script).

`Error: Could not create a valid NIS+ coldstart file`

This message is from `nisinit`, the NIS+ initialization routine. It is followed by another message preceded by a string that begins: "`lookup:..`". This second message will explain why a valid NIS+ cold-start file could not be created.

`**ERROR: could not restore file` *filename*

This message indicates that NIS+ was unable to rename *filename*`.no_nisplus` to *filename*.

Check your system console for system error messages.

- If there is a system error message, fix the problem described in the error message and rerun `nisclient -i`.
- If there aren't any system error messages, try renaming this file manually, and then rerun `nisclient -i`.

`**ERROR: Couldn't get the server NIS+_server's address.`

The script was unable to retrieve the server's IP address for the specified domain. Manually add the IP address for *NIS+_server* into the `/etc/hosts` or `/etc/inet/ipnodes` file, then rerun `nisclient -i`.

`**ERROR: directory` *directory-path* `does not exist.`

This message indicates that you typed an incorrect directory path. Type the correct directory path.

`**ERROR:` *domainname* `does not exist.`

This message indicates that you are trying to replicate a domain that does not exist.

- If *domainname* is spelled incorrectly, rerun the script with the correct domain name.
- If the *domainname* domain does not exist, create it. Then you can replicate it.

`**ERROR:` *parent-domain* `does not exist.`

This message indicates that the parent domain of the domain you typed on the command line does not exist. This message should only appear when you are setting up a nonroot master server.

- If the domain name is spelled incorrectly, rerun the script with the correct domain name.
- If the domain's parent domain does not exist, you have to create the parent domain first, and then you can create this domain.

`**ERROR: Don't know about the domain "`*domainname*`". Please check your` *domainname*`.`

This message indicates that you typed an unrecognized domain name. Rerun the script with the correct domain name.

`**ERROR: failed dumping` *tablename* `table.`

The script was unable to populate the cred table because the script did not succeed in dumping the named table.

- If `niscat` *tablename* `.org_dir` fails, make sure that all the servers are operating, then rerun the script to populate the *tablename* table.

- If `niscat` *tablename*`.org_dir` is working, the error might have been caused by the NIS+ server being temporarily busy. Rerun the script to populate the *tablename* table.

`**ERROR: host` *hostname* `is not a valid NIS+ principal in domain` *domainname*`. This host name must be defined in the credential table in domain` *domainname*`. Use nisclient -c to create the host credential`

A machine has to be a valid NIS+ client with proper credentials before it can become an NIS+ server. To convert a machine to an NIS+ root replica server, the machine first must be an NIS+ client in the root domain. Follow the instructions on how to add a new client to a domain, then rerun `nisserver -R`.

Before you can convert a machine to an NIS+ nonroot master or a replica server, the machine must be an NIS+ client in the parent domain of the domain that it plans to serve. Follow the instructions on how to add a new client to a domain, then rerun `nisserver -M` or `nisserver -R`.

This problem should not occur when you are setting up a root master server.

`Error in accessing NIS+ cold start file is NIS+ installed?`

This message is returned if NIS+ is not installed on a machine or if for some reason the file `/var/nis/NIS_COLD_START` could not be found or accessed. Check to see if there is a `/var/nis/NIS_COLD_START` file. If the file exists, make sure your path is set correctly and that `NIS_COLD_START` has the proper permissions. Then rename or remove the old cold-start file and rerun the `nisclient` script to install NIS+ on the machine.

This message is generated by the cache manager that sends the NIS+ error code constant: `NIS_COLDSTART_ERR`. See the `write` and `open` man pages for additional information on why a file might not be accessible.

`Error in RPC subsystem`

This fatal error indicates the RPC subsystem failed in some way. Generally, there will be a `syslog` message on either the client or server side indicating why the RPC request failed.

This message is generated by the NIS+ error code constant: `NIS_RPCERROR`. See the `nis_tables` and `nis_names` man pages for additional information.

`**ERROR: it failed to add the credential for root.`

The NIS+ command `nisaddcred` failed to create the root credential when trying to set up a root master server. Check your system console for system error messages:

- If there is a system error message, fix the problem described in the error message and then rerun `nisserver`.
- If there aren't any system error messages, check to see whether the `rpc.nisd` process is running. If it is not running, restart it and then rerun `nisserver`.

`**ERROR: it failed to create the tables.`

The NIS+ command `nissetup` failed to create the directories and tables. Check your system console for system error messages:

- If there is a system error message, fix the problem described in the error message and rerun `nisserver`.
- If there aren't any system error messages, check to see whether the `rpc.nisd` process is running. If it is not running, restart it and rerun `nisserver`.

`**ERROR: it failed to initialize the root server.`

The NIS+ command `nisinit -r` failed to initialize the root master server. Check your system console for system error messages. If there is a system error message, fix the problem described in the error message and rerun `nisserver`.

`**ERROR: it failed to make the` *domainname* `directory`

The NIS+ command `nismkdir` failed to make the new directory *domainname* when running `nisserver` to create a nonroot master. The parent domain does not have create permission to create this new domain.

- If you are not the owner of the domain or a group member of the parent domain, rerun the script as the owner or as a group member of the parent domain.
- If `rpc.nisd` is not running on the new master server of the domain that you are trying to create, restart `rpc.nisd`.

`**ERROR: it failed to promote new master for the` *domainname* `directory`

The NIS+ command `nismkdir` failed to promote the new master for the directory *domainname* when creating a nonroot master with the `nisserver` script.

- If you do not have modify permission in the parent domain of this domain, rerun the script as the owner or as a group member of the parent domain.
- If `rpc.nisd` is not running on the servers of the domain that you are trying to promote, restart `rpc.nisd` on these servers and rerun `nisserver`.

`**ERROR: it failed to replicate the` *directory-name* `directory`

The NIS+ command `nismkdir` failed to create the new replica for the directory *directory-name*.

- If `rpc.nisd` is not running on the master server of the domain that you are trying to replicate, restart `rpc.nisd` on the master server, rerun `nisserver`.
- If `rpc.nisd` is not running on the new replica server, restart it on the new replica and rerun `nisserver`.

`**ERROR: invalid group name. It must be a group in the` *root-domain* `domain.`

This message indicates that you used an invalid group name while trying to configure a root master server. Rerun `nisserver -r` with a valid group name for *root-domain*.

`**ERROR: invalid name "`*client-name*`" It is neither an host nor an user name.`

This message indicates that you typed an invalid *client-name*.

- If *client-name* was spelled incorrectly, rerun `nisclient -c` with the correct *client-name*.
- If *client-name* was spelled correctly, but it does not exist in the proper table, put *client-name* into the proper table and rerun `nisclient -c`. For example, a user client belongs in the passwd table, and a host client belongs in the hosts table.

`**ERROR: `*hostname*` is a master server for this domain. You cannot demote a master server to replica. If you really want to demote this master, you should promote a replica server to master using nisserver with the M option.`

You cannot directly convert a master server to a replica server of the same domain. You can, however, change a replica to be the new master server of a domain by running `nisserver -M` with the replica host name as the new master. This automatically makes the *old* master a replica.

`**ERROR: missing hostnames or usernames.`

This message indicates that you did not type the client names on the command line. Rerun `nisclient -c` with the client names.

`**ERROR: NIS+ group name must end with a "."`

This message indicates that you did not specify a fully qualified group name ending with a period. Rerun the script with a fully qualified group name.

`**ERROR: NIS+ server is not running on `*remote-host*`. You must do the following before becoming an NIS+ server: 1. become an NIS+ client of the parent domain or any domain above the domain which you plan to serve. (nisclient) 2. start the NIS+ server. (rpc.nisd)`

This message indicates that `rpc.nisd` is not running on the remote machine that you are trying to convert to an NIS+ server. Use the `nisclient` script to become an NIS+ client of the parent domain or any domain above the domain you plan to serve; start `rpc.nisd` on *remote-host*.

`**ERROR: nisinit failed.`

`nisinit` was unable to create the `NIS_COLD_START` file.

Check the following:

- That the NIS+ server you specified with the `-H` option is running—use `ping`
- That you typed the correct domain name
- That `rpc.nisd` is running on the server
- That the nobody class has read permission for this domain

`**ERROR: NIS map transfer failed. `*tablename*` table will not be loaded.`

NIS+ was unable to transfer the NIS map for this table to the NIS+ database.

- If the NIS server host is running, try running the script again. The error might have been due to a temporary failure.
- If all tables have this problem, try running the script again using a different NIS server.

`**ERROR: no permission to create directory` *domainname*
The parent domain does not have create permission to create this new domain. If you are not the owner of the domain or as a group member of the parent domain, rerun the script as the owner, or as a group member of the parent domain.

`**ERROR: no permission to replicate directory` *domainname*`.`
This message indicates that you do not have permission to replicate the domain. Rerun the script as the owner or as a group member of the domain.

`error receiving zone transfer`
DNS error message. This usually indicates a syntax error in one of the primary server's DNS files. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

`**ERROR: table` *tablename* `.org_dir.`*domainname* `does not exist."` *tablename* `table will not be loaded."`
The script did not find the NIS+ table *tablename*.

- If *tablename* is spelled incorrectly, rerun the script with the correct table name.
- If the *tablename* table does not exist, use `nissetup` to create the table if *tablename* is one of the standard NIS+ tables. Or use `nistbladm` to create the private table *tablename*. Then rerun the script to populate this table.
- If the *tablename* table exists, the error might have been caused by the NIS+ server being temporarily busy. Rerun the script to populate this *tablename* table.

`**ERROR: this name "`*clientname*`" is in both the passwd and hosts tables. You cannot have an username same as the host name.`
client-name appears in both the passwd and hosts tables. One name is not allowed to be in both of these tables. Manually remove the entry from either the passwd or hosts table. Then, rerun `nisclient -c`.

`**ERROR: You cannot use the -u option as a root user.`
This message indicates that the superuser tried to run `nisclient -u`. The `-u` option is for initializing ordinary users only. Superusers do not need be initialized as NIS+ clients.

`**ERROR: You have specified the` *Z* `option after having selected the` *X* `option. Please select only one of these options [`*list*`]. Do you want to see more information on this command?`
The script you are running allows you to choose only one of the listed options.

- Type `y` to view additional information.
- Type `n` to stop the script and exit.

After exiting the script, rerun it with just one of the options.

`**ERROR: you must specify a fully qualified groupname.`

This message indicates that you did not specify a fully qualified group name ending with a period. Rerun the script with a fully qualified group name.

`**ERROR: you must specify both the NIS domainname (-y) and the NIS server host name (-h).`

This message indicates that you did not type either the NIS domain name and/or the NIS server host name. Type the NIS domain name and the NIS server host name at the prompt or on the command line.

`**ERROR: you must specify one of these options: -c, -i, -u, -r.`

This message indicates that one of these options, `-c`, `-i`, `-u`, `-r` was missing from the command line. Rerun the script with the correct option.

`**ERROR: you must specify one of these options: -r, -M or -R"`

This message indicates that you did not type any of the `-r` or the `-M` or the `-R` options. Rerun the script with the correct option.

`**ERROR: you must specify one of these options: -C, -F, or -Y`

This message indicates that you did not type either the `-Y` or the `-F` option. Rerun the script with the correct option.

`**ERROR: You must be root to use -i option.`

This message indicates that an ordinary user tried to run `nisclient -i`. Only the superuser has permission to run `nisclient -i`.

`Error while talking to callback proc`

An RPC error occurred on the server while it was calling back to the client. The transaction was aborted at that time and any unsent data was discarded. Check the `syslog` on the server for more information.

This message is generated by the NIS+ error code constant: `NIS_CBERROR`. See the `nis_tables` man page for additional information.

`First/Next chain broken`

This message indicates that the connection between the client and server broke while a callback routine was posting results. This could happen if the server died in the middle of the process.

This message is generated by the NIS+ error code constant: `NIS_CHAINBROKEN`.

`Generic system error`

Some form of generic system error occurred while attempting the request. Check the `syslog` record on your system for error messages from the server.

This message usually indicates that the server has crashed or the database has become corrupted. This message might also be generated if you incorrectly specify the name of a server or replica as if it belonged to the domain it was servicing rather than the domain above. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information.

This message is generated by the NIS+ error code constant: `NIS_SYSTEMERROR`. See the `nis_tables` and `nis_names` man pages for additional information.

`Illegal object type for operation`

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for a description of these type of problems.

This message is generated by the NIS+ error code constant: `DB_BADOBJECT`.

`insufficient permission to update credentials.`

This message is generated by the `nisaddcred` command when you have insufficient permission to execute an operation. This could be insufficient permission at the table, column, or entry level. Use `niscat -o cred.org_dir` to determine what permissions you have for that cred table. If you need additional permission, you or the system administrator can change the permission requirements of the object or add you to a group that does have the required permissions.

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information about permission problems.

`Invalid Object for operation`

- *Name context*. The name passed to the function is not a legal NIS+ name.
- *Table context.* The object pointed to is not a valid NIS+ entry object for the given table. This could occur if it had a mismatched number of columns, or a different data type (for example, binary or text) than the associated column in the table.

This message is generated by the NIS+ error code constant: `NIS_INVALIDOBJ`. See the `nis_tables` and `nis_names` man pages for additional information.

`invalid usecs` *Routine_name*`: invalid usecs`

This message is generated when the value in the `tv_usecs` field of a variable of type `struct time stamp` is larger than the number of microseconds in a second. This is usually due to some type of software error.

tablename `is not a table`

The object with the name *tablename* is not a table object. For example, the `nisgrep` and `nismatch` commands will return this error if the object you specify on the command line is not a table.

`Link Points to illegal name`

The passed name resolved to a LINK type object and the contents of the object pointed to an invalid name.

You cannot link table entries. A link at the entry level can produce this error message.

This message is generated by the NIS+ error code constant: `NIS_LINKNAMEERROR`. See the `nis_tables` and `nis_names` man pages for additional information.

`Load limit of` *number* `reached!`

An attempt has been made to create a child process when the maximum number of child processes have already been created on this server. This message is seen on the server's system log, but only if the threshold for logging messages has been set to include `LOG_WARNING` level messages.

`login and keylogin passwords differ.`

This message is displayed when you are changing your password with `nispasswd` and the system has changed your password, but has been unable to update your credential entry in the cred table with the new password and also unable to restore your original password in the passwd table. This message is followed by the instructions:

```
Use NEW password for login and OLD password for
keylogin. Use "chkey -p" to reencrypt the credentials with
the new login password. You must keylogin explicitly after
your next login.
```

These instructions are then followed by a status message explaining why it was not possible to revert back to the old password. If you see these messages, be sure to follow the instructions as given.

`Login incorrect`

The most common cause of a "login incorrect" message is mistyping the password. Try it again. Make sure you know the correct password. Remember that passwords are case-sensitive (uppercase letters are considered different than lowercase letters) and that the letter "o" is not interchangeable with the numeral "0," nor is the letter "l" the same as the numeral "1".

For other possible causes of this message, see "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

`log_resync: Cannot truncate transaction log file`

An attempt has been made to checkpoint the log, and the `rpc.nisd` daemon is trying to shrink the log file after deleting the checkpointed entries from the log. See the `ftruncate` man pages for a description of various factors that might cause this routine to fail. See also "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

`Malformed Name or illegal name`

The name passed to the function is not a legal or valid NIS+ name.

One possible cause for this message is that someone changed an existing domain name. Existing domain names should not be changed. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

This message is generated by the NIS+ error code constant: `NIS_BADNAME`. See the `nis_tables` man page for additional information.

`_map_addr: RPC timed out.`

A process or application could not contact NIS+ within its default time limit to get necessary data or resolve host names from NIS+. In most cases, this problem will solve itself after a short wait. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*for additional information about slow performance problems.

`Master server busy full dump rescheduled`

This message indicates that a replica server has been unable to update itself with a full dump from the master server because the master is busy. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information.

String `Missing or malformed attribute`

The name of an attribute did not match with a named column in the table, or the attribute did not have an associated value.

This could indicate an error in the syntax of a command. The *string* should give an indication of what is wrong. Common causes are spelling errors, failure to correctly place the equals sign (=), an incorrect column or table name, and so forth.

This message is generated by the NIS+ error code constant: `NIS_BADATTRIBUTE`. See the `nis_tables` man page for additional information.

`Modification failed`

Returned by the `nisgrpadm` command when someone else modified the group during the execution of your command. Check to see who else is working with this group. Reissue the command.

This message is generated by the NIS+ error code constant: `NIS_IBMODERROR`.

`Modify operation failed`

The attempted modification failed for some reason.

This message is generated by the NIS+ error code constant: `NIS_MODFAIL`. See the `nis_tables` and `nis_names` man pages for additional information.

`Name not served by this server`

A request was made to a server that does not serve the specified name. Normally this will not occur; however, if you are not using the built-in location mechanism for servers, you might see this if your mechanism is broken.

Other possible causes are:

- Cold-start file corruption. Delete the `/var/nis/NIS_COLD_START` file and then reboot.
- Cache problem such as the local cache being out of date. Kill the `nis_cachemgr` by stopping the NIS+ service, remove the `/var/nis/NIS_SHARED_DIRCACHE` file, and then reboot. (If the problem is not in the root directory, you might be able to kill the domain cache manager and try the command again.)
- Someone removed the directory from a replica.

This message is generated by the NIS+ error code constant: `NIS_NOT_ME`. See the `nis_tables` and `nis_names` man pages for additional information.

`Named object is not searchable`

The table name resolved to an NIS+ object that was not searchable.

This message is generated by the NIS+ error code constant: `NIS_NOTSEARCHABLE`. See the `nis_tables` man page for additional information.

`Name/entry isn't unique`

An operation has been requested based on a specific search criteria that returns more than one entry. For example, you use `nistbladm -r`to delete a user from the passwd table, and there are two entries in that table for that user name as shown as follows:

```
mymachine# nistbladm -r [name=arnold],passwd.org_dir
Can't remove entry: Name/entry isn't unique
```

You can apply your command to multiple entries by using the `-R` option rather than `-r`. For example, to remove all entries for `arnold`:

```
mymachine# nistbladm -R name=arnold],passwd.org_dir
```

`NIS+ error`

The NIS+ server has returned an error, but the `passwd` command determines exactly what the error is.

`NIS+ operation failed`

This generic error message should be rarely seen. Usually it indicates a minor software problem that the system can correct on it own. If it appears frequently, or appears to be indicating a problem that the system is not successfully dealing with, contact the Sun Solutions Center.

This message is generated by the NIS+ error code constant: `NIS_FAIL`.

string`: NIS+ server busy try again later.`

See “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for possible causes.

`NIS+ server busy try again later.`

Self explanatory. Try the command later.

See also “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for possible causes.

`NIS+ server for` *string* `not responding still trying`

See “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for possible causes.

`NIS+ server not responding`

See “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for possible causes.

`NIS+ server needs to be checkpointed. Use nisping -C`*domainname*

Caution – Checkpoint immediately! Do not wait!

This message is generated at the `LOG_CRIT` level on the server’s system log. It indicates that the log is becoming too large. Use `nisping -C` *domainname* to truncate the log by checkpointing.

See also “NIS Troubleshooting” in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information on log size.

`NIS+ servers unreachable`

This soft error indicates that a server for the desired directory of the named table object could not be reached. This can occur when there is a network failure or the server has crashed. A new attempt might succeed. See the description of the `-HARD_LOOKUP` flag in the `nis_tables` and `nis_names` man pages.

This message is generated by the NIS+ error code constant: `NIS_NaMEUNREACHABLE`.

`NIS+ service is unavailable or not installed`

Self-explanatory. This message is generated by the NIS+ error code constant: `NIS_UNAVAIL`.

`NIS+: write ColdStart File: xdr_directory_obj failed`

The most likely causes for this message are:

- Bad or incorrect parameters. Check the syntax and accuracy of whatever command you most recently entered.
- An attempt to allocate system memory failed. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for a discussion of memory problems.
- If your command syntax is correct, and your system does not seem to be short of memory, contact the Sun Solutions Center.

`nis_checkpoint_svc: readonly child instructed to checkpoint ignored.`
This is a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_dumplog_svc: readonly child called to dump log, ignore`
This is a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_dump_svc: load limit reached.`
The maximum number of child processes permitted on your system has been reached.

`nis_dump_svc: one replica is already resyncing.`
Only one replica can resync from a master at a time. Try the command later.

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on these three error messages.

`nis_dump_svc: Unable to fork a process.`
The fork system call has failed. See the `fork` man page for possible causes.

`nis_mkdir_svc: read-only child called to mkdir, ignored`
This is a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_ping_svc: read-only child was ping ignored.`
This is a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nis_rmdir_svc: readonly child called to rmdir, ignored`
This is a status message indicating that a read-only process attempted to perform an operation restricted to the parent process, and the attempt was aborted. No action need be taken.

`nisaddcred: no password entry for uid` *userid* `nisaddcred: unable to create credential.`

These two messages are generated during execution of the `nispopulate` script. The NIS+ command `nisaddcred` failed to add a LOCAL credential for the user ID *userid* on a remote domain. (This only happens when you are trying to populate the `passwd table` in a remote domain.)

To correct the problem, add a table path in the local passwd table:

```
# nistbladm -u -p passwd.org_dir.remote-domain passwd.org_dir
```

The *remote-domain* must be the same domain that you specified with the `-d` option when you ran `nispopulate`. Rerun the script to populate the passwd table.

`No file space on server`

Self-explanatory.

This message is generated by the NIS+ error code constant: `NIS_NOFILESPACE`.

`No match`

This is most likely an error message from the shell, caused by failure to escape the brackets when specifying an indexed name. For example, failing to set off a bracketed indexed name with quote marks would generate this message because the shell would fail to interpret the brackets as shown as follows:

```
# nistbladm -m shell=/bin/csh [name=miyoko],passwd.org_dir No match
```

The correct syntax is:

```
# nistbladm -m shell=/bin/csh '[name=miyoko],passwd.org_dir'
```

`No memory`

Your system does not have enough memory to perform the specified operation. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information on memory problems.

`Non NIS+ namespace encountered`

The name could not be completely resolved. This usually indicates that the name passed to the function resolves to a namespace that is outside the NIS+ name tree. In other words, the name is contained in an unknown directory. When this occurs, this error is returned with an NIS+ object of type `DIRECTORY`.

This message is generated by the NIS+ error code constant: `NIS_FOREIGNNS`. See the `nis_tables` or `nis_names` man pages for additional information.

`No password entry for uid` *userid* `No password entry found for uid` *userid*

Both of these two messages indicate that no entry for this user was found in the passwd table when trying to create or add a credential for that user. (Before you can create or add a credential, the user must be listed in the passwd table.)

- The most likely cause is misspelling the user's *userid* on the command line. Check your command line for correct syntax and spelling.
- Check that you are either in the correct domain, or specifying the correct domain on the command line.
- If the command line is correct, check the passwd table to make sure the user is listed under the *userid* you are entering. This can be done with `nismatch`:

```
mymachine# nismatch uid=userid passwd.org_dir.
```

If the user is not listed in the passwd table, use `nistbladm` or `nisaddent` to add the user to the passwd table before creating the credential.

`No shadow password information`
This means that password aging cannot be enforced because the information used to control aging is missing.

`Not found` *String* `Not found`
Names context. The named object does not exist in the namespace.

Table context. No entries in the table matched the search criteria. If the search criteria was null (return all entries), then this result means that the table is empty and can safely be removed.

If the `-FOLLOW_PATH` flag was set, this error indicates that none of the tables in the path contain entries that match the search criteria.

This message is generated by the NIS+ error code constant: `NIS_NOTFOUND`. See the `nis_tables` and `nis_names` man pages for additional information.

See also "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for general information on this type of problem.

`Not Found no such name`
This hard error indicates that the named directory of the table object does not exist. This could occur when the server that should be the parent of the server that serves the table, does not know about the directory in which the table resides.

This message is generated by the NIS+ error code constant: `NIS_NOSUCHNAME`. See the `nis_names` and `nis_names` man pages for additional information.

See also "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for general information on this type of problem.

`Not master server for this domain`
This message might mean that an attempt was made to directly update the database on a replica server.

This message might also mean that a change request was made to a server that serves the name, but it is not the master server. This can occur when a directory object changes and it specifies a new master server. Clients that have cached copies of that directory object in their `/var/nis/NIS_SHARED_DIRCACHE` file should stop the NIS+ service, remove the `/var/nis/NIS_SHARED_DIRCACHE` file, and then restart the NIS+ service.

This message is generated by the NIS+ error code constant: `NIS_NOTMASTER`. See the `nis_tables` and `nis_names` man pages for additional information.

`Not owner`

The operation you attempted can only be performed by the object's owner, and you are not the owner.

This message is generated by the NIS+ error code constant: `NIS_NOTOWNER`.

`Object with same name exists`

An attempt was made to add a name that already exists. To add the name, first remove the existing name and then add the new name or modify the existing named object.

This message is generated by the NIS+ error code constant: `NIS_NAMEEXISTS`. See the `nis_tables` and `nis_names` man pages for additional information.

`parse error:` *string* `(key` *variable*`)`

This message is displayed by the `nisaddent` command when it attempts to use database files from a `/etc` directory and there is an error in one of the file's entries. The first variable should describe the problem, and the variable after `key` should identify the particular entry at fault. If the problem is with the `/etc/passwd` file, you can use `/usr/sbin/pwck` to check it.

`Partial Success`

This result is similar to `NIS_NOTFOUND`, except that it means the request succeeded but resolved to zero entries.

When this occurs, the server returns a copy of the table object instead of an entry so that the client can then process the path or implement some other local policy.

This message is generated by the NIS+ error code constant: `NIS_PARTIAL`. See the `nis_tables` man page for additional information.

`Passed object is not the same object on server`

An attempt to remove an object from the namespace was aborted because the object that would have been removed was not the same object that was passed in the request.

This message is generated by the NIS+ error code constant: `NIS_NOTSAMEOBJ`. See the `nis_tables` and `nis_names` man pages for additional information.

`Password does not decrypt secret key for` *name*

Possible causes:

- You might have incorrectly typed the password.
- There might not be an entry for *name* in the cred table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt).
- The Secure RPC password does not match the login password.
- The `nsswitch.conf` file might be directing the query to a local password in an `/etc/passwd` file that is different from the NIS+ password recorded in the cred table. (Note that the actual encrypted passwords are stored locally in the `/etc/shadow` file.)

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on diagnosing and solving these types of problems.

`Password has not aged enough`

This message indicates that your password has not been in use long enough and that you cannot change it until it has been in use for *N* (a number of) days.

`Permission denied`

Returned when you do not have the permissions required to perform the operation you attempted. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information.

This message might be related to a login or password matter, or an NIS+ security problem. The most common cause of a Permission denied message is that the password of the user receiving it has been locked by an administrator or the user's account has been terminated.

`Permissions on the password database may be too restrictive`

You do not have authorization to read (or otherwise use) the contents of the passwd field in an NIS+ table. See Chapter 15, for information on NIS+ access rights.

`Please notify your System Administrator`

When displayed as a result of an attempt to update password information with the `passwd` command, this message indicates that the attempt failed for one of many reasons. For example, the service might not be available, a necessary server is down, there is a "permission denied" type problem, and so forth. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for a discussion of various types of security problems.

`Please check your /etc/nsswitch.conf file`

The `nsswitch.conf` file specifies a configuration that is not supported for `passwd` update. See "`nsswitch.conf` File Requirements" on page 290 for supported configurations.

`Probable success`

Name context. The request was successful; however, the object returned came from an object cache and not directly from the server. (If you do not want to see objects from object caches, you must specify the flag `-NO_CACHE` when you call the lookup function.)

Table context. Even though the request was successful, a table in the search path was not able to be searched, so the result might not be the same as the one you would have received if that table had been accessible.

This message is generated by the NIS+ error code constant: `NIS_S_SUCCESS`. See the `nis_tables` and `nis_names` man pages for additional information.

`Probably not found`

The named entry does not exist in the table; however, not all tables in the path could be searched, so the entry might exist in one of those tables.

This message is generated by the NIS+ error code constant: `NIS_S_NOTFOUND`. See the `nis_tables` man page for additional information.

`Query illegal for named table`

A problem was detected in the request structure passed to the client library.

This message is generated by the NIS+ error code constant: `NIS_BADREQUEST`. See the `nis_tables` man page for additional information.

`Reason: can't communicate with ypbind.`

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*

`replica_update: Child process attempting update, aborted`

This is a status message indicating that a read-only process attempted an update and the attempt was aborted.

`replica_update: error result was string`

This message indicates a problem (identified by *string*) in carrying out a dump to a replica. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for further information.

`replica_update: error result was Master server busy, full dump rescheduled replica_update: master server busy rescheduling the resync. replica_update: master server is busy will try later. replica_update: nis dump result Master server busy, full dump rescheduled`

These messages all indicate that the server is busy and the dump will be done later.

`replica_update: nis dump result nis_perror` *errorstring*

This message indicates a problem (identified by the *error string*) in carrying out a dump to a replica. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for further information.

`replica_update:` *nnnn* `updates` *nnnn* `errors`

A status message indicating a successful update.

`replica_update: WARNING: last_update (`*directoryname*`) returned 0!`

An NIS+ process could not find the last update time stamp in the transaction log for that directory. This will cause the system to perform a full resync of the problem directory.

`Results Sent to callback proc`

This is a status message. No action need be taken.

This message is generated by the NIS+ error code constant: `NIS_CBRESULTS`. See the `nis_tables` man page for additional information.

`root_replica_update: update failed` *string*`: could not fetch object from master.`

This message indicates a problem in carrying out a dump to a replica. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for further information.

`RPC failure: "RPC failure on yp operation.`

This message is returned by `ypcat` when an NIS client's `nsswitch.conf` file is set to `files` rather than `nis`, and the server is not included in the `/etc/hosts` or `/etc/inet/ipnodes` file.

`Security exception on local system. UNABLE TO MAKE REQUEST.`

This message might be displayed if a user has the same login ID as a machine name. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information.

date: hostname: `sendmail (`*nnnn*`) : gethostbyaddr failed`

One common cause of this problem is entering IP addresses in NIS+, NIS, files, or DNS data sets with leading zeros. For example, you should never enter an IP address as `151.029.066.001`. The correct way to enter that address is: `151.29.66.1`.

`Server busy, try again`

The server was too busy to handle your request.

- For the add, remove, and modify operations, this message is returned when either the master server for a directory is unavailable or it is in the process of checkpointing its database.
- This message can also be returned when the server is updating its internal state.

- In the case of `nis_list`, if the client specifies a callback and the server does not have enough resources to handle the callback.

Retry the command at a later time when the server is available.

This message is generated by the NIS+ error code constant: `NIS_TRYAGAIN`. See the `nis_tables` and `nis_names` man pages for additional information.

`Server out of memory`
In most cases this message indicates a fatal result. It means that the server ran out of heap space.

This message is generated by the NIS+ error code constant: `NIS_NOMEMORY`. See the `nis_tables` and `nis_names` man pages for additional information.

`Sorry`
This message is displayed when a user is denied permission to login or change a password, and for security reasons the system does not display the reason for that denial because such information could be used by an unauthorized person to gain illegitimate access to the system.

`Sorry: less than` *nn* `days since the last change`
This message indicates that your password has not been in use long enough and that you cannot change it until it has been in use for *N* days. See "Changing Your Password" on page 287 for further information.

`_svcauth_des: bad nickname`
The nickname received from the client is invalid or corrupted, possibly due to network congestion. The severity of this message depends on what level of security you are running. At a low security level, this message is informational only; at a higher level, you might have to try the command again later.

`_svcauth_des: corrupted window from` *principalname*
The window that was sent does not match the one sent in the verifier.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level you might have to try the command again at some later time or take corrective action as described below.

Possible causes:

- The server's key pair has been changed. The client used the server's old public key while the server has a new secret key cached with `keyserv`. Run `keylogin` on both client and server.
- The client's key pair has been changed and the client has not run `keylogin` on the client system, so system is still sending the client's old secret key to the server, which is now using the client's new public key. Naturally, the two do not match. Run `keylogin` again on both client and server.

- Network corruption of data. Try the command again. If that does not work, use the `snoop` command to investigate and correct any network problems. Then run `keylogin` again on both server and client.

`_svcauth_des: decryption failure`
DES decryption for some authentication data failed. Possible causes:

- Corruption to a library function or argument.
- A problem with a DES encryption chip, if you are using one.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you might have to call the Sun Solutions Center for assistance. If the problem appears to be related to a DES encryption chip, call the Sun Solutions Center.

`_svcauth_des: decryption failure for` *principalname*
DES decryption for some authentication data failed. Possible causes:

- Corruption to a library function or argument.
- A problem with a DES encryption chip, if you are using one.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you might have to call the Sun Solutions Center for assistance. If the problem appears to be related to a DES encryption chip, call the Sun Solutions Center.

`_svcauth_des: invalid timestamp received from` *principalname*
The time stamp received from the client is corrupted, or the server is trying to decrypt it using the wrong key. Possible causes:

- Congested network. Retry the command.
- Server cached out the entry for this client. Check the network load.

`_svcauth_des: key_decryptsessionkey failed for` *principalname*
The `keyserv` process failed to decrypt the session key with the given public key. Possible causes are:

- The `keyserv` process is dead or not responding. Use `ps -e` to check if the `keyserv` process is running on the `keyserv` host. If it is not, then restart the NIS+ service and run `keylogin`.
- The server principal has not keylogged in. Run `keylogin` for the server principal.
- The server principal (host) does not have credentials. Run `nismatch` *hostname*`.`*domainname*`. cred.org_dir` on the client's home domain cred table. Create new credentials if necessary.
- `keyserv` might have been restarted, in which case certain long-running applications, such as `rpc.nisd`, `sendmail`, and `automountd`, also need to be restarted.
- DES encryption failure. Call the Sun Solutions Center.

`_svcauth_des: no public key for` *principalname*

The server cannot get the client's public key. Possible causes are:

- The principal has no public key. Run `nismatch` on the cred table of the principal's home domain. If there is no DES credential in that table for the principal, use `nisaddcred` to create one, and then run `keylogin` for that principal.
- The naming service specified by a `nsswitch.conf` file is not responding.

`_svcauth_des: replayed credential from` *principalname*

The server has received a request and finds an entry in its cache for the same client name and conversation key with the time stamp of the incoming request *before* that of the one currently stored in the cache.

The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information. At a higher level, you might have to take corrective action as described below.

Possible causes are:

- The client and server clocks are out of sync. Use `rdate` to resync the client clock to the server clock.
- The server is receiving requests in random order. This could occur if you are using multithreading applications. If your applications support TCP, then set `/etc/netconfig` (or your `NETPATH` environment variable) to `tcp`.

`_svcauth_des: timestamp is earlier than the one previously seen from` *principalname*

The time stamp received from the client on a subsequent call is earlier than one seen previously from that client. The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you might have some corrective action as described below.

Possible causes are:

- The client and server clocks are out of sync. Use `rdate` to resynch the client clock to the server clock.
- The server cached out the entry for this client. The server maintains a cache of information regarding the current clients. This cache size equals 64 client handles.

`_svcauth_des: timestamp expired for` *principalname*

The time stamp received from the client is not within the default 35-second window in which it must be received. The severity of this message depends on what level of security you are running. At a low security level, this message is primarily for your information; at a higher level, you might have to take corrective action as described below.

Possible causes are:

- The 35-second window is too small to account for slow servers or a slow network.
- The client and server clocks are so far out of sync that the window cannot allow for the difference. Use `rdate` to resynchronize the client clock to the server clock.
- The server has cached out the client entry. Retry the operation.

`Too Many Attributes`

The search criteria passed to the server had more attributes than the table had searchable columns.

This message is generated by the NIS+ error code constant: `NIS_TOOMANYATTRS`. See the `nis_tables` man page for additional information.

`Too many failures - try later`

`Too many tries; try again later`

These messages indicate that you have had too many failed attempts (or taken too long) to either log in or change your password. See "The `Login incorrect` Message" on page 286 or "Password Change Failures" on page 288 for further information.

`Unable to authenticate NIS+ client`

This message is generated when a server attempts to execute the callback procedure of a client and gets a status of `RPC_AUTHERR` from the RPC `clnt_call()`. This is usually caused by out-of-date authentication information. Out-of-date authentication information can occur when the system is using data from a cache that has not been updated, or when there has been a recent change in the authentication information that has not yet been propagated to this server. In most cases, this problem should correct itself in a short period of time.

If this problem does not self-correct, it might indicate one of the following problems:

- Corrupted `/var/nis/NIS_SHARED_DIRCACHE` file. Stop the NIS+ service, remove the `/var/nis/NIS_SHARED_DIRCACHE` file, and restart the NIS+ service.
- Corrupted `/var/nis/NIS_COLD_START` file. Remove the file and then run `nisinit` to recreate it.
- Corrupted `/etc/.rootkey` file. Run `keylogin -r`.

This message is generated by the NIS+ error code constant: `NIS_CLNTAUTH`.

`Unable to authenticate NIS+ server`

In most cases, this is a minor software error from which your system should quickly recover without difficulty. It is generated when the server gets a status of `RPC_AUTHERR` from the RPC `clnt_call`.

If this problem does not quickly clear itself, it might indicate a corrupted `/var/nis/NIS_COLD_START`, `/var/nis/NIS_SHARED_DIRCACHE`, or `/etc/.rootkey` file.

This message is generated by the NIS+ error code constant: `NIS_SRVAUTH`.

`Unable to bind to master server for name '`*string*`'`
See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on this type of problem. This particular message might be caused by adding a trailing dot to the server's domain name in the `/etc/defaultdomain` file.

`Unable to create callback.`
The server was unable to contact the callback service on your machine. This results in no data being returned.

See the `nis_tables` man page for additional information.

`Unable to create process on server`
This error is generated if the NIS+ service routine receives a request for a procedure number which it does not support.

This message is generated by the NIS+ error code constant: `NIS_NOPROC`.

string`: Unable to decrypt secret key for `*string*`.`
Possible causes:

- You might have incorrectly typed the password.
- There might not be an entry for *name* in the cred table.
- NIS+ could not decrypt the key because the entry might be corrupt.
- The `nsswitch.conf` file might be directing the query to a local password in an `/etc/passwd` file that is different than the NIS+ password recorded in the cred table.

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on diagnosing and solving these type of problem.

`Unknown error`
This is displayed when the NIS+ error handling routine receives an error of an unknown type.

`Unknown object`
The object returned is of an unknown type.

This message is generated by the NIS+ error code constant: `NIS_UNKNOWNOBJ`. See the `nis_names` man page for additional information.

`update_directory:` *nnnn* `objects still running.`
This is a status message displayed on the server during the update of a directory during a replica update. You do not need to take any action.

`User` *principalname* `needs Secure RPC credentials to login but has none.`
The user has failed to perform a keylogin. This problem usually arises when the user has different passwords in `/etc/shadow` and a remote NIS+ passwd table.

`Warning: couldn't reencrypt secret key for` *principalname*
The most likely cause of this problem is that your Secure RPC password is different from your login password (or you have one password on file in a local `/etc/shadow` file and a different one in a remote NIS+ table) and you have not yet done an explicit `keylogin`. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information on these types of problems.

`WARNING: db::checkpoint: could not dump database: No such file or directory`
This message indicates that the system was unable to open a database file during a checkpoint. Possible causes:

- The database file was deleted.
- The server is out of file descriptors.
- There is a disk problem
- You or the host do not have correct permissions.

`WARNING: db_dictionary::add_table: could not initialize database from scheme`
The database table could not be initialized. Possible causes:

- There was a system resource problem See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*).
- You incorrectly specified the new table in the command syntax.
- The database is corrupted.

`WARNING: db_query::db_query:bad index`
In most cases this message indicates incorrect specification of an indexed name. Make sure that the indexed name is found in the specified table. Check the command for spelling and syntax errors.

`**WARNING: domain` *domainname* `already exists.`
This message indicates that the domain you tried to create already exists.

- If you are trying to promote a new nonroot master server or are recovering from a previous `nisserver` problem, continue running the script.
- If *domainname* was spelled incorrectly, rerun the script with the correct domain name.

`**WARNING: failed to add new member` *NIS+_principle* `into the` *groupname* `group. You will need to add this member manually: 1. /usr/sbin/nisgrpadm -a groupname NIS+_principal`

The NIS+ command `nisgrpadm` failed to add a new member into the NIS+ group *groupname*. Manually add this NIS+ principal by typing:

```
# /usr/sbin/nisgrpadm -a groupname NIS+_principal
```

`**WARNING: failed to populate` *tablename* `table.`

The `nisaddent` command was unable to load the NIS+ *tablename* table. A more detailed error message usually appears before this warning message.

`**WARNING: hostname specified will not be used. It will use the local hostname instead.`

This message indicates that you typed a remote host name with the `-H` option. The `nisserver -r`script does not configure remote machines as root master servers.

- If the local machine is the one that you want to convert to an NIS+ root master server, no other action is needed. The `nisserver -r`script will ignore the host name you typed.
- If you actually want to convert the remote host (instead of the local machine) to an NIS+ root master server, exit the script. Rerun the `nisserver -r`script on the remote host.

`**WARNING:` *hostname* `is already a server for this domain. If you choose to continue with the script, it will try to replicate the groups_dir and org_dir directories for this domain.`

This is a message warning you that *hostname* is already a replica server for the domain that you are trying to replicate.

- If you are running the script to fix an earlier `nisserver` problem, continue running the script.
- If *hostname* was mistakenly entered, rerun the script with the correct host name.

`**WARNING:` *alias-hostname* `is an alias name for host` *canonical_hostname*`. You cannot create credential for host alias.`

This message indicates that you have typed a host alias in the name list for `nisclient -c`. The script asks you if you want to create the credential for the canonical host name, since you should not create credentials for host alias names.

`**WARNING: file` *directory-path/tablename* `does not exist!` *tablename* `table will not be loaded.`

The script was unable to find the input file for *tablename*.

- If *directory-path*/*tablename* is spelled incorrectly, rerun the script with the correct table name.
- If the*directory-path*/*tablename* file does not exist, create and update this file with the proper data. Then rerun the script to populate this table.

`**WARNING: NIS auto.master map conversion failed. auto.master table will not be loaded.`

The `auto.master` map conversion failed while trying to convert all the dots to underscores in the auto_master table. Rerun the script with a different NIS server.

`**WARNING: NIS netgroup map conversion failed. netgroup table will not be loaded.`

The `netgroup` map conversion failed while trying to convert the NIS domain name to the NIS+ domain name in the `netgroup` map. Rerun the script with a different NIS server.

`**WARNING: nisupdkeys failed on directory` *domainname*`. This script will not be able to continue. Please remove the` *domainname* `directory using 'nisrmdir'.`

The NIS+ command `nisupdkeys` failed to update the keys in the listed directory object. If `rpc.nisd` is not running on the new master server that is supposed to serve this new domain, restart `rpc.nisd`. Then use `nisrmdir` to remove the *domainname* directory. Finally, rerun `nisserver`.

`WARNING: nisupdkeys failed on directory` *directory-name* `You will need to run nisupdkeys manually: 1. /usr/lib/nis/nisupdkeys` *directory-name*

The NIS+ command `nisupdkeys` failed to update the keys in the listed directory object. Manually update the keys in the directory object by typing:

```
# /usr/lib/nis/nisupdkeys directory-name
```

`**WARNING: once this script is executed, you will not be able to restore the existing NIS+ server environment. However, you can restore your NIS+ client environment using "nisclient -r" with the proper domainname and server information. Use "nisclient -r" to restore your NIS+ client environment.`

These messages appear if you have already run the script at least once before to set up an NIS+ server and indicate that NIS+-related files will be removed and recreated as needed if you decide to continue running this script.

- If it is all right for these NIS+ files to be removed, continue running the script.
- If you want to save these NIS+ files, exit the script by typing "**`n`**" at the `Do you want to continue?` prompt. Then save the NIS+ files in a different directory and rerun the script.

`**WARNING: this script removes directories and files related to NIS+ under /var/nis directory with the exception of the NIS_COLD_START and NIS_SHARED_DIRCACHE files which will be renamed to <file>.no_nisplus. If you want to save these files, you should abort from this script now to save these files first.`

See "WARNING: once this script is executed,..." above.

`**WARNING: you must specify the NIS domainname.`

This message indicates that you did not type the NIS domain name at the prompt. Type the NIS server domain name at the prompt.

`**WARNING: you must specify the NIS server` *hostname*`. Please try again.`

This message indicates that you did not type the NIS server host name at the prompt. Type the NIS server host name at the prompt.

`Window verifier mismatch`

This is a debugging message generated by the `_svcauth_des` code. A verifier could be invalid because a key was flushed out of the cache. When this occurs, `_svcauth_des` returns the `AUTH_BADCRED` status.

`You (`*string*`) do not have Secure RPC credentials in NIS+ domain '`*string*`'`

This message could be caused by trying to run `nispasswd` on a server that does not have the credentials required by the command. (Keep in mind that servers running at security level 0 do not create or maintain credentials.)

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for additional information on credential, ownership, and permission problems.

`You may not change this password`

This message indicates that your administrator has forbidden you to change your password.

`You may not use nisplus repository`

You used `-r nisplus` in the command line of your command, but the appropriate entry in the NIS+ passwd table was not found. Check the passwd table in question to make sure it has the entry you want. Try adding `nisplus` to the `nsswitch.conf` file.

`Your password has been expired for too long`

`Your password is expired`

These messages refer to password aging and indicate that your password has been in use too long and needs to be changed now. See "The `will expire` Message" on page 286 for further information.

`Your password will expire in` *nn* `days`

`Your password will expire within 24 hours`

These messages refer to password aging and indicate that your password is about to become invalid and should be changed now. See "The `will expire` Message" on page 286 for further information.

`Your specified repository is not defined in the nsswitch file!`

This warning indicates that you have specified a password information repository with the `-r` option, but that password repository is not included in the passwd entry of the `nsswitch.conf` file. The command you have just used will perform its job and make whatever change you intend to the password information repository you specified with the `-r` flag. However, the change will be made to information that the `nsswitch.conf` file does not point to, so no one will ever gain the benefit of it until the switch file is altered to point to that repository.

For example, suppose the passwd entry of the switch file reads: `files nis`, and you used

```
passwd -r nisplus
```

to establish a password age limit. That limit would be ignored, as the switch file remains set to `files nis`.

`verify_table_exists: cannot create table for` *string* `nis_perror` *message*`.`

To perform an operation on a table, NIS+ first verifies that the table exists. If the table does not exist, NIS+ attempts to create it. If it cannot create the table, it returns this error message. The *string* portion of the message identifies the table that could not be located or created; the nis_perror *message* portion provides information as to the cause of the problem (you can look up that portion of the message as if it were an independent message in this appendix). Possible causes for this type of problem:

- The server was just added as a replica of the directory and it might not have the directory object. Run `nisping -C` to checkpoint.
- You are out of disk space. See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.
- Database corruption.
- Some other type of software error. Contact the Sun Solutions Center.

`ypcat: can't bind to NIS server for domain` *domainname*`. Reason: can't communicate with ypbind.`

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*

`ypoll: can't get any map parameter.`

See "NIS Troubleshooting" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*

APPENDIX **B**

System Administration Guide: Naming and Directory Services (NIS+) Updates

Solaris 10

There are no new features specifically related to NIS+ in this release.

FNS

FNS is no longer supported on Solaris. Information pertaining to FNS has been removed from the Solaris System Administration documentation set.

Service Management Facility (SMF)

NIS+ is now managed by the Service Management Facility (SMF). When you start, stop, or restart a NIS+ service, use the `svcadm` command. The NIS+ processes managed by SMF include `rpc.nisd` and `keyserv`. `nis_cachemgr` starts and stops in association with the `rpc.nisd` service. These services should no longer be started and stopped individually.

Use of SMF also impacts the implementation of optional behaviors with NIS+. See "NIS+ and the Service Management Facility" on page 85 for more information about using SMF with NIS+.

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.
application-level name service	Application-level name services are incorporated in applications offering services such as files, mail, and printing. Application-level name services are bound below enterprise-level name services. The enterprise-level name services provide contexts in which contexts of application-level name services can be bound.
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.
cache manager	The program that manages the local caches of NIS+ clients (`NIS_SHARED_DIRCACHE`), 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.
child domain	See *domain.*
checkpointing	The process of writing changes to NIS+ data that are stored in server memory and recorded in the transaction log to the NIS+ tables stored on disk. In other words, updating the NIS+ tables with recent changes to the NIS+ data set.
client	(1) In NIS+, the client is a principal (machine or user) requesting an NIS+ service from an NIS+ server. (2) 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.

	(3) In the client-server model, the client is an *application* that accesses 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*.
cold-start 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: 172.27.67.20.
DES	See *data encryption standard (DES)*.
directory	(1) An NIS+ directory is a container for NIS+ objects such as NIS+ tables, groups, or subdirectories (2) In UNIX, a container for files and subdirectories.
directory cache	A local file used to store data associated with directory objects.
distinguished name	A distinguished name is an entry in an X.500 directory information base (DIB) composed of selected attributes from each entry in the tree along a path leading from the root down to the named entry.
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 files wherein the DNS software stores the names and IP addresses of all the workstations in a domain.

domain	(1) In NIS+ a group of hierarchical objects managed by NIS+. There is one highest level domain (root domain) and zero or more subdomains. Domains and subdomains may be organized around geography, organizational or functional principles. ■ *Parent domain*. Relative term for the domain immediately above the current domain in the hierarchy. ■ *Child domain*. Relative term for the domain immediately below the current domain in the hierarchy. ■ *Root domain*. Highest domain within the current NIS+ hierarchy. (2) In the Internet, a part of a naming hierarchy usually corresponding to a Local Area Network (LAN) or Wide Area Network (WAN) or a portion of such a network. Syntactically, an Internet domain name consists of a sequence of names (labels) separated by periods (dots). For example, `sales.doc.com`. (3) 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 DNS administrative files. The domain name is required for the network information service database to work properly. See also *domain*.
Domain Name Service (DNS)	A service that provides the naming policy and mechanisms for mapping domain and machine names to addresses outside of the enterprise, such as those on the Internet. DNS is the network information service used by the Internet.
encryption key	See *data encrypting key*.
enterprise-level name service	An enterprise-level naming service identifies (names) machines (hosts), users and files within an enterprise-level network.
enterprise-level network	An "enterprise-level" network can be a single Local Area Network (LAN) communicating over cables, infra-red beams, or radio broadcast; or a cluster of two or more LANs linked together by cable or direct phone connections. Within an enterprise-level network, every machine is able to communicate with every other machine without reference to a global naming service such as DNS or X.500/LDAP.
entry	A single row of data in a database table.
GID	See *group ID*.

global name service	A global naming service identifies (names) those enterprise-level networks around the world that are linked together via phone, satellite, or other communication systems. This world-wide collection of linked networks is known as the “Internet.” In addition to naming networks, a global naming service also identifies individual machines and users within a given network.
group	(1) A collection of users who are referred to by a common name. (2) In NIS+ a collection of users who are collectively given specified access rights to NIS+ objects. NIS+ group information is stored in the NIS+ group table. (3) In UNIX, groups determine 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	The world-wide collection of networks interconnected by a set of routers that enable them to function and communicate with each other as a single virtual network.
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 system process that stores private keys.
local-area network (LAN)	Multiple systems at a single geographical site connected together for the purpose of sharing and exchanging data and software.
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 for a particular domain. Namespace changes are always made to the name service database kept by the domain’s master server. Each domain has only *one* master server.
MIS	Management information systems (or services)

name resolution	The process of translating workstation or user names to addresses.
name server	Servers that run one or more network name services.
name service switch	A configuration file (`/etc/nsswitch.conf`) that defines the sources from which an NIS+ client can obtain its network information.
name service	A network service that handles machine, user, printer, domain, router, an other network names and addresses.
namespace	(1) A namespace stores information that users, workstations, and applications must have to communicate across the network. (2) The set of all names in a naming system. *(3) NIS+ namespace*, A collection of hierarchical network information used by the NIS+ software. *(4) NIS namespace*. A collection of *non*-hierarchical network information used by the NIS software. *(5) DNS namespace*. A collection of networked workstations that use the DNS software.
network mask	A number used by software to separate the local subnet address from the rest of a given Internet protocol address.
network password	See Secure RPC password.
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 *replica* or *slave servers*.
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 *NIS*.
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.
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+ environment	For administrative purposes, an NIS+ environment refers to any situation in which the applicable `nsswitch.conf` file points to `nisplus`. Or any time a command is run with an option that forces it to operate on objects in an NIS+ namespace (for example, `passwd -r nisplus`).
NIS+ object	An NIS+ domain, directory, table, or group. See *domain*, *directory*, *group*, and *table*.

NIS+ principal	See *principal*.
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 cleared only after all of a master server's replicas have been updated.
NNSP	See *next naming system pointer*.
parent domain	See *domain*.
pinging	The process by which an NIS+ master server transfers a change a NIS+ data to the domain's replica servers.
preference rank number	A number which a machine uses to rank the order in which it tries to obtain namespace information from NIS+ servers. A machine will first try all servers with a given rank number before trying any server with the next highest rank number. For example, a machine will query NIS+ servers with a rank number of 0 before it queries any server with a rank number of 1.
preferred server	From the point of view of a client machine, a preferred NIS+ server is a server that the client should try to use for namespace information ahead of non-preferred servers. Servers that are listed in a client or domain's preferred server list are considered preferred servers for that client or domain.
preferred server list	A `client_info` table or a `client_info` file. Preferred server lists specify the preferred servers for a client or domain.
principal	Any user of NIS+ information whose credentials have been stored in the namespace. Any user or machine that can generate a request to a NIS+ server. There are two kinds of NIS+ principal: client users and client machines: ■ *Root principal*. A machine root user (user ID = 0). Requires only a DES credential. ■ *User principal*. Any nonroot user (user ID > 0). Requires local and DES credentials.
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 private key of the sender is only available to the owner of the key. Every user or machine has its 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.
record	See *entry*.
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 server	NIS+ server that maintains a duplicate copy of the domain's master NIS+ server database. Replicas run NIS+ server software and maintain copies of NIS+ tables. A replica server increases the availability of NIS+ services. Each NIS+ domain should have at least one, and perhaps more, replicas. (In an NIS namespace, a replica server was known as a *slave* server.)
reverse resolution	The process of converting workstation IP addresses to workstation names using the DNS software.
root domain	See *domain*.
root master server	The master server for a NIS+ root domain.
root replica server	NIS+ server that maintains a duplicate copy of the root domain's master NIS+ server database.
RPC	See remote procedure call (RPC).
Secure RPC password	Password required by Secure RPC protocol. This password is used to encrypt the private key. This password should always be identical to the user's login password.
server	(1) In NIS+, NIS, and DNS, a host machine providing naming services to a network. (2) In the *client-server model* for file systems, the server is a machine with computing 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. (3) A *daemon* that actually handles the providing of files.
server list	*See* preferred server list.

slave server	(1) A server system that maintains a copy of the NIS database. It has a disk and a complete copy of the operating system. (2) Slave servers are called *replica servers* in NIS+.
subnet	A working scheme that divides a single logical network into smaller physical networks to simplify routing.
table	In NIS+ a two-dimensional (nonrelational) database object containing NIS+ data in rows and columns. (In NIS an NIS map is analogous to a NIS+ table with two columns.) A table is the format in which NIS+ data is stored. NIS+ provides 16 predefined or system tables. Each table stores a different type of information.
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. Solaris 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.
wide-area network (WAN)	A network that connects multiple local-area networks (LANs) or systems at different geographical sites via phone, fiber-optic, or satellite links.
X.500	A global-level directory service defined by an Open Systems Interconnection (OSI) standard.

Index