Name Services Administration Guide

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

SunSoft
A Sun Microsystems, Inc. Business

Please
Recycle

Adobe PostScript

Contents

Figures

Tables

Preface

Name Services Administration Guide describes how to administer an existing NIS+ (Network Information Service Plus) and DNS (Domain Name Service) namespace. (You can also use it to initially set up a namespace using the NIS+ command set, although the recommended installation and setup method is to use the scripts as described in *Name Services Configuration Guide*.)

This manual is part of the Solaris™ 2.x System and Network Administration manual set.

Related Books

- *NIS+ Transition Guide*, describes how to make the transition from NIS to NIS+.
- *Name Services Configuration Guide*, gives an overview description of the NIS+ namespace, and describes how to install, set up, and configure NIS+ and DNS using the NIS+ installation and setup scripts and commands.

Who Should Use This Book

This book is written primarily for system and network administrators, but MIS managers can use it to evaluate NIS+.

How This Book Is Organized

This manual is organized into two parts:

- Setting Up NIS+ Components
- Administering NIS+

Part 1— Setting Up NIS+ Components

Part 1 provides step-by-step instructions for setting up the various components of an NIS+ namespace, from the root domain itself to individual clients. (If you are installing an entire NIS+ namespace from scratch, it is recommend that you use the installation and set up scripts described in *Name Services Configuration Guide*.)

Once you have an operational NIS+ namespace, you can add or change your namespace components as described in these chapters.

Chapter 1, "Setting Up the Root Domain", provides step-by-step instructions for setting up the root domain, including using the NIS-compatibility mode.

Chapter 2, "Setting Up NIS+ Clients", provides step-by-step instructions for setting up an NIS+ client and includes three different initialization methods. These instructions apply to clients in both the root domain and subdomains, whether all-NIS+ or NIS-compatible.

Chapter 3, "Setting Up NIS+ Servers", provides step-by-step instructions for setting up any kind of NIS+ server except the root master.

Chapter 4, "Setting Up a Non-Root Domain", provides step-by-step instructions for creating and setting up a subdomain, including designating its master and replica servers.

Chapter 5, "Setting Up NIS+ Tables", provides step-by-step instructions for populating NIS+ tables with information from input files or NIS maps.

Chapter 6, "Setting Up the Name Service Switch", provides step-by-step instructions for setting up the name service switch to be used with NIS, NIS+, or DNS, as well as to provide backward compatibility with the +/- syntax.

Part 2 — Administering NIS+

Part 2 covers the administration of a functioning NIS+ namespace.

Chapter 7, "Administering NIS+ Security", describes the NIS+ security system, how it affects the entire NIS+ namespace, and how to administer NIS+ security.

Chapter 8, "Administering NIS+ Credentials", describes how to use the commands that administer NIS+ credentials, `nisaddcred` and `nispasswd`; and other commands related to credential administration, `nisupdkeys` and `keylogin`.

Chapter 9, "Administering NIS+ Access Rights", describes how to use the commands that administer access rights to NIS+ objects and entries, such as `nisaddcred`, `nischmod`, `nischown`, `nischgrp`, and the `-c` and `-a` options of `nistbladm`.

Chapter 10, "Administering NIS+ Groups", describes how to use the `nisgrpadm` command to perform a variety of group administration tasks, from creating an NIS+ group to testing for membership in one.

Chapter 11, "Administering NIS+ Directories", describes how to use the commands that administer NIS+ directories, `nismkdir` and `nisrmdir`; the related commands `nisls`, `nis_cachemgr`, `nisshowcache`, and `nischttl`; and the utilities `rpc.nisd` and `nisinit`.

Chapter 12, "Administering NIS+ Tables", describes how to use the commands that administer NIS+ tables and the information in them, `nistbladm`, `niscat`, `nismatch`, `nisgrep`, `nisln`, `nisping`, and `nisaddent`, including the `nissetup` utility.

Chapter 13, "Problems and Solutions", describes various types of problems that an NIS+ administrator may encounter and how to solve those problems.

Appendices

The appendices provide useful reference material:

Appendix A, "Error Messages", provides an alphabetic listing of the most commonly encountered error messages.

Appendix B, "Information in NIS+ Tables", summarizes the contents of the standard NIS+ tables.

What Typographic Changes and Symbols Mean

The following table describes the type changes and symbols used in this book:.

Table P-1 Typographic Conventions

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

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

AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.
Code samples are included in boxes and may display the following:		
`%`	UNIX™ C shell prompt	`system%`
`$`	UNIX Bourne and Korn shell prompt	`system$`
`#`	Superuser prompt, all shells	`system#`

Part 1 — Setting Up NIS+ Components

This part of the manual provides step-by-step instructions for creating and setting up NIS+ components. It has six chapters.

Setting Up the Root Domain 1

This chapter provides step-by-step instructions for one task, setting up the root domain with DES authentication.

This task describes how to set up the root domain for DES operation; that is, with the root master server running at security level 2. Setting up the root domain involves three major tasks:

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

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

Standard versus NIS-Compatible Setup Procedures

The steps in this section apply to both a standard NIS+ root domain and an NIS-compatible root domain. There are, however, some important differences. The NIS+ daemon for an NIS-compatible domain must be started with the `-Y` option, which allows the root master server to answer requests from NIS clients. This is described in Step 7. The equivalent step for standard NIS+ domains is Step 8.

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 9. The standard NIS+ domain version uses the same command but without the `-Y` option.

Establishing the Root Domain

The procedure describes each step in detail and provides related information. For those who do not need detailed instructions, a summary listing of the necessary commands is provided at the end of the chapter.

Summary of Steps

Here is a summary of the entire setup process:

1. Log on — as superuser — to the root master server.
2. Check the root master server's domain name.
3. Check the root master server's Switch configuration file.
4. Clean out leftover NIS+ material and processes.
5. Name the root domain's admin group.
6. Create the root directory and initialize the root master server.
7. —NIS-Compatiblity Only— Start the NIS+ daemon with -Y.
8. —Standard NIS+ Only—Start the NIS+ daemon.
9. Create the root domain's subdirectories and tables.
10. Create DES credentials for the root master server.
11. Create the root domain's admin group.
12. Add the root master to the root domain's admin group.
13. Update the root domain's public keys.
14. Start the NIS+ cache manager.
15. Restart the NIS+ daemon with security level 2.

16. Add your LOCAL credentials to the root domain.
17. Add your DES credentials to the root domain.
18. Add credentials for other administrators.
19. Add yourself and other administrators to the root domain's admin group.

Security Considerations

NIS+ provides preset security defaults for the root domain. You can override or change these security defaults. To do this while setting up the root domain, see "Overriding Defaults" on page 147.

Prerequisites

- The `/etc/passwd` file on the root master server must contain an entry for you and every other administrator whose credentials will be added to the root domain in this setup process.
- If the server will operate in NIS-compatibility mode and support DNS forwarding for Solaris 1.x clients, it must have a properly configured `/etc/resolv.config` file (described in *Name Services Configuration Guide*.")

Information You Need

- The superuser password of the workstation that will become the root master server (for Step 1)
- The name of the root domain (for Step 2)
- The name of the root domain's admin group (for Step 5)
- Your UID and password
- The UID of any administrator whose credentials you will add to the root domain.

▼ How to Set Up a Root Domain

1. **Log on — as superuser — to the root master server.**
   Log on as *superuser* to the workstation that will become the root master server. The examples in these steps use "rootmaster" as the root master server and "Wiz.Com." as the root domain.

2. **Check the root master server's domain name.**
   Make sure the root master server is using the correct domain name. Use the `domainname` command, as shown below:

```
rootmaster# domainname
Strange.Domain
rootmaster# domainname Wiz.Com
rootmaster# domainname
Wiz.Com
rootmaster# domainname > /etc/defaultdomain
```

   The `domainname` command returns a workstation's current domain name. If the name is not correct, change it. (Complete instructions are provided in Chapter 2, "Setting Up NIS+ Clients") The above example changes the domain name of the root master server from StrangeDomain to Wiz.Com: Make sure that you don't include a trailing dot with the domain name in this instance. The `domainname` command is not an NIS+ command, so it does not follow the NIS+ conventions of a trailing dot. Also make sure that the domain name has at least two labels; e.g., "Wiz.Com" instead of "Wiz."

3. **Check the root master server's Switch configuration file.**
   Make sure the root master server is using the NIS+ version of the `nsswitch.conf` file, even if it will run in NIS-compatibility mode. This step ensures that the primary source of information for the root master will be NIS+ tables. Figure 1-1 on page 6 shows the NIS+ version of the file.

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

   If the root master server's configuration file is different from the one in Figure 1-1 on page 6, change it to the NIS+ version. Complete instructions are provided in Chapter 6, "Setting Up the Name Service Switch," but here is an example:

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

```
rootmaster# more /etc/nsswitch.conf
#
# /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 contains "switch.so" as a nametoaddr library for "inet" transports.

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

# consult /etc "files" only if nisplus is down.
hosts:   nisplus [NOTFOUND=return] files
#Uncomment the following line, and comment out the above, to use both DNS and NIS+
#hosts:   nisplus dns [NOTFOUND=return] files

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

publickey: nisplus

netgroup:  nisplus

automount: files nisplus
aliases:  files nisplus
```

Figure 1-1 NIS+ Version of `nsswitch.conf` File

4. **Clean out leftover NIS+ material and processes.**
   If the workstation you are working on was previously used as an NIS+ server or client, remove any files that might exist in /var/nis and kill the cache manager, if it is still running. In this example, a coldstart file and a directory cache file still exist in /var/nis:

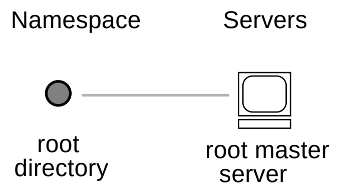
```
rootmaster# ls /var/nis
NIS_COLD_START   NIS_SHARED_CACHE
rootmaster# rm -rf /var/nis/*
rootmaster# ps -ef | grep nis_cachemgr
  root 295  260 10 15:26:58 pts/0 0:00 grep nis_cachemgr
  root 286   1 57 15:21:55 ?   0:01 /usr/sbin/nis_cachemgr
rootmaster# kill -9 286
```

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

5. **Name the root domain's admin group.**
   Although you won't actually create the admin group until Step 11, you need to 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 9.

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

   (To view the workstation's environment variables, use the env command.)



6. **Create the root directory and initialize the root master server.**
   This step creates the first object in the namespace — the root directory — and converts the workstation into the root master server. Use the nisinit -r command, as shown below. (This is the only instance in which you will create a domain's directory object and initialize its master server in one step.

In fact, `nisinit -r` performs an automatic `nismkdir` for the root directory. In any case except the root master, these two processes are performed as separate tasks.)

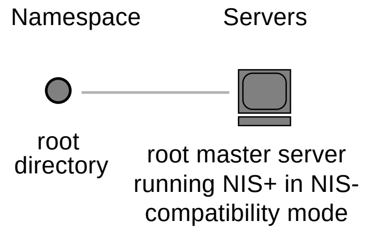
```
rootmaster# nisinit -r

This machine is in the Wiz.Com. NIS+ domain
Setting up root server ...
All done.
```

A UNIX directory with the name of the server (e.g., `rootmaster`) is created under `/var/nis`. Beneath it is a file named `root.object`:

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

This is not the root directory object; it is actually a file that NIS+ uses to describe the root of the namespace for internal purposes. The NIS+ root directory object will be created in the following step.

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



7. **—NIS-Compatibility Only— Start the NIS+ daemon with -Y.**
Perform this step only if you are setting up the root domain in NIS-compatibility mode; if setting up a standard NIS+ domain, perform Step 8 instead. This step includes instructions for supporting the DNS forwarding capabilities of NIS clients.

This step consists of two sub-steps, a and b. Step a starts the NIS+ daemon in NIS-compatibility mode and Step b makes sure that when the server is rebooted, the NIS+ daemon restarts in NIS-compatibility mode. After Step b, skip to Step 9.

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

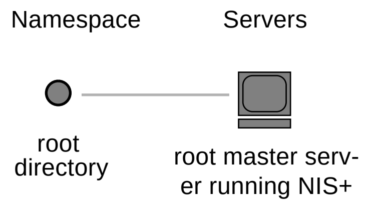
```
rootmaster# rpc.nisd -Y -B -S 0
```

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

b. Edit the `/etc/init.d/rpc` **file.**
Search for the string `EMULYP="Y"` in the `/etc/init.d/rpc` file. Uncomment the line and, to retain DNS forwarding capabilities, add a `-B` flag:

```
rootmaster# vi /etc/init.d/rpc
.
.
.
#   EMULYP="-Y"
.
.
       -------uncomment and change to------

    EMULYP="-Y -B"
```

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



8. **—Standard NIS+ Only—Start the NIS+ daemon.**
Use the `rpc.nisd` and be sure to add the `-S 0` flag.

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

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

As a result of this step (or Step 7), your namespace now has:
- A root directory object (`root_dir`)
- A root master server (rootmaster) running the NIS+ daemon (`rpc.nisd`)
- A coldstart file for the master server (`NIS_COLD_START`)
- A transaction log file (*servername*`.log`)
- A table dictionary file (*servername*`.dict`).

The root directory object is stored in the directory created in Step 6:

```
rootmaster# ls -l /var/nis/rootmaster
-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 (described on page 169):

```
rootmaster# nisls -l Wiz.Com.
Wiz.Com.:
```

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

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

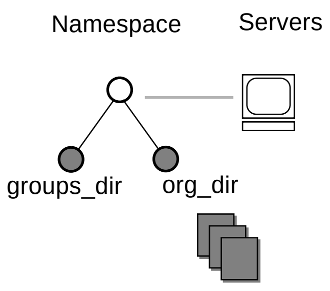
Note that the root directory object provides full (Read, Modify, Create, and Destroy) permissions to both the Owner and the Group, while providing only Read access to the World and Nobody categories. (If your directory object does not provide these rights, you can change them using the nischmod command.)

You can verify that the NIS+ daemon is running, by using the ps command:

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

The root domain's NIS_COLD_START file, which contains the Internet address (and eventually, public keys) of the root master server, is placed in /var/nis. Although there is no NIS+ command that you can use to examine its contents, its contents are loaded into the server's directory cache (NIS_SHARED_DIRCACHE). You can examine those contents with the /usr/lib/nis/nisshowcache command, described on page 180.

Also created are a transaction log file (*servername*.log) and a dictionary file (*servername*.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 (described on page 184). The dictionary file is used by NIS+ for internal purposes; it is of no interest to an administrator.



9. 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:

```
rootmaster# /usr/lib/nis/nissetup -Y     # NIS-compatible only

rootmaster# /usr/lib/nis/nissetup      # Standard NIS+ only
```

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

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

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

Recall that when you examined the contents of the root directory with nisls (in Step 8), it was empty. Now, however, it has two subdirectories:

```
rootmaster# nisls Wiz.Com.
Wiz.Com.:
org_dir
groups_dir
```

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

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

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

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

```
Enter login password: if-you-enter-different-password
nisaddcred: WARNING: password differs from login password.
Retype password:
```

You can persist and retype the same password, and NIS+ will still create the credential. The new password will be stored in `/etc/.rootkey` and used by the keyserver when it starts up. To give the keyserver the new password right away, do a `keylogin -r`, as described in Chapter 8, "Administering NIS+ Credentials."

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

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

If for any reason you need to change the root master's DES credential, you can remove the old credential with the `nisaddcred -r` command, as described in Chapter 8.

As a result of this step, the root server's private and public keys are stored in the root domain's Cred table ("cred.org_dir.Wiz.Com.") and its secret key is stored in `/etc/.rootkey`. You can verify the existence of its credentials in the Cred table by using the `niscat` command (described in Chapter 8.) Since the default domain name is Wiz.Com., you don't have to enter the Cred table's fully-qualified name; the "org_dir" suffix is enough. You can locate the root master's credential by looking for its secure RPC netname.

11. **Create the root domain's admin group.**
This step creates the admin group named in Step 5. Use the `nisgrpadm` with the `-c` option. The example below creates the "admin.Wiz.Com." group:

```
rootmaster# nisgrpadm -c admin.Wiz.Com.
Group "admin.Wiz.Com." created.
```

This step only creates the group — it does not identify its members. That is done in Step 12. To observe the object properties of the group, use `niscat -o`, but be sure to use "`groups_dir`" in the group's name. For example:

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

12. **Add the root master to the root domain's admin group.**
Since at this point, the root master server is the only NIS+ principal that has DES credentials, it is the only member you should add to the admin group. Use the `nisgrpadm` command again, but with the `-a` option. The first

argument is the group name, the second is the name of the root master server. This example adds "rootmaster.Wiz.Com." to the "admin.Wiz.Com." group:

```
rootmaster# nisgrpadm -a admin.Wiz.Com. rootmaster.Wiz.Com.
Added "rootmaster.Wiz.Com." to group "admin.Wiz.Com."
```

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 "targetted" at the `groups_dir` subdirectory.

To verify that the root master is indeed a member of the group, use the `nisgrpadm` command with the `-l` option (see Chapter 10, "Administering NIS+ Groups"):

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

13. **Update the root domain's public keys.**
Normally, directory objects are created by an NIS+ principal who 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 11, "Administering NIS+ Directories.")

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, as shown below:

```
rootmaster# /usr/lib/nis/nisupdkeys Wiz.Com.
rootmaster# /usr/lib/nis/nisupdkeys org_dir.Wiz.Com.
rootmaster# /usr/lib/nis/nisupdkeys groups_dir.Wiz.Com.
```

After each instance, you'll get a confirmation message such as this one:

```
Fetch Public key for server rootmaster.Wiz.Com.
  netname = 'unix.rootmaster@Wiz.Com.'
Updating rootmaster.Wiz.Com.'s public key.
  Public key : public-key
```

Now, if you look in any of those directories (use `niscat` -o), you'll see this entry under the "Public Key:" field:

```
Public Key: Diffie-Hellman (196 bits)
```

14. **Start the NIS+ cache manager**
    The cache manager maintains a local cache of location information for an NIS+ client (in this case, the root master server). It obtains its initial set of information from the client's coldstart file (created in Step 7 or Step 8), and downloads it into a file named `NIS_SHARED_DIRCACHE` in `/var/nis`.

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

    ```
    rootmaster# nis_cachemgr
    ```

    Once the cache manager has been started, you only need to restart it if you have explicitly killed it. You don't need to restart it if you reboot, since the coldstart file in `/var/nis` starts it automatically when the client is rebooted. For more information about the NIS+ cache manager, see Chapter 11, "Administering NIS+ Directories."

15. **Restart the NIS+ daemon with security level 2.**
    Now that the root master server has DES credentials and the root directory object has a copy of the root master's public key, you can restart the root master with security level 2. First kill the existing daemon, then restart one with security level 2. For an NIS-compatible root domain, be sure to use the `-Y` flag (and `-B` flag). Here is an example:

    ```
    rootmaster# ps -e | grep rpc.nisd
    1081 ?     0:03 rpc.nisd
    rootmaster# kill 1081
    rootmaster# rpc.nisd          # Standard NIS+ domain only
    rootmaster# rpc.nisd -Y -B       # NIS-compatible NIS+ domain
    ```

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

16. **Add your LOCAL credentials to the root domain.**
Since you don't have access rights to the root domain's Cred table, you must perform this operation as superuser. In addition, as mentioned in "*Prerequisites*," the root master's `/etc/passwd` file must contain an entry for you. Use the `nisaddcred` command with the `-p` and `-P` flags. Here is the syntax, followed by an example:

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

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

```
rootmaster# nisaddcred -p 11177 -P topadmin.Wiz.Com. local
```

For more information about the `nisaddcred` command, see Chapter 8.

17. **Add your DES credentials to the root domain.**
Use the `nisaddcred` command again, but with the following syntax:

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

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

```
rootmaster# nisaddcred -p unix.11177@Wiz.Com \
            -P topadmin.Wiz.Com. des
Adding key pair for unix.11177@Wiz.Com (topadmin.Wiz.Com.).
Enter login password: enter-login-password
```

If after entering your login password you get a warning such as this one . . .

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

. . . and yet the password you entered is your correct login password, ignore the error message. The message appears because NIS+ cannot read the protected `/etc/shadow` file which stores the password, as expected. The message would not have appeared if you had no user password information stored in the `/etc/passwd` file.

18. **Add credentials for other administrators.**
    Add the credentials, both LOCAL and DES, of the other administrators who will work in the root domain. You can do this in 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 that adds credentials for an administrator with a UID of 33355 and a principal name of “bobee.Wiz.Com.”

```
rootmaster# nisaddcred -p 33355 -P bobee.Wiz.Com. local
rootmaster# nisaddcred -p unix.33355@Wiz.Com \
            -P bobee.Wiz.Com. des
Adding key pair for unix.33355@Wiz.Com (bobee.Wiz.Com.).
Enter login password: enter-bobee’s-login-password
```

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

```
rootmaster# /usr/lib/nis/nisaddent -a -f \
      /etc/passwd.xfr passwd
rootmaster# /usr/lib/nis/nisaddent -a -f \
      /etc/shadow.xfr shadow
rootmaster# nisaddcred -p 33355 -P bobee.Wiz.Com. local
rootmaster# nisaddcred -p unix.33355@Wiz.Com \
            -P bobee.Wiz.Com. des
Adding key pair for unix.33355@Wiz.Com (bobee.Wiz.Com.).
Enter bobee’s login password: enter-dummy-password
nisaddcred: WARNING: password differs from login passwd.
Retype password: re-enter-dummy-password
```

The first instance of `nisaddent` populates the Passwd table —except for the actual password column. The second instance populates the shadow column. Each administrator can later change his or her network password using the `chkey` command. Chapter 8, “Administering NIS+ Credentials,” describes how to do this.

19. **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 bobee, to the "admin.Wiz.Com." group:

```
rootmaster# nisgrpadm -a admin.Wiz.Com. topadmin.Wiz.Com. \
              bobee.Wiz.Com.
Added "topadmin.Wiz.Com." to group "admin.Wiz.Com.".
Added "bobee.Wiz.Com." to group "admin.Wiz.Com.".
```

This step completes this task. A summary of the entire task is provided below.

Summary

Below is a summary of the steps required to set up the 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. Also, this summary does not show the server's responses to each command.

`rootmaster% su` `Password:` ***enter-root-password***	1. Log on as superuser to rootmaster.
`# domainname`	2. Check domain name
`# more /etc/nsswitch.conf`	3. Check Switch file.
`# rm -rf /var/nis*`	4. Remove leftover NIS+ material.
`# NIS_GROUP=admin.Wiz.Com.;export NIS_GROUP`	5. Name the admin group.
`# nisinit -r`	6. Initialize the root master.
	7. NIS-compat only:
`# rpc.nisd -Y -B -S 0`	a. Start daemon with -Y -B, S 0.
`# vi /etc/inet.d/rpc`	b. Change to EMULYP="-Y -B".
`# rpc.nisd -S 0`	8. NIS+ Only: Start daemon with S 0.
`# /usr/lib/nis/nissetup [ -Y ]`	9. Create org_dir, groups_dir, tables.
`# nisaddcred des` `Enter login password:` ***enter-login-password***	10. Create DES credentials for root master.
`# nisgrpadm -c admin.Wiz.Com.`	11. Create admin group.
`# nischmod g+rmcd Wiz.Com.`	12. Assign full group rights to root directory
`# nisgrpadm -a admin.Wiz.Com. rootmaster.Wiz.Com.`	13. Add root master to admin group.
`# /usr/lib/nis/nisupdkeys Wiz.Com.`	14. Update root directory's keys.
`# /usr/lib/nis/nisupdkeys org_dir.Wiz.Com.`	Update org_dir's keys.
`# /usr/lib/nis/nisupdkeys groups_dir.Wiz.Com.`	Update groups_dir's keys.
`# nis_cachemgr`	15. Start NIS+ cache manager
`# ps -ef | grep rpc.nisd`	16. Restart the NIS+ daemon w/ sec 2.
`# kill -9` ***process-id***	First kill existing daemon.
`# rpc.nisd [ -Y ] [ -B ]`	Use -Y for NIS compat & -B for DNS.
`# nisaddcred -p 11177 -P topadmin.Wiz.Com. local`	17. Add your LOCAL credentials.

`# nisaddcred -p unix.11177@Wiz.Com \`
`      -P topadmin.Wiz.Com. des`
`Enter login password:` ***enter-login-password***
	18. Add your DES credentials.

`# nisaddcred . . .`	19. Add credentials for other admins.
`# nisgrpadm -a admin.Wiz.Com.` ***member ...***	20. Add other admins to admin group.

Figure 1-2 Summary of Steps to Set Up the Root Domain

Setting Up NIS+ Clients 2

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

You can also use Administration Tool's Host Manager to add and set up an NIS+ client machine.

This chapter describes how to set up clients in a standard NIS+ domain and in a NIS-compatible domain. At Step 7 in the client set up instructions you must choose which of three methods to use: broadcast, hostname, or coldstart file. Since each method is implemented differently, each has its own task description. After initializing a client by one of these methods, you can continue setting up the client by returning to Step 8.

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 at the end of the chapter.

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

2

Setting Up NIS+ Clients

This task describes how to set up a typical NIS+ client, whether in the root domain or in a non-root domain. It 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. Setting up an NIS+ client involves the following tasks:

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

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

1. Log on to the domain's master server.
2. Create DES credentials for the new client workstation.
3. Log on —as superuser—to the client.
4. Assign the client its new domain name.
5. Check the client's Switch configuration file.
6. Clean out leftover NIS+ material and processes.
7. Initialize the client.
8. Kill and restart the keyserv daemon.
9. Run keylogin.
10. Reboot the client.

Security Considerations

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

Once an administrator creates the client's credentials, the client can complete the setup process. However, the client still needs Read access to the directory object of its home domain. If you set up the client's home domain according to the instructions in either Chapter 1 or Chapter 4, 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. It displays the properties of the directory, including its access rights. Here is an example:

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

You can change the directory object's access rights, provided you have Modify rights to it yourself, by using the `nischmod` command, described in Chapter 9, "Administering NIS+ Access Rights."

Prerequisites

- The administrator setting up the client's credentials must have valid DES credentials and Modify rights to the Cred table in the client's home domain.
- The client must have Read rights to the directory object of its home domain.
- The client's home domain must already be set up and running NIS+.
- The client must have an entry either in the master server's `/etc/hosts` file or in its domain's Hosts table.

Information You Need

- The name of the client's home domain

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

▼ How to Set up an NIS+ Client

1. **Log on to the domain's master server.**
   You can log on as superuser or as yourself, depending on which NIS+ principal has the proper access rights to add credentials to the domain's Cred table.

2. **Create DES credentials for the new client workstation.**
   Use the `nisaddcred` command with the `-p` and `-P` arguments. Here is the syntax, followed by an example:

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

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

   This example adds a DES credential for a client workstation named client1 in the Wiz.Com. domain:

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

   For more information about the `nisaddcred` command, see Chapter 8, "Administering NIS+ Credentials."

3. **Log on —as superuser— to the client.**
   Now that the client workstation has credentials, you can log out of the master server and begin working from the client itself. You can do this locally or remotely.

4. **Assign the client its new domain name.**
   Assign the client its new domain name, (see “Changing a Client’s Domain Name”) using the task listed below. Then return to Step 5.

5. **Check the client's Switch configuration file.**
   Make sure the client is using the NIS+ version of the `nsswitch.conf` file. This ensures that the primary source of information for the client will be NIS+ tables. Figure 2-1 shows the correct version of the file.

```
client1# more /etc/nsswitch.conf
#
# /etc/nsswitch.nisplus:
#
# An example file that could be copied over to /etc/nsswitch.conf; it
# uses NIS+ (NIS Version 3) in conjuction with files.
#
# "hosts:" and "services:" in this file are used only if the /etc/netconfig
# file contains "switch.so" as a nametoaddr library for "inet" transports.

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

# consult /etc "files" only if nisplus is down.
hosts:      nisplus [NOTFOUND=return] files
#Uncomment the following line, and comment out the above, to use both DNS and NIS+
#hosts:      nisplus dns [NOTFOUND=return] files

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

publickey:  nisplus

netgroup:   nisplus


automount:  files nisplus
aliases:    files nisplus
```

Figure 2-1 NIS+ Version of `nsswitch.conf` File

If the file does not look like the one above, change it to the version recommended for NIS+. Complete instructions are provided in Chapter 6, "Setting Up the Name Service Switch," but here is an example.

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

Although the instructions in Chapter 6 tell you to kill and restart the keyserver, you don't need to at this point, since you'll do so in Step 8.

6. **Clean out leftover NIS+ material and processes.**
   If the workstation you are working on was previously used as an NIS+ server or client and you no longer want to retain those bindings, remove any files that might exist in `/var/nis` and kill the cache manager, if it is still running. In this example, a coldstart file and a directory cache file still exist in `/var/nis`:

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

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

7. **Initialize the client.**
   You can initialize a client in three different ways: by hostname, by coldstart file, or by broadcast. (See "Initializing an NIS+ Client by Broadcast", "Initializing an NIS+ Client by Hostname", or "Initializing an NIS+ Client by Coldstart File".) Select a method then proceed with Step 8.

8. **Kill and restart the keyserv daemon.**
   The following step stores the client's secret key on the keyserver. Before that can be done, you must kill and restart the keyserv daemon. This also has the side effect of updating the key server's Switch information about the client.

   First kill the keyserv daemon, then remove the `/etc/.rootkey` file, then restart the keyserver. Here is an example:

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

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

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

10. **Reboot the client.**
    This step completes this task.

Initializing an NIS+ Client by Broadcast

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

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

Security Considerations

You must perform this task as superuser on the client.

Prerequisites

At least one NIS+ server must exist on the same subnet as the client.

Information You Need

You need the superuser password to the client.

▼ How to Initialize an NIS+ Client — Broadcast Method

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

   ```
   Client1# nisinit -c -B
   This machine is in the Wiz.Com. NIS+ domain.
   Setting up NIS+ client ...
   All done.
   ```

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

This step completes this task.

Initializing an NIS+ Client by Hostname

Initializing a client by hostname 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 coldstart file.

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

Security Considerations

You must perform this operation as superuser on the client.

Prerequisites

- The NIS+ service must be running in the client's domain.
- The client must have an entry in its `/etc/hosts` file for the trusted server.

Information You Need

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

▼ How to Initialize an NIS+ Client — Hostname Method

1. **Check the client's `/etc/hosts` file.**
   Make sure the client has an entry for the trusted server.

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

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

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

   This step completes this task.

Initializing an NIS+ Client by Coldstart File

This task initializes an NIS+ client by using the coldstart file of another NIS+ client — preferably 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 hostname or broadcast method.

Security Considerations

You must perform this task as superuser on the client.

Prerequisites

The servers specified in the coldstart file must already be set up and running NIS+.

Information You Need

You need the name and location of the coldstart file you will copy.

▼ How to Initialize an NIS+ Client — Coldstart Method

1. **Copy the other client's coldstart file.**
   Copy the other client's coldstart 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` though, because during initialization that file gets overwritten. This example copies the coldstart file of "client1" into the `/tmp` directory of "client2."

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

2. **Initialize the client from the coldstart file.**
   Use the `nisinit` command with the `-c` and `-C` options, as shown below.

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

   This step completes this task.

Changing a Client's Domain Name

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

Specifying a Domain Name After Installation

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

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

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

Security Considerations

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

Information You Need

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

▼ How to Change a Client's Domain Name

1. **Log on to the workstation and become superuser.**
   The examples in this task use "client1" as the workstation and "Wiz.Com." as the new domain name.

```
client1% su
Password: enter-password
```

2. **Change the workstation's domain name.**
   Enter the new name with the `domainname` command, as shown below. Do not use a trailing dot.

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

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

3. **Verify the result.**
   Use the `domainname` command again, this time without an argument, to display the server's current domain.

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

4. **Save the new domain name.**
   Redirect the output of the `domainname` command into the `/etc/defaultdomain` file.

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

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

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

   This step completes this task.

NIS+ Client Set Up Summary

Figure 2-2 shows a summary of the steps required to set up a client. It assumes the simplest case, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. For brevity, this summary does not show the responses to each command.

```
rootmaster%                                     1. Log on to domain’s master.
rootmaster% nisaddcred -p unix.client1.Wiz.Com \ 2. Create DES credentials for client.
            -P client1.Wiz.Com. des
client1% su                                     3. Log on, as superuser, to the client.
Password: enter-password
client1# domainname Wiz.Com                     4. Assign the client a domain name.
client1# domainname  /etc/defaultdomain
client1# more /etc/nsswitch.conf                5. Check the swich configuration file.
client1# rm -rf /var/nis/*                      6. Clean out /var/nis.
client1# nisinit -c -H rootmaster               7. Initialize the client.
client1# ps -ef | grep keyserv                  8. Kill and restart the keyserver.
client1# kill -9 process-id
client1# keyserv
client1# keylogin -r                            9. Keylogin the client.
Password: enter-superuser-password
client1# init 6                                10. Reboot the client.
```

Figure 2-2 Summary: How to Set Up a Client

Setting Up NIS+ Servers 3

This chapter provides step-by-step procedures for three server-related tasks:

The first task describes how to set up an NIS+ server. The second describes how to add a server to an existing domain, whether root or non-root. The third describes how to change a server's security level, whether to upgrade it for normal DES operation or to downgrade it for debugging.

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

Setting Up an NIS+ Server

This task applies to any NIS+ server except the root master; that is, to a root replica, a non-root master, or a non-root replica, whether running in NIS-compatibility mode or not.

Standard versus NIS-Compatible Setup Procedures

The differences between an NIS-compatible and a standard NIS+ server are the same as for the root master server. The NIS+ daemon for an NIS-compatible server must be started with the `-Y` option (and the `-B` option for DNS forwarding), which allows the server to answer requests from NIS clients. This is described in Step 2. The equivalent step for standard NIS+ servers is Step 3.

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

As you may recall, the instructions for setting up the root master server in NIS-compatibility mode (Chapter 1) also required a `-Y` flag with the `nissetup` utility, which creates the NIS+ tables with the proper permissions for an NIS-compatible domain. That step is not included in this task because with every server other than the root master, the `nissetup` utility is not used until the server is associated with a domain (as described in Chapter 4, “Setting Up a Non-Root Domain.).

Here is a summary of the entire setup process:

1. Log on as superuser to the new replica server.
2. —NIS-Compatiblity Only— Start the NIS+ daemon with -Y.
3. —Standard NIS+ Only— Start the NIS+ daemon.

Security Considerations

You must perform this operation as superuser on the server. The security level at which you start the server (Step 4) determines the credentials that its clients must have. For instance, if the server is set up with security level 2, the clients in the domain it supports must have DES credentials. If you have set up the client according to the instructions in this book, the client has DES credentials in the proper domain, and you can start the server with security level 2.

Prerequisites

- The root domain must already be set up — see Chapter 1, “Setting Up the Root Domain.”
- The server must have already been initialized as an NIS+ client — see Chapter 2, “Setting Up NIS+ Clients.”
- If the server will run in NIS-compatibility mode and support DNS forwarding, it must have a properly configured `/etc/resolv.config` file (described in *Name Services Configuration Guide.*”)

Information You Need

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

▼ How to Set Up an NIS+ Server

1. **Log on as superuser to the new replica server.**
   The following steps assume you rebooted the workstation after you set it up as an NIS+ client, as instructed in “Setting Up NIS+ Clients” on page 26”. Rebooting, among other things, starts the cache manager, which is a recommended prerequisite to the following step. If you did not reboot the workstation, restart the cache manager now, using `nis_cachemgr`.

2. **—NIS-Compatibility Only— Start the NIS+ daemon with -Y.**
   Perform this step only if you are setting up the server in NIS-compatibility mode; if setting up a standard NIS+ server, perform Step 3 instead. This step includes instructions for supporting the DNS forwarding capabilities of NIS clients.

   This step consists of two sub-steps, a and b. Step a starts the NIS+ daemon in NIS-compatibility mode and Step b makes sure that when the server is rebooted, the NIS+ daemon restarts in NIS-compatibility mode.

a. Use `rpc.nisd` with the `-Y` and `-B` flags.

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

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

b. Edit the `/etc/init.d/rpc` file.
Search for the string `EMULYP="-Y"` in the `/etc/init.d/rpc` file. Uncomment the line and, to retain DNS forwarding capabilities, add a `-B` flag:

```
compatserver# vi /etc/init.d/rpc
.
.
.
#      EMULYP="-Y"
.
.
            -------uncomment and change to------

         EMULYP="-Y -B"
```

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

This step creates a directory with the same name as the server and the server's `.log` file. They are placed in `/var/nis`, as you can see:

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

The `compatserver.log` file is a transaction log. You can examine the contents of the transaction log by using the `nislog` command, described in Chapter 11, "Administering NIS+ Directories."

3. **—Standard NIS+ Only— Start the NIS+ daemon.**
   Use the `rpc.nisd` command.

```
server# rpc.nisd
```

To verify that the NIS+ daemon is indeed running, use the `ps` command, as shown below:

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

This step creates a directory with the same name as the server and the server's `.log` file. They are placed in `/var/nis`, as you can see:

```
server# ls -F /var/nis
NIS_COLD_START   server/  server.log
```

The `compatserver.log` file is a transaction log. You can examine the contents of the transaction log by using the `nislog` command, described in Chapter 11, "Administering NIS+ Directories."

Now this server is ready to be designated a master or replica of a domain, as described in Chapter 4, "Setting Up a Non-Root Domain." This step completes this task. A task summary is provided at the end of the chapter.

Adding a Replica to an Existing Domain

This section describes how to add a server using the raw NIS+ commands. An easier way to add a replica server to an existing domain is to use the `nisserver` script as described in *Name Services Configuration Guide.*.

This task describes how to add a replica server to an existing domain, whether root or non-root. Here is a list of the steps:

1. Log on to the domain's master server.
2. Add the replica to the domain.
3. Ping the replica.

Security Considerations

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

Prerequisites

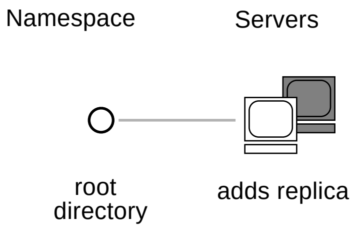
- The server that will be designated a replica must have already been set up.
- The domain must have already been set up and assigned a master server.

Information You Need

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

▼ How to Add a Replica Server

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



2. **Add the replica to the domain.**
   Use the `nismkdir` command with the `-s` option, as shown in the example below. The example adds the replica "rootreplica" to the "Wiz.Com." domain.

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

   When you use the `nismkdir` command on a directory object that already exists, it does not recreate the directory, it simply modifies it according to the flags you provide. In this case, the `-s` flag assigns the domain an additional replica server. You can verify that the replica was added by examining the directory object's definition, using the `niscat -o` command.

3. **Ping the replica**
   This step sends a message (a "ping") to the new replica, telling it to ask the master server for an update. If the replica does not belong to the root domain, be sure to specify its domain name. (The example below includes

the domain name only for completeness; since the example used throughout this task adds a replica to the root domain, the “Wiz.Com.” domain name in the example below is not necessary.)

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

You should see results similar to these:

```
rootmaster# /usr/lib/nis/nisping Wiz.Com.
Pinging replicas serving directory Wiz.Com. :
Master server is rootmaster.Wiz.Com.
        Last update occured at Wed Nov 25 10:53:37 1992

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

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

If you have set up the domain’s tables immediately after completing the domain setup, this step propagates the tables down to the replica. For more information about `nisping`, see Chapter 11, “Administering NIS+ Directories.”

This step completes this task. A summary is provided at the end of this chapter.

Changing a Server’s Security Level

This task changes the security level of a previously set up NIS+ server. You can assign it security level 0 (lowest), 1, or 2 (highest). The default is 2. Here is a list of the steps:

1. Log on —as superuser— to the server.
2. Kill the NIS+ daemon.
3. Restart the NIS+ daemon with the desired security level.

Security Considerations

You must perform this task as superuser on the server. If changing to security level 1, at least one NIS+ principal must have LOCAL credentials in the Cred table of the server's home domain. Otherwise, the server will be unable to authenticate anyone and no one will be able to operate on that domain. If changing to security level 2, at least one NIS+ principal must have DES credentials in the domain's Cred table. Security level 0 requires no credentials.

▼ How to Change a Server's Security Level

1. **Log on —as superuser— to the server.**

If you started `rpc.nisd` with the `-Y` or `-B` option, you must also kill the `rpc.nisd_resolv` daemon.

2. **Kill the NIS+ daemon.**
   Find the daemon's process ID and then kill it, using the `kill` command as shown below. Note that this will interrupt NIS+ service.

```
server# ps -e | grep rpc.nisd
root 1081     1  61  16:43:33  ?      0:01  rpc.nisd
root 1386  1004  11  16:44:09  pts/1  0:00  grep rpc.nisd
server# kill 1081
```

   If you reinvoke the `ps` command, it should no longer list the daemon process.

```
server# ps -ef | grep rpc.nisd
root  1094  1004  11  16:54:28  pts/1  0:00 grep rpc.nisd
```

3. **Restart the NIS+ daemon with the desired security level.**
   Use the `rpc.nisd` command as shown below.

```
server# rpc.nisd -S security-level
```

   Security-level can be 0 (lowest), 1, or 2 (highest). Security level 2 is the default; to select it, you don't have to use the `-S` option. If the server is running in NIS-compatibility mode, make sure you use the `-Y` (and `-B` for DNS forwarding) options.

   To verify that the NIS+ daemon is indeed running, use the `ps` command, as shown below:

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

   This step completes this task. A summary is provided below.

Summary

Below is a summary of the tasks described in this chapter. 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. Also, this summary does not show the server's responses to each command.

`server%` **`su`**	1. Log on to the server.
	2. NIS-compat only:
`compatserver#` **`rpc.nisd -Y - B`**	a. Start daemon with -Y -B
`compatserver#` **`vi /etc/inet.d/rpc`**	b. Change to EMULYP="-Y -B"
`server#` **`rpc.nisd`**	3. NIS+-Only: Start daemon.

Figure 3-1 To Set up an NIS+ Server

`rootmaster%` **`su`**	1. Log on as superuser to domain master.
`#` **`nismkdir -s rootreplica Wiz.Com.`**	2. Designate the new replica.
`#` **`nismkdir -s rootreplica org_dir.Wiz.Com.`**	
`#` **`nismkdir -s rootreplica groups_dir.Wiz.Com.`**	
`#`**`/usr/lib/nis/nisping Wiz.Com`**	3. Ping the replica.
`#`**`/usr/lib/nis/nisping org_dir.Wiz.Com`**	
`#`**`/usr/lib/nis/nisping groups_dir.Wiz.Com`**	

Figure 3-2 To Add a Replica to an Existing Domain

`server%` **`su`**	1. Log on as superuser to the server.
`server#` **`ps -ef | grep rpc.nisd`**	2. Kill the daemon.
`server#` **`kill`** ***process-id***	
`server#` **`rpc.nisd -S`** ***security-level***	3. Restart the NIS+ daemon. Use -r for root domain servers. Use -Y and -B as appropriate

Figure 3-3 To Change a Server’s Security Level

Setting Up a Non-Root Domain 4

This chapter provides step-by-step instructions for setting up a non-root domain (also known as a subdomain). You should not set up a non-root domain until *after* you have set up its servers.

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

Setting Up a Non-Root Domain

This task describes how to set up a non-root domain, whether in NIS-compatibility mode or in standard NIS+ mode. 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

However, as with setting up the root domain, these tasks cannot be performed sequentially. To make the set up process easier to execute, they have been broken down into individual steps, and the steps have been arranged into the most efficient order.

Standard versus NIS-Compatible Setup Procedures

The differences between an NIS-compatible and a standard NIS+ server are the same as for the root domain. The NIS+ daemon for each server in an NIS-compatible domain should have been started with the `-Y` option, as instructed in Chapter 3.

An NIS compatible domain also requires its tables to provide Read rights for the Nobody class, which allows NIS clients to access the information stored in them. This is accomplished with the `-Y` option to the `nissetup` command, in Step 4. The standard NIS+ domain version uses the same command but without the `-Y` option, so it is described in the same step.

Here is a summary of the entire setup process:

1. Log on 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

In most sites, to preserve the security of the parent domain, only the parent domain's master server or an administrator who belongs to the parent domain's admin group is allowed to create a domain beneath it. Although this is a policy decision and not a requirement of NIS+, the instructions in this chapter assume that you are following that policy. Of course, the parent domain’s admin group must have Create rights to the parent directory object. To verify this, use the `niscat -o` command:

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

If you are more concerned about convenience than security, you could simply make the new domain’s master server a member of its parent domain’s admin group and then perform the entire procedure from the server. Use the `nisgrpadm` command, described in Chapter 10, “Administering NIS+ Groups.”

Prerequisites

- The parent domain must be set up and running.
- The server that will be designated as this domain’s master must be initialized and running NIS+.
- If you will designate a replica server, the master server must be able to obtain the replica’s IP address through its `/etc/hosts` file or from its NIS+ Hosts table.

Information You Need

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

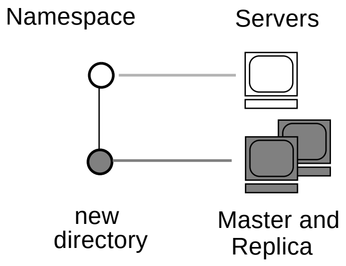
▼ How to Set Up a Non-Root Domain

1. **Log on to the domain's master server.**
   Log on to the server that you will designate as the new domain's master. The steps in this task use the server named "salesmaster," which belongs to the Wiz.Com. domain, and will become the master server of the Sales.Wiz.Com. domain. The administrator performing this task is "topadmin.Wiz.Com.," a member of the admin.Wiz.Com. group. That group has full access rights to the Wiz.Com. directory object.

2. **Name the domain's administrative group.**
   Although you won't actually create the admin group until Step 5, you need to identify it now. This enables the `nismkdir` command used in the following step to create the directory object with the proper access rights for the group. It does the same for the `nissetup` utility used in Step 4.

   Set the value of the environment variable NIS_GROUP to the name of the domain's admin group. Here are two examples, one for `csh` users, and one for `sh/ksh` users. They both set NIS_GROUP to "admin.Sales.Wiz.Com."

```
salesmaster# setenv NIS_GROUP admin.Sales.Wiz.Com.    # for csh

salesmaster# NIS_GROUP=admin.Sales.Wiz.Com.          # for sh/ksh
salesmaster# export NIS_GROUP                        # for sh/ksh
```



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
```

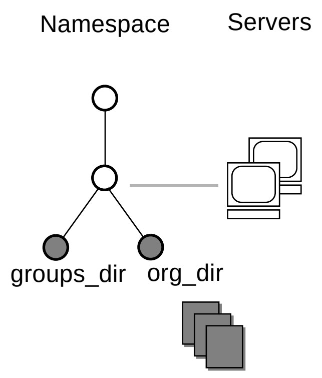
   The `-m` flag designates its master server, and the `-s` flag designates its replica. Here is an example:

```
salesmaster# nismkdir -m salesmaster -s salesreplica \
             Sales.Wiz.Com.
```

The directory is loaded into /var/nis, but to view it, use the niscat -o command:

```
salesmaster# niscat -o Sales.Wiz.Com.
Object Name   : Sales
Owner         : topadmin.Wiz.Com.
Group         : admin.Sales.Wiz.Com.
Domain        : Wiz.Com.
Access Rights : ----rmcdr---r---
.
.
.
```

Unlike the root directory, this directory object *does* have the proper group assignment. As a result, you won’t have to use nischgrp.



4. **Create the domain's subdirectories and tables.**
   This step adds the "org_dir" and "groups_dir" directories, and the NIS+ tables, beneath the new directory object. Use the `nissetup` utility, but be sure to add the new domain name. And, for an NIS-compatible domain, include the `-Y` flag. Here are examples of both versions:

```
salesmaster# /usr/lib/nis/nissetup -Y Sales.Wiz.Com.

    —OR—

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

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

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

   The `-Y` option creates the same tables and subdirectories as for a standard NIS+ domain, but assigns Read rights to the Nobody class so that requests from NIS clients, which are unauthenticated, can access information in the NIS+ tables.

   If you are curious, you can verify the existence of the org_dir and groups_dir directories by looking in your master's equivalent of `/var/nis/salesmaster`. They are listed along with the root object and other NIS+ files. 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 5 (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. The example below creates the "admin.Sales.Wiz.Com." group:

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

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

6. **Assign full group access rights to the directory object.**
   By default, the directory object only grants its group Read access, which makes the group no more useful than the World class. To make the setup of clients and subdomains easier, change the access rights that the directory object grants its group from Read to Read, Modify, Create, and Destroy. Use the `nischmod` command, as shown below:

```
salesmaster# nischmod g+rmcd Sales.Wiz.Com.
```

   Complete instructions for using the `nischmod` command are provided in Chapter 9, "Administering NIS+ Access Rights."

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 rest are the names of the new members. This example adds "salesmaster.Wiz.Com." and "salesreplica.Wiz.Com." to the "admin.Sales.Wiz.Com." group:

```
salesmaster# nisgrpadm -a admin.Sales.Wiz.Com. \
             salesmaster.Wiz.Com. salesreplica.Wiz.Com.
Added "salesmaster.Wiz.Com." to group "admin.Sales.Wiz.Com."
Added "salesreplica.Wiz.Com." to group "admin.Sales.Wiz.Com."
```

   To verify that the servers are indeed members of the group, use the `nisgrpadm` command with the `-l` option (see Chapter 10, "Administering NIS+ Groups"):

```
salesmaster# nisgrpadm -l admin.Sales.Wiz.Com.
Group entry for "admin.Sales.Wiz.Com." group:
    Explicit members:
        salesmaster.Wiz.Com.
        salesreplica.Wiz.Com.
    No implicit members
    No recursive members
    No explicit nonmembers
    No implicit nonmembers
    No recursive nonmembers
```

8. **Add credentials for other administrators.**
   Add the credentials of the other administrators who will work in the domain.

   For administrators who already have DES credentials in another domain, simply add LOCAL credentials. Use the `nisaddcred` command with both the `-p` and the `-P` flags. For example:

```
salesmaster# nisaddcred -p 33355 -P topadmin.Wiz.Com. local
```

   For administrators that do not yet have credentials, you can proceed in two different ways.

- One way is to ask them to add their own credentials. However, they will have to do this as superuser. Here is an example in which an administrator with a UID of 22244 and a principal name of "moe.Sales.Wiz.Com."adds his own credentials to the Sales.Wiz.Com. domain.

```
salesmaster# nisaddcred -p 22244 -P moe.Sales.Wiz.Com. local
salesmaster# nisaddcred -p unix.22244@Sales.Wiz.Com \
                        -P moe.Sales.Wiz.Com. des
Adding key pair for unix.22244@Sales.Wiz.Com.
Enter login password:   enter-moe's-login-password
```

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

```
salesmaster# nisaddcred -p 22244 -P moe.Sales.Wiz.Com. local
salesmaster# nisaddcred -p unix.22244@Sales.Wiz.Com \
                        -P moe.Sales.Wiz.Com. des
Adding key pair for unix.22244@Sales.Wiz.Com.
Enter moe's login password: enter-dummy-password
nisaddcred: WARNING: password differs from login passwd.
Retype password: re-enter-dummy-password
```

Each administrator can later change his or her network password using the `chkey` command. Chapter 8, "Administering NIS+ Credentials," describes how to do this.

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, the remaining arguments are the names of the administrators. This example adds the administrator "moe" to the "admin.Sales.Wiz.Com." group:

```
salesmaster# nisgrpadm -a admin.Sales.Wiz.Com. \
                       moe.Sales.Wiz.Com.
Added "moe.Sales.Wiz.Com." to group "admin.Sales.Wiz.Com.".
```

   This step completes this task. A summary of this task is provided below.

Summary

This is a summary of the steps required to set up a non-root domain. It assumes the simplest case, so be sure you are familiar with the more thorough task descriptions before you use this summary as a reference. Also, this summary does not show the server's responses to each command.

```
salesmaster% su                                     1. Log on as superuser to domain master.
# NIS_GROUP=admin.Sales.Wiz.Com.                    2. Name the domain's admin group.
# export NIS_GROUP                                  3. Create the domain's directory and
# nismkdir -m salesmaster -s salesreplica \            designate its servers.
          Sales.Wiz.Com.                            4. Create org_dir, groups_dir, and
# /usr/lib/nis/nissetup Sales.Wiz.Com.                 tables.
# nisgrpadm -c admin.Sales.Wiz.Com.                    For NIS-compatibility, use -Y.
# nischmod g+rmcd Sales.Wiz.Com.                    5. Create the admin group.
                                                    6. Assign full group rights to the
# nisgrpadm -a admin.Sales.Wiz.Com. \                  domain's directory.
           salesmaster.Wiz.Com. \                   7. Add servers to admin group.
           salesreplica.Wiz.Com.
# nisaddcred -p 22244 -P moe.Sales.Wiz.Com. local   8. Add credentials for other admins.
# nisaddcred -p unix.22244@Sales.Wiz.Com. \
            -P moe.Sales.Wiz.Com. des
# nisgrpadm -a admin.Sales.Wiz.Com. \               9. Add admins to domain's admin group.
           moe.Sales.Wiz.Com.
```

Figure 4-1 Summary of Steps of How to Set Up a Non-Root Domain

Setting Up NIS+ Tables 5

You can populate NIS+ tables in four ways: from files or from NIS maps, via the `nispopulate` script, or with Administration Tool's Database Manager. The easiest method is to use the `nispopulate` script as described in *Name Services Configuration Guide.*

This chapter provides step-by-step instructions for populating NIS+ tables from `/etc` files or NIS maps. The tables should have already been created in the process of setting up a domain, either root or non-root. 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 would already be available in the domain's tables. If you waited till later, you would have to add the clients' information into the master's `/etc` files.

This chapter also describes how to transfer information back from NIS+ tables to NIS maps, a procedure that may be required during a transition from NIS to NIS+. Finally, it includes two tasks that describe how to limit access to the passwd column of the Passwd table:

Replace, Append, and Merge

When you populate a table — whether from a file or an NIS map — 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 vastly different from those of the table, the merge option can spare the server a great many operations. Because it only deletes 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.

Populating NIS+ Tables From Files

This task transfers the contents of an ASCII file, such as `/etc/hosts`, into an NIS+ table.

Here is a list of the steps:

1. Log on to an NIS+ client.
2. Add `/usr/lib/nis` to the search path for this shell.
3. Use `nisaddent` to transfer any of these files one at a time: aliases, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`.
4. Transfer the `publickey` file.

5. Transfer the automounter information.
6. Checkpoint the tables.

Security Considerations

You can perform this task from any NIS+ client, including the root master server, as long as you have the proper credentials and access rights. If you are going to replace or merge the entries in the table with the entries from the text file, you must have Create and Destroy rights to the table. If you are going to append the new entries, you only need Create rights.

After you complete this operation, the table entries will be owned by the NIS+ principal that performed the operation and the group specified by the NIS_GROUP environment variable.

Prerequisites

- The domain must have already been set up and its master server must be running.
- The domain's servers must have enough swap space to accommodate the new table information. See *Name Services Configuration Guide*.
- The information in the file must be formatted appropriately for the table into which it will be loaded. The *Name Services Configuration Guide* describes the format a text file must have to be transferred into its corresponding NIS+ table. Local `/etc` files are usually formatted properly, but may have several comments that you would need to remove.

Information You Need

You need the name and location of the text files that will be transferred.

▼ How to Populate NIS+ Tables From Files

1. **Log on to an NIS+ client.**
   You can perform this task from any NIS+ client— just be sure that the client belongs to the same domain as the tables into which you want to transfer the information. The examples in this task use the root master server. Since

the administrator in these examples is logged on as superuser, the NIS+ principal actually performing this operation (and therefore needing the proper credentials and access rights) is the root master server.

2. **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 `csh` users, one for `sh/ksh` users:

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

rootmaster# PATH=$PATH:/usr/lib/nis            # sh/ksh users
rootmaster# export PATH
```

   If you are using the Solaris 1.x distribution of NIS+, the path may be different; check the README file.

3. **Use `nisaddent` to transfer any of these files, one at a time:**
   aliases, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `protocols`, `rpc`, `services`

   The `publickey, automounter`, `passwd`, and `shadow` files require slightly different procedures; for the `publickey` file, go to Step 4, for the `automounter` file, go to Step 5, for the passwd and shadow file go to Step 6.

   By default, `nisaddent` *appends* the file information to the table information. To replace or merge instead, use the `-r` or `-m` options.

```
rootmaster# nisaddent -r -f filename table [domain]   # to replace
rootmaster# nisaddent -a -f filename table [domain]   # to append
rootmaster# nisaddent -m -f filename table [domain]   # to merge
```

The best option for populating the tables for the first time is the `-a` option, which is the default anyway. 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. *Table* is the name of the NIS+ table. The *domain* argument is optional; use it only to populate tables in a different domain. Here are some examples, entered from the root domain's master server. The source files are simply edited versions of the `/etc` files:

```
rootmaster# nisaddent -m -f /etc/hosts.xfr hosts
rootmaster# nisaddent -m -f /etc/groups.xfr groups
```

If you perform this operation from a non-root server, keep in mind that a non-root server belongs to the domain above the one it supports. Therefore, it is a client of another domain. For example, the Sales.Wiz.Com. master server belongs to the Wiz.Com. domain. To populate tables in the Sales.Wiz.Com. domain from that master server, you would have to append the Sales.Wiz.Com. domain name to the `nisaddent` statement. For example:

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

If you performed this operation as a client of the Sales.Wiz.Com. domain, you would not need to append the domain name to the syntax. For more information about `nisaddent`, see Chapter 12, "Administering NIS+ Tables."

To verify that the entries were transferred into the NIS+ table, use the `niscat` command (described in Chapter 12, "Administering NIS+ Tables"). Here is an example:

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

4. **Transfer the publickey file.**
   Since the domain's Cred table already stores some credentials, you need to make sure they are not overwritten by the contents of the publickey text file that you transfer into the Cred table. You can avoid this by removing those credentials from the publickey text file. For rootmaster, that line would be:

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

   Then you can transfer the contents of the publickey file into the Cred table. Use `nisaddent`, but with the `-a` (add) option.

```
rootmaster# nisaddent -a -f /etc/publickey.xfr -t cred.org_dir
publickey [domain]
```

   Note, however, that this operation only transfers DES credentials into the Cred table. Principals will still need to create their LOCAL credentials to the Cred table.

5. **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
```

6. **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 as shown below. (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
```

7. **Checkpoint the tables.**
   This step ensures that all the servers supporting the domain transfer the new information from their .log files to the disk-based copies of the tables. If you have just setup 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 (upper case) option:

```
rootmaster# nisping -C org_dir
Checkpointing replicas serving directory org_dir.Wiz.Com. :
Master server is rootmaster.Wiz.Com.
     Last update occured at date

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

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

This step completes this task.

5

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. Log on to an NIS+ client.
2. Add `/usr/lib/nis` to the search path for this shell.
3. Use `nisaddent` to transfer any of these maps, one at a time: `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `netgroup`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`.
4. Transfer the `publickey` map.
5. Transfer the automounter information.
6. Checkpoint the tables.

Security Considerations

You can perform this task from any NIS+ client as long as you (or superuser on the client) have the proper credentials and access rights. If you are going to replace or merge the entries in the table with the entries from the NIS map, you must have Create and Destroy rights to the table. If you are going to append the new entries, you only need Create rights.

After you complete this operation, the table entries will be owned by the NIS+ principal that performed the operation (either you or, if logged on as superuser, the client) and the group specified by the NIS_GROUP environment variable.

Prerequisites

- The domain must have already been set up and its master server must be running.
- The `dbm` files (`.pag` and `.dir` files) for the NIS maps you are going to load into the NIS+ tables must already be in a subdirectory of `/var/yp`.

Information You Need

You need the name and location of the NIS maps.

▼ How to Populate NIS+ Tables From Maps

1. **Log on to an NIS+ client.**
   You can perform this task from any NIS+ client— just be sure that the client belongs to the same domain as the tables into which you want to transfer the information. The examples in this task use the root master server. Since the administrator in these examples is logged on as superuser, the NIS+ principal actually performing this operation (and therefore needing the proper credentials and access rights) is the root master server.

2. **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 `csh` users, one for `sh/ksh` users:

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

rootmaster# PATH=$PATH:/usr/lib/nis                # sh/ksh users
rootmaster# export PATH
```

   If you are using the Solaris 1.x distribution of NIS+, the path may be different; check the README file.

3. **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 4, and for the automounter files, go to Step 5.

By default, nisaddent *appends* the file information to the table information. To replace or merge instead, use the -r or -m options:

```
rootmaster# nisaddent -r -y nisdomain  table          # to replace
rootmaster# nisaddent -a -y nisdomain  table          # to append
rootmaster# nisaddent -m -y nisdomain  table          # to merge
```

The best option for populating the tables for the first time is the -a option, which is the default anyway. The best option to synchronize the NIS+ tables with NIS maps or /etc files is the -m (merge) option.

The -y (lower case) option indicates an NIS domain instead of a text file. The *nisdomain* argument is the name of the NIS domain whose map you are going transfer into the NIS+ table. You don't have to name the actual map; the nisaddent utility automatically selects the NIS map that correspond to the *table* argument. Here are some examples:

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

The first example transfers the contents of the hosts.byname and hosts.byaddr maps in the OldWiz (NIS) domain to the NIS+ Hosts table in the root domain (NIS+). The second transfers the NIS maps that store password-related information into the NIS+ Passwd table. The third does the same with group-related information. For more information about the nisaddent command, see Chapter 12, "Administering NIS+ Tables."

4. **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. First, dump the publickey map to a file:

```
rootmaster# makedbm -u /var/yp/OldWiz/publickey.byname  \
            /etc/publickey.xfr
rootmaster# vi /tmp/publickey
```

Open the file and remove the credentials of the workstation you are logged onto from the publickey map. For rootmaster, that line would be:

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

Now you can transfer the contents of the *file* — not the map — into the Cred table. Use `nisaddent`, but with the `-a` (add) option.

```
rootmaster# nisaddent -a -f /etc/publickey.xfr -t cred.org_dir
Publickey
```

Note, however, that this operation only transfers DES credentials into the Cred table. Principals will still need to create their LOCAL credentials to the Cred table.

5. **Transfer the automounter information.**
Although the `nissetup` utility creates auto_master and auto_home tables, they are not considered standard NIS+ tables. Therefore, transferring information into them requires a slightly different syntax:

```
rootmaster# nisaddent -y OldWiz -Y auto.master \
                      -t auto_master.org_dir key-value
rootmaster# nisaddent -y OldWiz -Y auto.home \
                      -t auto_home.org_dir key-value
```

The `-m` and `-y` options are still required, as is the NIS domain name (in this instance, OldWiz). However, you must precede the name of the NIS map (e.g., auto.master) with a `-Y` (upper case). 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+ table (auto_master.org_dir) and the type of table (key-value). Be sure to append the “org_dir” suffixes to the NIS+ table names.

6. **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` (upper case) option:

```
rootmaster# nisping -C org_dir
Checkpointing replicas serving directory org_dir.Wiz.Com. :
Master server is rootmaster.Wiz.Com.
     Last update occured at date

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

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

   This step completes this task.

Transferring Information From NIS+ to NIS

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

1. Log on to the NIS+ server.
2. Transfer the NIS+ tables into output files.
3. Transfer the contents of the output files into the NIS maps.

Security Considerations

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

Prerequisites

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

▼ How to Transfer Information From NIS+ to NIS

1. **Log on to the NIS+ server.**
   This example uses the server named "dualserver."

2. **Transfer the NIS+ tables into output files.**
   Use the `nisaddent` command with the `-d` option, as shown below, once for each table.

   ```
   dualserver% /usr/lib/nis/nisaddent -d [-t table] tabletype > filename
   ```

   The `-d` option dumps the contents of *table* to *filename*, converting the contents back to standard `/etc` file format.

3. **Transfer the contents of the output files into the NIS maps.**
   The NIS+ output files are ASCII files that you can use as input files for the NIS maps. Copy them into the NIS master's `/etc` directory, and then use `makedbm` as usual:

   ```
   dualserver# makedbm flags output-file   NIS-dbm-file
   ```

Limiting Access to the Passwd Column

This task describes how to limit Read access to the passwd column of the Passwd table only to authenticated users without affecting the Read access of unauthenticated principals (i.e., applications) to the remaining columns in the Passwd table. Use this task only for NIS+ releases prior to Solaris 2.3, since Solaris 2.3 and later supplies as a default the access rights provided by this task.

Security Considerations

- The domain must *not* be running in NIS-compatibility mode.
- All NIS+ clients of the domain must have DES credentials.

Prerequisites

- The Passwd table must have already been set up. It need not have any information in it, however.
- The NIS+ principal performing this task must have Modify rights to the Passwd table.

Information You Need

All you need is the name of the Passwd table.

▼ How to Limit Access to the Passwd Column

1. **Log onto 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
```

If the table has the permissions shown below, you don't need to perform this task; the table is already set up properly:

```
rootmaster# niscat -o passwd.org_dir
Object Name      : passwd
.
.
.
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---
```

3. **Change the table permissions.**
   Use the `nischmod` command to change the table's object-level permissions to:

```
--- rmcd rmcd r---
```

```
rootmaster# nischmod n=,og=rmcd,w=r  passwd.org_dir
```

4. **Change the column permissions.**
Use the `nistbladm` command with the `-u` option to change the column permissions to:

```
name     r--- ---- ---- r---
passwd    ---- -m-- ---- ----
uid      r--- ---- ---- r---
gid      r--- ---- ---- r---
gcos     r--- ---- ---- r---
home     r--- ---- ---- r---
shell    r--- ---- ---- r---
shadow   r--- ---- ---- r---
```

You could use `nistbladm` once . . .

```
rootmaster# nistbladm -u [name=n+r,uid=n+r,gid=n+r,gcos=n+r, \
            home=n+r,shell=n+r,shadow=n+r],passwd.org_dir
```

. . . but if you made a typing error, you'd have to retype the whole line, possibly several times over. You might prefer to use the command once for each column:

```
rootmaster# nistbladm -u [name=n+r],passwd.org_dir
rootmaster# nistbladm -u [uid=n+r],passwd.org_dir
rootmaster# nistbladm -u [gid=n+r],passwd.org_dir
rootmaster# nistbladm -u [gcos=n+r],passwd.org_dir
rootmaster# nistbladm -u [home=n+r],passwd.org_dir
rootmaster# nistbladm -u [shell=n+r],passwd.org_dir
rootmaster# nistbladm -u [shadow=n+r],passwd.org_dir
```

This example assumes the columns had no previous Read permissions assigned to the Nobody class. Notice that no Read permissions are added to the passwd column. If your table provided different permissions, you would use a different syntax; for details, see the description of the `nistbladm -u` command on page 151.

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.

Security Considerations

- The domain must *not* be running in NIS-compatibility mode.
- All clients of the domain must have DES credentials.
- All clients of the domain must be running Solaris 2.3 or a later release.
- Users' network passwords (used to encrypt their DES credentials) must be the same as their login passwords.

Prerequisites

- The Passwd table must have already been set up. It need not have any information in it, however.
- The NIS+ principal performing this task must have Modify rights to the Passwd table.
- The previous task, "Limiting Access to the Passwd Column" must have already been performed.

Information You Need

All you need is the name of the Passwd table.

▼ How to Limit Read Access to the Passwd Column

1. **Log onto 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 9, "Administering NIS+ Access Rights."

3. **Change the table permissions.**
   Use the `nischmod` command to change the table's object-level permissions to:

```
---- rmcd rmcd ----
```

```
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.

Summaries

Following is a summary of the steps required to populate NIS+ tables. 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 brevity, these summaries do not show the server's responses to each command.

rootmaster%	1. Log on to an NIS+ client.
% **PATH=$PATH:/usr/lib/nis; export PATH**	2. Add /usr/lib/nis to search path.
% nisaddent -m -f /etc/hosts.xfr hosts	3. Transfer each file, one at a time.
% .	
% .	
% .	
% **vi /etc/publickey.xfer**	4. Transfer publickey file.
	First remove new credentials from
	the file.
% **nisaddent -a -f /etc/publickey.xfr cred**	Then transfer it.
% **nisaddent -f auto.master.xfr \**	5. Transfer the automounter files.
-t auto_master.org_dir key-value	
% **nisaddent -f auto.home.xfr \**	
-t auto_home.org_dir key-value	
% nisping -C org_dir	6. Checkpoint the table directory.

Figure 5-1 Summary of Steps to Transfer Files Into NIS+ Tables

```
rootmaster%
% PATH=$PATH:/usr/lib/nis; export PATH
$ nisaddent -m -y OldWiz hosts
% .
% .
% .
% makedbm -u /var/yp/OldWiz/publickey.byname > \
          /etc/publickey.xfr
% vi /etc/publickey.xfr
% nisaddent -a -f /etc/publickey.xfr -t
cred.ortg_dir publickey
% nisaddent -y OldWiz -Y auto.master \
            -t auto_master.org_dir key-value
% nisaddent -y OldWiz -Y auto.home \
            -t auto_home.org_dir key-value
% nisping -C org_dir
```

```
1. Log on to an NIS+ client.
2. Add /usr/lib/nis to search path.
3. Transfer each map, one at a time.
4. Transfer publickey file. First dump
   the publickey map to a file.
   Then remove new credentials.
   Then transfer the file.
5. Transfer the automounter maps.
6. Checkpoint the table directory.
```

Figure 5-2 Summary of Steps to Transfer Maps into NIS+ Tables

```
dualserver%
% /usr/lib/nis/nisaddent -d [-t table] tabletype >
filename
% .
% .
% .
% makedbm flags output-file NIS-dbm-file
```

```
1. Log on to NIS+ server.
2. Transfer NIS+ tables to files.
3. Transfer files to NIS maps.
```

Figure 5-3 Summary of Steps to Transfer NIS+ Tables to NIS Maps

```
rootmaster#
# niscat -o passwd.org_dir
# nischmod n=,og=rmcd,w=r  passwd.org_dir
# nistbladm -u [name=n+r,uid=n+r,gid=n+r, \
                gcos=n+r,home=n+r,shell=n+r, \
                shadow=n+r], passwd.org_dir
# niscat -o passwd.org_dir
```

```
1. Log onto the domain's master server.
2. Check the table's existing rights.
3. Assign the table new rights.
4. Assign the columns new rights
5. Verify the new rights.
```

Figure 5-4 Summary of Steps to Limit Acces to Passwd Column

`rootmaster#`	1. Log onto the domain’s master server.
`#` **`niscat -o passwd.org_dir`**	2. Check the table’s existing rights.
`#` **`nischmod og=rmcd,nw=  passwd.org_dir`**	3. Assign the table new rights.
`#` **`nistbladm -u [passwd=o+r, shadow=n+r], \`** `passwd.org_dir`	4. Assign the columns new rights
`#` **`niscat -o passwd.org_dir`**	5. Verify the new rights.

Figure 5-5 Summary of Steps to Change Access Rights to Passwd Column

Setting Up the Name Service Switch 6

This section provides the following step-by-step instructions for using the Name Service Switch:

Selecting an Alternate Configuration File

This task describes how to select an alternate Switch configuration file for an NIS+ client. Make sure the sources listed in the file are properly set up. In other words, if you are going to select the NIS+ version, the client must eventually have access to NIS+ service; if you are going to select the local files version, those files must be properly set up on the client.

Here is a list of the steps:

1. Log on as superuser to the client.
2. Copy the alternate file over the `nsswitch.conf` file.
3. Kill and restart the key server at an appropriate time.
4. Reboot the workstation at an appropriate time.

Security Considerations

You must perform this operation as superuser.

▼ How to Select an Alternate Configuration File

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

```
client1# cd /etc
client1# cp nsswitch.nisplus nsswitch.conf    # NIS+ version
client1# cp nsswitch.nis nsswitch.conf        # NIS version
client1# cp nsswitch.files nsswitch.conf      # local files version
client1# cp nsswitch.custom  nsswitch.conf    # custom version
```

3. **Kill and restart the key server at an appropriate time.**
   The key server reads the publickey entry in the Name Service Switch configuration file only when the key server is started. As a result, if you change the switch configuration file, the key server does not become aware of changes to the publickey entry until it is restarted. If you are performing this task as part of the task called "Setting Up NIS+ Clients" on page 26", you don't need to kill and restart the keyserver now, since you'll do that as part of the overall client setup procedure.

   If you are performing this task independently, or as part of the task called "How to Set Up a Root Domain" on page 3", you can kill and restart the keyserver now. Here is an example:

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

4. **Reboot the workstation at an appropriate time.**
   Because some library routines do not periodically check the `nsswitch.conf` file to see whether it has been changed, you must reboot the workstation to make sure those routines have the latest information in the file. If selecting a different configuration file as part of a setup procedure, wait until an appropriate time to reboot.

   This step completes this task.

Enabling an NIS+ Client to Use DNS.

This task describes how to set up the Name Service Switch configuration file so that an NIS+ client can also use the DNS service. Here is a list of the steps:

1. Log on as superuser.
2. Open the `/etc/nsswitch.conf` file.
3. Specify DNS as a source of hosts information.
4. Save the file and reboot the workstation.

Prerequisites

The NIS+ client must have a properly configured `/etc/resolv.config` file (as described in *Name Services Configuration Guide.*")

Security Considerations

You must perform this operation as superuser.

▼ How to Enable an NIS+ Client to use DNS

1. **Log on as superuser.**
2. **Open the** `/etc/nsswitch.conf` **file.**

3. **Specify DNS as a source of hosts information.**
   DNS can be the only source or an additional source for the hosts information. Locate the hosts line and use "dns" in one of the ways shown below:

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

hosts:   files dns
```

   Do *not* use the above syntax for NIS clients, since it will make them look for unresolved names twice in DNS.

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

   This step completes this task.

Adding Compatibility with +/- Syntax

This task describes how to add compatibility with the +/- syntax used in `/etc/passwd`, `/etc/shadow`, and `/etc/group` files. Here is a list of the steps:

1. Log on as superuser.
2. Open the `/etc/nsswitch.conf` file.
3. Change the passwd and group sources to `compat`.
4. Save the file and reboot the workstation.

Security Considerations

You must perform this operation as superuser.

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

▼ How to Add Compatibility with +/- Syntax

1. **Log on as superuser.**
2. **Open the** `/etc/nsswitch.conf` **file.**
3. **Change the passwd and groups sources to** `compat`**.**

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

This provides the same semantics as in SunOS 4.1: it looks up an `/etc` files and NIS maps as indicated by the `+/-` entries in the files.

If you would like to use the `+/-` semantics with NIS+ instead of NIS, add the following two entries to the `nsswitch.conf` file:

```
passwd_compat:  nisplus
group_compat:   nisplus
```

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

This step completes this task.

Part 2 — Administering NIS+

This part of the manual describes how to administer the NIS+ service. It has seven chapters.

Administering NIS+ Security 7

This chapter describes how NIS+ protects its namespace. The first section provides a simplified overview of the security process. More detailed descriptions are provided in the remaining two sections.

Overview of the Security Process

You can, of course, run NIS+ without any security. Simply run NIS+ servers at security level 0, as described "About Server Security Levels" on page 94.

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 is provided by two means: *authentication* and *authorization*. Authentication is the process by which an NIS+ server identifies the NIS+ *principal* who sent a particular request. Authorization is the process by which a server identifies the access rights granted to that principal.

About NIS+ Principals

An NIS+ principal is the entity that submits a request for NIS+ service from an NIS+ client (It can also be the entity that supplies the NIS+ service from an NIS+ server. Keep in mind that all NIS+ servers are also NIS+ clients, so much of this discussion, though focused on clients, also applies to servers.) That entity may be someone who is logged into the client as a regular user, or someone who is logged on as superuser. In the first instance, the request actually comes from the client user; in the second instance, the request comes from the client workstation. Therefore, an NIS+ principal can be a client user or a client workstation.

NIS+ principals are identified by their *credentials*. When a user is logged into an NIS+ client as a regular user requests for NIS+ service include *user* credentials:

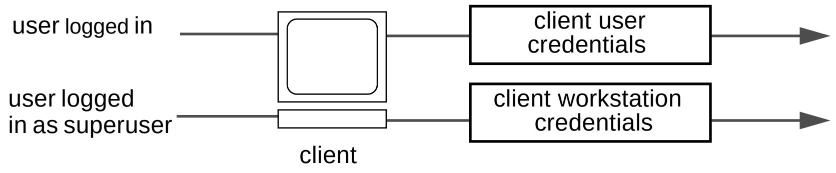


Figure 7-1 Schematic: Requests for Credentials

When a user is logged into an NIS+ client as superuser, request for services carries with it the *client workstation's* credentials. This distinction is important, since a client request could be denied because it is coming from the right NIS+ client but the wrong NIS+ principal.

NIS+ uses two types of credential: LOCAL and DES. (See"NIS+ Authentication" on page 95 for a description of these two types of credential.) These credentials are used to *authenticate* the NIS+ principal so that the server can determine whether the principal has been *authorized* to perform the requested operation.

About NIS+ Access Rights

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 an NIS+ object.

NIS+ operations vary among different types of objects, but they fall into four classes: Read, Modify, Create, and Destroy. 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 workstation, 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.

NIS+ objects specify their access rights as part of their object definitions. (You can examine these by using the `niscat -o` command, described on page 168.) So if the operation that a principal tries to perform on an object is *authorized* by the object's definition, the server performs it. If not, the request is denied.

In this discussion ***group*** refers to NIS+ groups, not UNIX or net groups. (See "The Object's Group" on page 110 for a description of NIS+ groups.)

An object does not grant access rights directly to a particular principal. Instead, it grants access rights to four *classes of principal*: Owner, Group, World, and Nobody. The principal who happens to be the object's owner gets the rights granted to the Owner class. The principals who belong to the object's Group class get the rights granted to the Group class. The World class encompasses all NIS+ principals that a server has been able to authenticate. Finally, the Nobody class is reserved for everyone, whether an authenticated principal or not. (Commands for working with access rights are described in Chapter 9, "Administering NIS+ Access Rights.") Figure 7-2 summarizes the entire process.

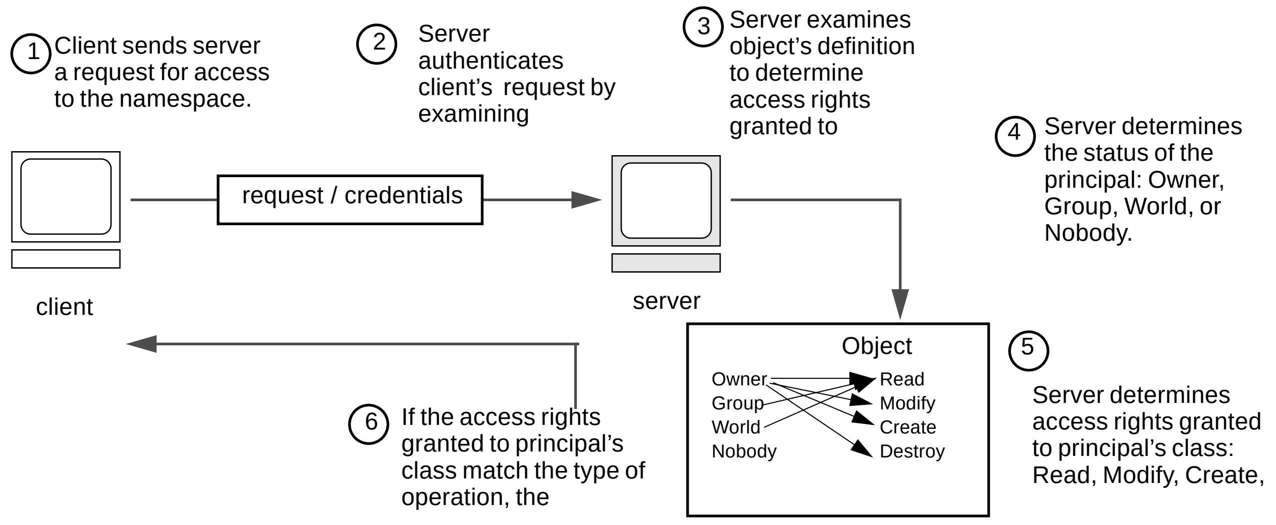

1 Client sends server a request for access to the namespace.
2 Server authenticates client's request by examining
3 Server examines object's definition to determine access rights granted to
4 Server determines the status of the principal: Owner, Group, World, or Nobody.
request / credentials
client
server
Object
Owner
Group
World
Nobody
Read
Modify
Create
Destroy
5 Server determines access rights granted to principal's class: Read, Modify, Create,
6 If the access rights granted to principal's class match the type of operation, the

Figure 7-2 Summary of NIS+ Security Process

About Server Security Levels

NIS+ servers can operate at three security levels. These levels determine the types of credential a principal must submit for its request to be authenticated. They are described in Table 7-1 below:

Table 7-1 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. Requests that use invalid `DES` credentials are denied. No NIS+ credentials are created or maintained. Note that `nis_passwd` cannot function without credentials. Unless there are credentials existing from some previous higher level, `nis_passwd` cannot function in a level 0 environment. At level 0, other NIS+ commands that check or work with permissions, default down to using the UNIX file permission security system.
1	Security level 1 is designed only for testing and debugging; in particular, to debug without the complications of DES authentication. Do not use it on networks to which untrusted clients may have access. Security level 1 authenticates requests that use either `LOCAL` or `DES` credentials. Requests that don't use any credentials at all are assigned only the access rights granted to the Nobody class.
2	Security level 2, the default, is the highest level of security currently provided by NIS+. It only authenticates requests that use `DES` credentials. Requests that use `LOCAL` credentials or none at all are assigned the access rights granted to the Nobody class. Requests that use invalid `DES` credentials are denied.

A Note About Setting Up NIS+ Security

Security cannot be added to an existing 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, in Part II of this manual.

NIS+ Authentication

Authentication is the mechanism used by an NIS+ server to verify the credentials of the NIS+ principal that has sent a request for NIS+ service. To explain the authentication mechanism, this section discusses:

- Types of credentials
- Where credentials are stored
- How credential information is created by the administrator
- How credentials are created by the client
- How credentials are examined by the server
- The cached public keys

Types of Credential

NIS+ principals can have two types of credential: LOCAL and DES. A client user can have both types, but a client workstation can only have a DES credential:

Table 7-2 Types of Credentials

Type of Credential	Client User	Client Workstation
LOCAL	Yes	No
DES	Yes	No

DES credential information can only be stored in the Cred table of the principal's home domain, regardless of whether that principal is a client user or a client workstation. LOCAL credentials, however can be stored in any domain. In fact, in order to log into a remote domain and be authenticated, a client user *must* have LOCAL credentials in the Cred table of the remote domain. If the user does not have LOCAL credentials in the remote domain, they would have the access rights of the user *nobody*.

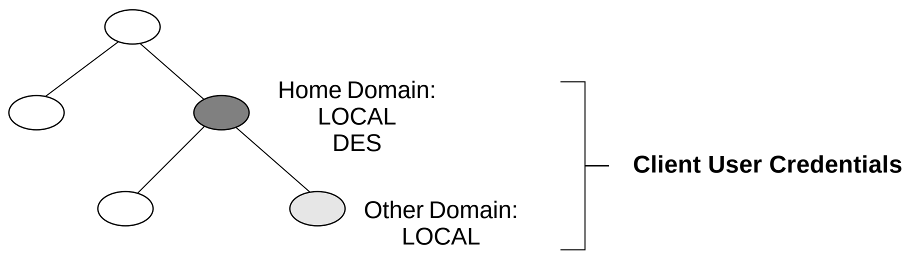


Figure 7-3 Credentials and Domains

The LOCAL credentials are used to map the user ID of an NIS+ principal to its NIS+ principal name. Since the UID of the root user on every workstation is zero, a LOCAL credential doesn't make sense for a client *workstation*; thus, it is allowed only for a client *user*.

A DES credential is more complex than a LOCAL credential, not only because of the information it requires, but because of the process involved in creating and verifying it. To understand how DES authentication works, you need to distinguish between the credential itself and the information that is used to create and verify it. This book uses the term *credential* for the former and uses the term *credential information* for the latter. Thus, the credential is the bundle of numbers that is sent by the client to the server, and the credential information is the data that is stored in the Cred table, and used by the client to generate the credential and by the server to verify the credential.

The DES credential itself consists of a principal’s *secure RPC netname* plus a *verification* field:

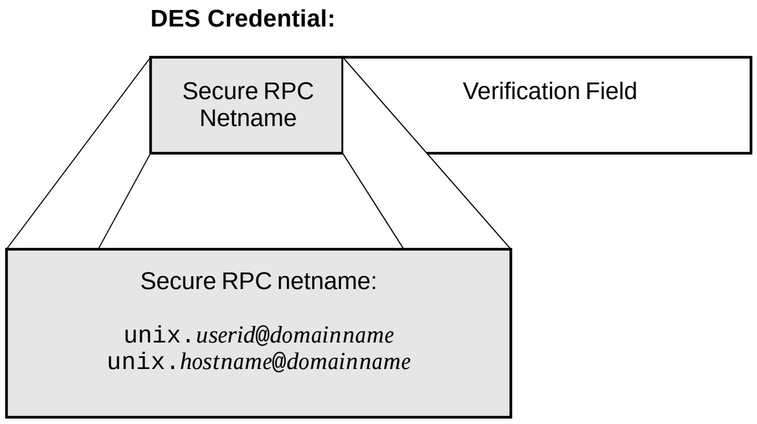


Figure 7-4 The DES Credential

The secure RPC netname portion of the credential is the part used to actually identify the NIS+ principal. Every secure RPC netname begins with the prefix `unix`. If the principal is a client user, the second field in the netname is the user’s UID. If the principal is a client workstation, the second field in the netname is the workstation’s hostname. The last field is the principal’s home domain:

Table 7-3 Secure RPC Netname Format

If the Principal Is:	**The Domain Must Be:**
A client user	The domain that contains the user’s password entry and DES credentials.
A client workstation	The domain name returned by executing the `domainname` command on that workstation.

Remember that an NIS+ principal name *always* has a trailing dot, while a secure RPC netname *never* does.

The verification field of the credential is used to make sure the credential is not forged. It is generated from the *credential information* stored in the Cred table.

Where Credentials Are Stored

NIS+ credentials are stored in a *Cred* table. The Cred table is one of the 16 default NIS+ tables. Each domain has one Cred table, which stores the credentials of client workstations that belong to that domain and client users who are allowed to log onto them. The Cred tables are located in their domains' `org_dir` subdirectory:

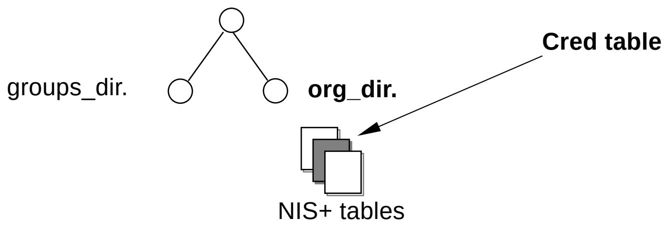


Figure 7-5 Credential Location

You can view the contents of a Cred table with the `niscat` command, described in Chapter 12, "Administering NIS+ Tables."

The Cred table as shown in Table 7-4 has five columns:

Table 7-4 Cred Table

NIS+ Principal Name	Authentication Type	Authentication Name	Public Data	Private Data
NIS+ principal name of a client user	LOCAL	UID	GID list	none
NIS+ principal name of a client user or client workstation	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. If the authentication type is LOCAL, the other columns contain a client user's NIS+ principal name, UID, and GID; the last column is empty. If the authentication type is DES, the other columns contain an NIS+ 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, as described later in this section.

Creating Credential Information

Credentials for NIS+ principals can be created any time after their domain has been set up; in other words, once a Cred table exists. When a namespace is first set up, credentials are created first for the administrators who will support the domain. Once they have credentials, they can create credentials for other administrators, client workstations, and client users. Step-by-step instructions for creating NIS+ credentials are distributed throughout this manual (see Table 7-5 below).

Table 7-5 Summary of Steps for Creating NIS+ Credentials

Chapter	Task/Step		Page
Chapter 1	▼ “How to Set Up a Root Domain		page 3
	Step 10.	Create DES credentials for the root master server.	page 14
	Step 16.	Add your LOCAL credentials to the root domain.	page 19
	Step 17.	Add your DES credentials to the root domain.	page 19
	Step 18.	Add credentials for other administrators.	page 20
Chapter 2	▼ Setting Up NIS+ Clients		page 26
	Step 2.	Create DES credentials for the new client workstation.	page 28
Chapter 4	▼ “Setting Up a Non-Root Domain		page 51
	Step 8.	Add credentials for other administrators.	page 103
Chapter 8	▼ How to Create Credentials for an NIS+ Principal		page 127

You could add credentials to your namespace simply by following the instructions listed above. However, understanding the process by which credentials are used may help you uncover mistakes in the setup process or solve problems that may arise later.

The command used to create credentials is `nisaddcred`. (See “The nisaddcred Command” on page 125.) The `nisaddcred` command creates either LOCAL credentials or DES credentials.

Note – You can also use the `nispopulate` and `nisclient` scripts to create credentials. They, in turn, call the `nisaddcred` command

---

When used to create LOCAL credentials, `nisaddcred` simply extracts the client user’s UID (and GID) from the client’s login record and places it in the domain’s Cred table.

When used to create DES credential information, `nisaddcred` goes through a two-part process: forming the principal’s secure RPC netname and generating the principal’s private and public keys. A secure RPC netname is formed by the principal’s user ID from the password record and the domain name (e.g. `unix.1050@sun.com`).

To encrypt the private key, `nisaddcred` needs the principal’s network password. So, when the `nisaddcred` command is invoked with the `des` argument, it prompts the principal for a network password. Normally, this password is the same as the principal’s login password (if it is different, additional steps are required as described in “Network Password Different from Login Password” on page 103). 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, 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 network password:

nisaddcred:

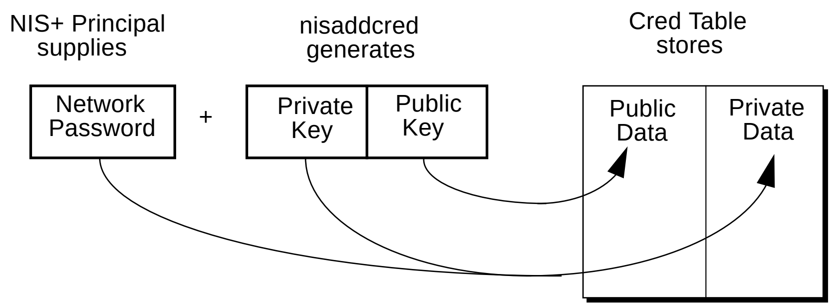


Figure 7-6 Keys and `nisaddcred`

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.

How Credentials Are Created by the Client

When an NIS+ client sends a request to an NIS+ server it first sends along its DES credential if it is available. If the DES credential fails because either the client or the server does not have the proper keys, the client tries to send its LOCAL credential. If the LOCAL credential fails, the client attempts again, without credentials as Nobody.

To generate its DES credential, the client depends on the `keylogin` command, which must have been executed before the client tries to generate its credential. The `keylogin` command (often referred to simply as a "keylogin") is executed automatically by the client when an NIS+ principal logs on. (Note that keylogin is not performed automatically if the client's login password is different from the client's network password.) The purpose of the keylogin is to give the client access to its private key. The keylogin fetches the principal's private key from the Cred table, decrypts it with the principal's *network* password (remember that the private key was originally encrypted with the principal's *network* password), and stores it locally for future NIS+ requests:

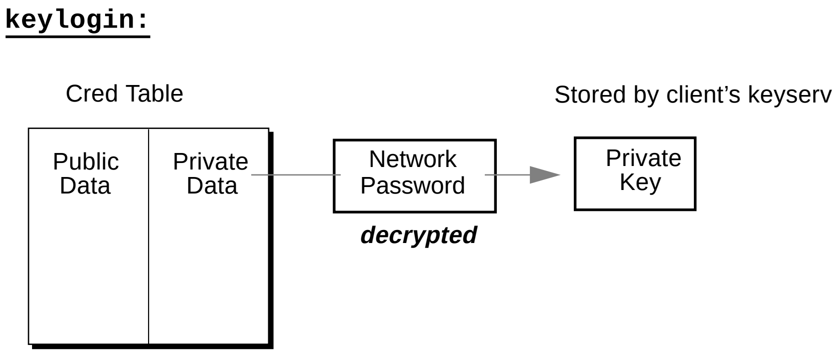


Figure 7-7 Private Key and `keylogin`

To generate its DES credential, the client still needs the public key of the server to which it will send the request. This information is stored in the client's directory cache. Once the client has this information, it can form the verification field of the credential.

First, the client generates a random DES key (Kcs) for encrypting various credential information. Then the client uses its own private key and the server's public key to generate a common DES key that is used to encrypt the random DES key generated earlier. It then generates a time stamp that is encrypted with the DES key (Kcs) and combines it with other credential-related information into the verification field:

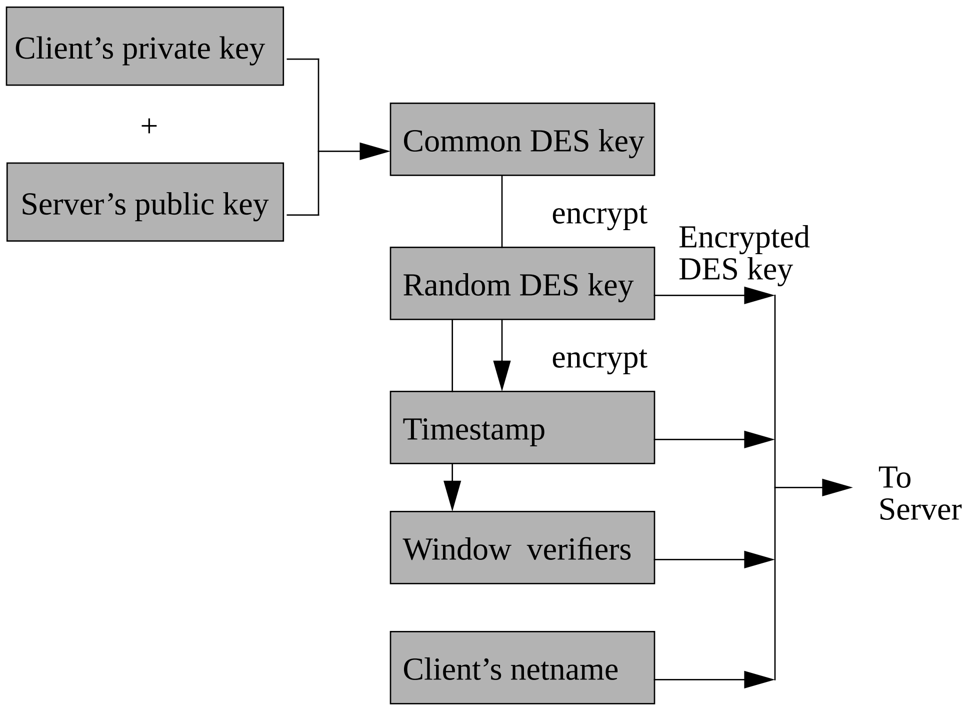


Figure 7-8 DES Keys and `keylogin`

To understand how a server examines these credentials, see “How Credentials are Examined by the Server” on page 106.

Network Password Different from Login Password

A principal’s network password is usually the same as its login password, as mentioned earlier. They can, however, be different. If so, the private key cannot be decrypted by the client. When the principal logs on to the client, the client attempts an automatic keylogin, as usual. In other words, the client fetches the principal’s private key from the Cred table, decrypting it with the principal’s login password. However, when the credential information was first stored in the Cred table, the private key was *encrypted* with the principal’s *network* password. As a result, the private key cannot be decrypted by the client and cannot be used for authentication:

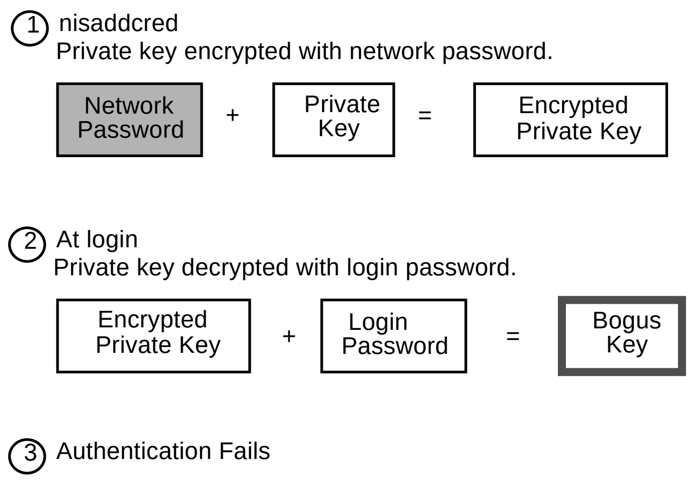


Figure 7-9 When Network and Login Passwords are Different

To solve this problem, the NIS+ principal must give the NIS+ client a network password after the principal logs on. This requires an explicit *keylogin*. In general, a principal must keylogin any time its credentials changes. In this particular case, it should be after providing the `nisaddcred` command a network password that is different from the login password:

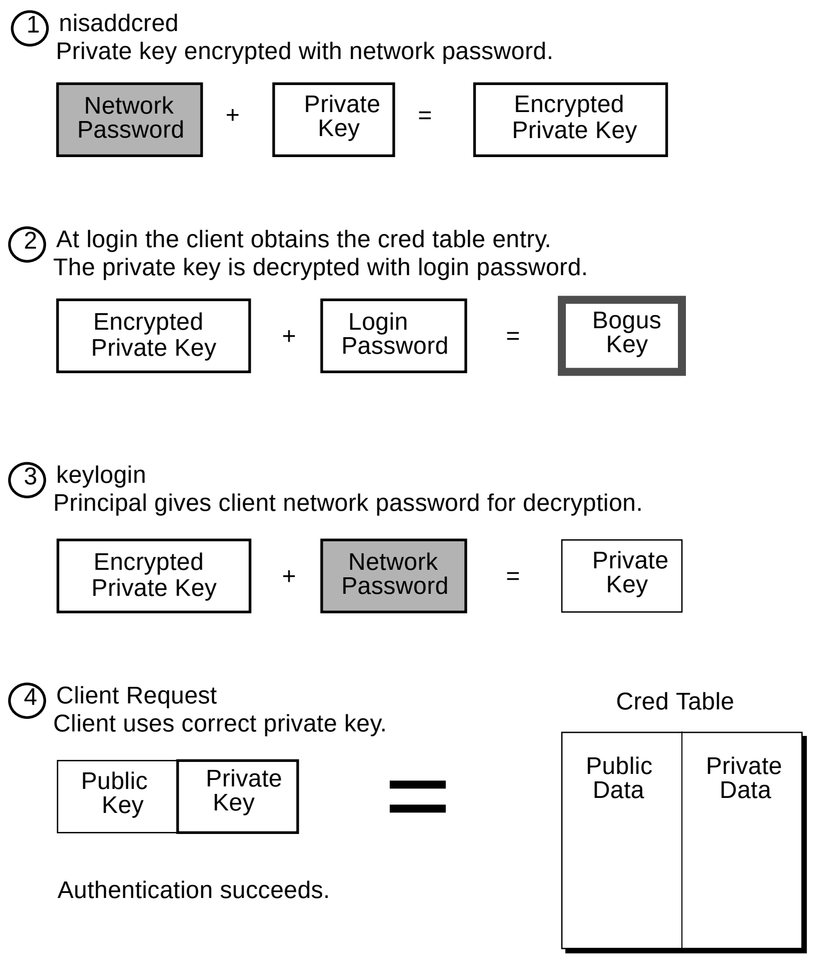


Figure 7-10 Solving Different Network and Login Password Problem

See "How to Keylogin" on page 139 for instructions.

How Credentials are Examined by the Server

To decrypt the DES credential, the server essentially reverses the encryption process performed by the client. First, the server uses the secure RPC netname portion of the credential to look up the principal's public key in the Cred table. Then, using its own private key (keep in mind that servers, because they are also clients, have credentials too) and the principal's public key, the server decrypts the DES key. Then it uses the DES key to decrypt the timestamp. If the timestamp is within a predetermined tolerance of the time the server received the message, the server authenticates the request.

This process satisfies the server. However, to let the client know that the information it receives indeed comes from a trusted server, the server encypts the timestamp with the DES key and sends it back to the client.

LOCAL credentials are not verified. Instead, the NIS+ server gets the NIS+ principal name of the principal who sent the request by looking up the principal's UID the third column of the Cred table.

Cached Public Keys

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. 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 Chapter 8 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'll need to understand how a server's public key, once created, is propagated through namespace objects. The process generally has five stages of propagation:

- Stage 1—Server's public keys are generated
- Stage 2—Public keys are propagated to directory objects
- Stage 3—Directory objects are propagated into client files
- Stage 4—When a replica is addes to the domain
- Stage 5—When the server's public keys are changed

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 `Wiz.Com.` root domain, and is made the master server of the `Sales.Wiz.Com.` domain:

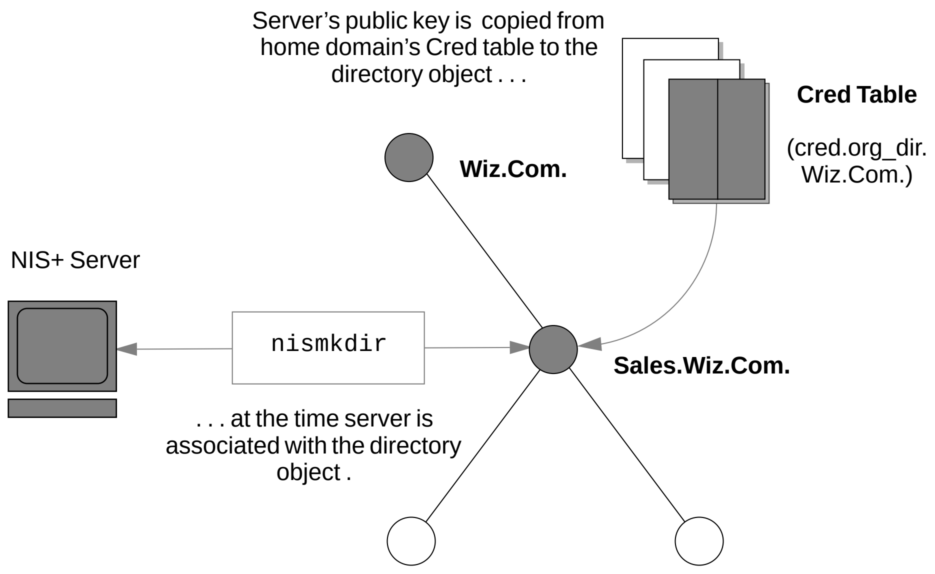

Server's public key is copied from home domain's Cred table to the directory object . . .
Cred Table
(cred.org_dir. Wiz.Com.)
Wiz.Com.
NIS+ Server
nismkdir
Sales.Wiz.Com.
. . . at the time server is associated with the directory object .

Figure 7-11 Public Key is Propagated to Directory Objects

Its public key is copied from the `cred.org_dir.Wiz.Com.` domain and placed in the `Sales.Wiz.Com.` directory object. This can be verified with the `niscat -o Sales.wiz.com.` command.

Stage 3 — Directory Objects Are Propagated into Client Files

All NIS+ clients are initialized with the `nisinit` utility or as described in Chapter 2, "Setting Up NIS+ Clients." (Clients can also be initialized with the `nisclient` script.) Among other things, this utility creates a coldstart file `/var/nis`/ NIS_COLDSTART. The coldstart file is used to initialize the client's directory cache `/var/nis/NIS_SHARED_DIRCACHE`. The coldstart 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 coldstart 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 180.

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 on page 181) 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)

As indicated above, 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 following commands:

Table 7-6 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	page 181
Directory object of domain supported by server	nisupdkeys	page 136
NIS_COLDSTART file of clients	nisinit -c	page 178
NIS_SHARED_CACHE file of clients	nis_cachemg	page 179

Note – You must first kill the existing nis_cachemgr before restarting nis_cachemgr.

NIS+ Authorization in Depth

Access rights specify the type of operation that NIS+ principals, both authenticated and unauthenticated, can perform on an NIS+ object. NIS+ offers four types of access rights as listed in Table 7-7.

Table 7-7 NIS+ Access Rights

Access Right	Description
Read	Principal can read the contents of the object.
Modify	Principal can modify the contents of the object.
Destroy	Principal can destroy objects in a table or directory.
Create	Principal can create new objects in a table or directory.

These rights are granted by each object to four different classes of NIS+ principals, not to a particular NIS+ principal. These are called *authorization classes*.

Authorization Classes

To assign access rights, every NIS+ object uses four authorization classes: Owner, Group, World, and Nobody. An object can grant any combination of access rights to each of these classes. For instance, an object could grant Read access to the World class, but Modify access only to the Group and Owner. Thus, any NIS+ principal that belonged to the World class could read the object, but only the NIS+ principals that belong to the Group and Owner class could modify the object. Each class is described below.

The Object's Owner

The Owner is a single NIS+ principal. 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 it creates the object (see "Overriding Defaults" on page 147). Another way is for the principal to change the ownership of the object after the object is created (see "The nischown Command" on page 154).

Once a principal gives up ownership, it 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 Object's Group

The object's group is a single NIS+ group. 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. By default, when an object is created, it is assigned the NIS+ principal's default group. (An object's Owner, however, does not need to belong to the object's Group.)

Information about NIS+ groups is *not* stored in the NIS+ Group table. That table stores information about UNIX groups. Information about NIS+ groups is stored in NIS+ group *objects*, under the `groups_dir` subdirectory of every NIS+ domain:

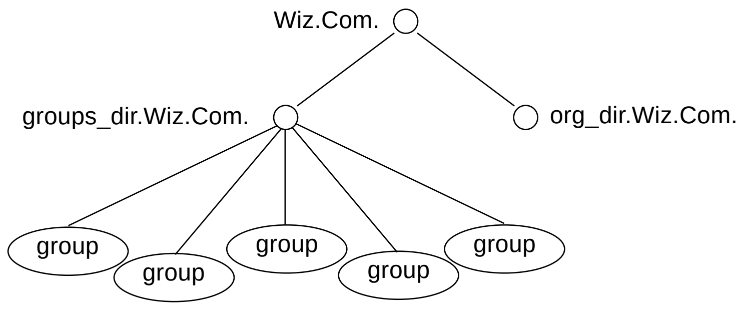


Figure 7-12 NIS+ Directory Structure

Instructions for administering NIS+ groups are provided in Chapter 10, “Administering NIS+ Groups”.

The World

The World is the class of all NIS+ principals that are authenticated by NIS+. Access rights granted to the World apply to all authenticated principals.

Nobody

Nobody is a class reserved for unauthenticated requests. Note that requests that are not authenticated because they sent along invalid credentials are simply denied, and not given any access rights, not even those of the Nobody class.

Where Access Rights are Stored

An object’s access rights are specified in its definition. (Note that this information is part of the object’s *definition*; it is *not* an NIS+ table.) The access rights can be viewed by using `niscat -o` *object_name*.

Table 7-8 Access Rights

Object's Owner	Object's Group Owner	Access Rights: Owner	Access Rights: Group	Access Rights: World	Access Rights: Nobody
The NIS+ principal that created the object or that was assigned ownership by the `nischown` command.	The NIS+ group owner of the object.	The access rights granted to the object owner	The access rights granted to the principals in the object's group.	The access rights granted to any authenticated NIS+ principal.	The access rights granted to everyone, whether authenticated or not.

Access rights are displayed as a list of 16 characters, like this:

```
r---rmcdr---r---
```

Each character represents a type of access right. Thus, `r` represents Read, `m` represents Modify, `d` represents Destroy, `c` represents Create, and `-` 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:

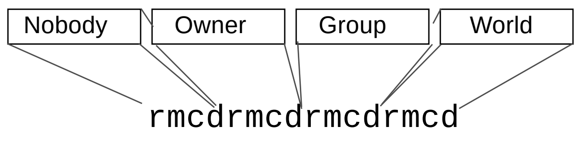


Figure 7-13 Access Rights Display

Note – Unlike UNIX file systems, the first set of rights is for Nobody, not for the Owner.

How Access Rights Are Assigned

When you create an object, NIS+ assigns the object a default owner, group, and set of access rights. The default owner is the NIS+ principal who creates the object. The default group is the group named in the `NIS_GROUP` environment variable. The default set of access rights is:

Table 7-9 Default Access Rights

Nobody	Owner	Group	World
-	Read	Read	Read
-	Modify	-	-
-	Create	-	-
-	Destroy	-	-

NIS+ provides several different ways to change these default rights. The first is the NIS_DEFAULTS environment variable. That variable stores a set of security related default values, one of which is access rights. The defaults stored in the NIS_DEFAULTS variable are assigned to every object that is created while they are in effect. If the value of this variable is changed on a particular client, any object created from that client will be assigned the new values. However, previously created objects will not be affected. Instructions for changing the NIS_DEFAULTS variable are provided in the section titled "Changing Defaults" on page 146.

A second way to affect access rights is with the `-D` option, which is available with several NIS+ commands. The `-D` option specifies the default values that will be applied to all the objects acted upon by the command. In other words, it overrides the default values stored by the NIS_DEFAULTS variable, but only for the object affected by that particular instance of the command. For instructions, see "Overriding Defaults" on page 147.

The third way is to explicitly change an object's access rights (or other security defaults) using one of the NIS+ commands designed for that purpose, such as `nischmod`. Instructions are provided in Chapter 9, "Administering NIS+ Access Rights."

NIS+ tables provide additional levels of security not provided by other types of NIS+ objects. Information in an NIS+ table can be accessed by column or entry:

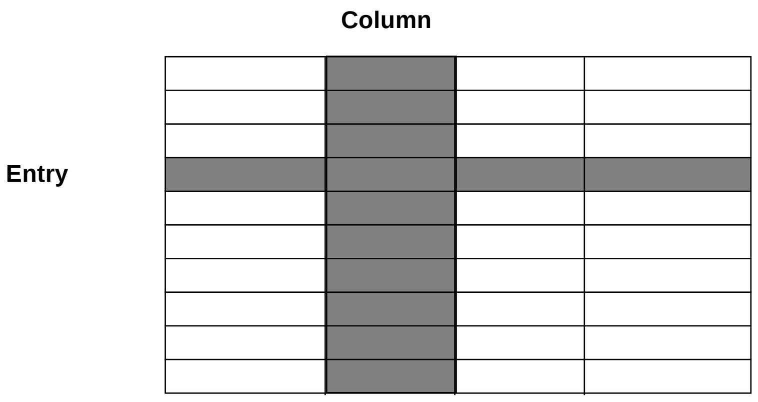


Figure 7-14 Table Access Columns and Rows

Besides the rights that can be assigned to the table as a whole, NIS+ allows you to assign access rights to the columns and entries of a table. Those rights can provide *additional* access, but they cannot restrict the access provided by the table as a whole. For instance, if the table provided Read rights to the World class, a column could give the World class Modify rights, but it could not restrict Read access to only the Owner or the Group.

Note – Any Read rights assigned to the Owner or Group in the Read column would be overridden by the World Read rights granted by the table object.

A column or entry 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 a couple of examples.

Assume a table object granted Read rights to the table's Owner:

```
                           Nobody   Owner    Group    World
  Table Access Rights   :  ----     r---     ----     ----
```

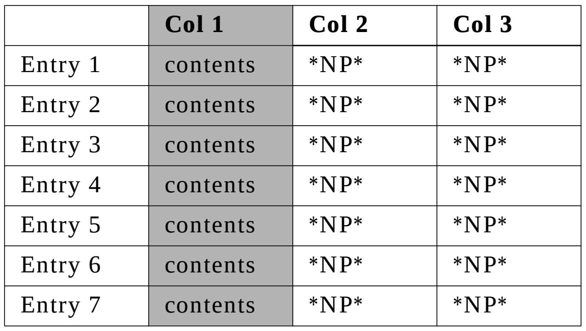
This would mean that only the table's owner could read the contents of the entire table. A principal who is not the owner would have no access. However, a particular entry in the table could also grant Read rights to the Group class:

```
                                  Nobody   Owner    Group    World
   Table Access Rights   :   ----     r---     ----     ----
   Entry1 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:

```
                                    Nobody   Owner    Group    World
   Table Access Rights    :   ----     r---     ----     ----
   Entry1 Access Rights   :   ----     ----     r---     ----
   Column1 Access Rights :   ----     ----     ----     r---
```

Members of the World class could now read that column *for all entries* in the table. Here is a representation of what would be displayed to members of the World class who tried to read the table:



	Col 1	**Col 2**	**Col 3**
Entry 1	contents	*NP*	*NP*
Entry 2	contents	*NP*	*NP*
Entry 3	contents	*NP*	*NP*
Entry 4	contents	*NP*	*NP*
Entry 5	contents	*NP*	*NP*
Entry 6	contents	*NP*	*NP*
Entry 7	contents	*NP*	*NP*

Figure 7-15 Column Access Example

Here is another example. Assume a table assigned Read rights to the Group:

```
                                    Nobody   Owner    Group    World
   Table Access Rights    :   ----     ----     r---     ----
```

Any member of the group could read the contents of the entire table, but could not create, modify, or destroy them. However, a particular entry could assign the group Modify rights:

```
                                    Nobody   Owner    Group    World
   Table Access Rights    :   ----     ----     r---     ----
   Entry1 Access Rights   :   ----     ----     -m--     ----
```

Members of the *entry*'s group could now modify the contents of the *entry*. However, they could still not destroy the entry. However, if a particular column assigned Create and Destroy rights to the Group . . .

```
                              Nobody  Owner   Group    World
Table Access Rights    :      ----    ----    r---     ----
Entry1 Access Rights   :      ----    ----    -m--     ----
Col1 Access Rights     :      ----    ----    --cd     ----
```

. . . every member of an entry's group could create or destroy information about the entry as long as it was contained in that column.

How a Server Grants Access Rights to Tables

The process for granting access rights at the object level has been already described (see "Overview of the Security Process" on page 91). However, because of the overlapping rights of table objects, columns, and entries, 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 that at security level 0, a server enforces no access rights and all clients are granted full access rights to the table object.

As described earlier, after authenticating a request, an NIS+ server determines the type of operation and the object of the request. If that object is a directory or group, the server simply examines the object's definition and grants the request, provided the object has assigned the proper rights to the principal. However, if the object is a table, the server follows a more involved process. The process varies somewhat depending on the operation, but it follows this general rule of thumb:

First, check rights at the table level

Then at the entry level

Then at the column level.

Following are detailed examples of the process involved in each type of operation. While going through them, keep in mind the four factors that a server must consider when deciding whether to grant access

1. The type of operation requested by the principal
2. The table, entry, or column the principal is trying to access

3. The authorization class the principal belongs to for that particular object
4. The access rights that the table, entry, or column has assigned to the principal's authorization class

Granting Access to Read or Modify a Table

When a principal requests to read or modify the contents of a table (for example, by issuing the `niscat` or `nistbladm` command), an NIS+ server employs the following logic to grant access to the principal. For this example, assume that the principal is a member of the *group* named in the table's group authorization class, and is trying to *read* or *modify* the entire table.

The server first checks the table object's rights. If the table grants Read access to the group, the principal is allowed to read all the contents of the table, and the server does not proceed to check rights of columns or entries. (The shaded boxes represent the columns and entries that the principal is allowed to read.)

Rights to Entire Table Granted to Group: READ				
	Col 1	**Col 2**	**Col 3**	**Col 4**
Entry 1	▨	▨	▨	▨
Entry 2	▨	▨	▨	▨
Entry 3	▨	▨	▨	▨
Entry 4	▨	▨	▨	▨
Entry 5	▨	▨	▨	▨
Entry 6	▨	▨	▨	▨
Entry 7	▨	▨	▨	▨

Figure 7-16 Entire Table Access Rights Example

If the table object does not grant Read access to the group, the server checks the access rights granted by the table entries the principal is trying to access.

If any of those entries grants its group Read rights, and if the principal belongs to that group (which may be different from the table's group), the server allows the principal to read that entry's contents. Then it checks the rights granted by the next entry. (The shaded rows below represent the entries that have given the group Read rights.)

	Rights	**Col 1**	**Col 2**	**Col 3**	**Col 4**
Entry 1	READ				
Entry 2					
Entry 3	READ				
Entry 4					
Entry 5	READ				
Entry 6	READ				
Entry 7					

Figure 7-17 Selected Row Access Rights Example

The server proceeds to check column rights only if none of the entries the principal has tried to access has granted the group Read rights.

If a server finds that no columns in the table have granted their group Read rights, it returns an error message, stating that the principal does not have permission to access the object. However, if any column grants its group Read rights, and if the principal belongs to that group (which may also be different from the table or even the entries' group), the server displays *the entries that specify the same group,* but censors the columns that do not grant their group Read rights. Censored columns display the string *NP* (for "no permission"). If all entries specified the same group, this would be displayed:

	Col 1	**Col 2**	**Col 3**	**Col 4**
Entry 1	READ	*NP*	READ	*NP*
Entry 2	READ	*NP*	READ	*NP*
Entry 3	READ	*NP*	READ	*NP*
Entry 4	READ	*NP*	READ	*NP*
Entry 5	READ	*NP*	READ	*NP*
Entry 6	READ	*NP*	READ	*NP*
Entry 7	READ	*NP*	READ	*NP*

Figure 7-18 Selected Columns Access Rights Example

However, if only some entries (e.g., 1, 3, 4, and 7) specified the same group, this would be displayed:

	Col 1	**Col 2**	**Col 3**	**Col 4**
Entry 1	READ	*NP*	READ	*NP*
Entry 2		*NP*		*NP*
Entry 3	READ	*NP*	READ	*NP*
Entry 4	READ	*NP*	READ	*NP*
Entry 5		*NP*		*NP*
Entry 6		*NP*		*NP*
Entry 7	READ	*NP*	READ	*NP*

Figure 7-19 Selected Row and Column Access Rights Example

Granting Access to Destroy Table Entries

When a principal requests to delete a table, the server checks the access rights granted to the principal by the directory that contains the table. However, when a principal requests to delete a table entry, the server checks its various

Destroy rights. Assume that the principal is a member of the group named in the table's group authorization class, and that the principal is trying to delete entries 1 and 5.

If the table object grants its group Destroy rights, the principal is allowed to remove any entry from the table (columns cannot be removed). The server checks no further. If the table object does not grant the group Destroy rights, the server checks access rights at the entry level.

At the entry level, the server only checks the rights of the entries that the principal is trying to destroy (that is, 1 and 5). If one of those entries grants its group Destroy rights, and if the principal belongs to that group, the principal is allowed to delete that entry. If one of those entries does not grant the group Destroy rights, or if the principal does not belong to the entry's group authorization class, the principal is not allowed to delete the entry.

If no entries grant their group Destroy rights, an error message is returned, stating that the principal does not have permission to access the object.

Since no columns can be deleted from a table, the server does not check column access rights during a Destroy operation.

Granting Access to Create Table Entries

When a principal tries to create a table, the server checks the access rights granted to the principal by the directory under which the table will be created. However, when a principal tries to add new table entries to an existing table, the server checks various Create and Modify rights. For this example, assume that the principal is a member of the group named in the table's group authorization class, and that the principal is trying to add entries 8 and 9.

If the table object grants its group Create rights, the principal is allowed to add entries to the table (columns cannot be added). The server checks no further. However, if the object does not grant its group Create rights, the server checks whether the entry that the principal is trying to create already exists.

If the entry exists, the server checks whether the table object has granted its group Modify rights. If it has, the server replaces the existing entry with the new one and checks no further. If the table has *not* granted its group Modify rights, the server checks rights at the entry level.

At the entry level, the server checks not whether the entry has granted the group Create rights, but whether it has granted the group *Modify* rights. If the entry has granted Modify rights to its group, and if the principal is a member of that group (which may not be the same as the table object's group), the server replaces the existing entry with the new one and no further checking is done.

If the entry has not granted Modify rights to its group, or if the principal is not a member of that group, an error message is returned, stating that the principal does not have permission to modify the object.

Since no columns can be added to a table, the server does not check column access rights during a Create operation.

Administering NIS+ Credentials 8

This chapter describes how to use the NIS+ credential administration commands to perform the following tasks.

For a complete description of these commands, their syntax, and options, see the NIS+ man pages.

Related Commands

The commands listed above handle most credential-related administration tasks, but two other commands can provide marginally useful information about credentials:

Table 8-1 Additional Credential Related Commands

Comman d	Description	See
niscat -o	Lists an object's properties. By looking in the Public Key field of the object's server, you can tell whether the object definition is storing a public key.	page 168
niscat	Lists the contents of the Cred table. By looking through the entries, you can tell whether a principal has LOCAL and DES credentials.	page 198

Where Credential-Related Information Is Stored

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 8-2, below, lists all the objects, tables, and files that store credential-related information and how to reset it.

Table 8-2 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 who is currently logged in.	Run `keylogin` for a client user or `keylogin -r` for a client workstation.
NIS+ Daemon	Copies of directory objects, which in turn contain copies of their servers' public keys.	Kill the daemon and the cache manager. Then restart both.
Directory cache	A copy of directory objects, which in turn contain copies of their servers' public keys.	Kill the NIS+ cache manager and restart it with the `nis_cachemgr -i` command. The `-i` option resets the directory cache from the coldstart file and restarts the cache manager.

Table 8-2 Where Credential-Related Information is Stored

Item	Stores	To Reset or Change
Coldstart file	A copy of a directory object, which in turn contains copies of its servers' public keys.	In the root master, kill the NIS+ daemon and restart it. The daemon reloads new information into the existing NIS_COLD_START file. In a client, first remove the coldstart and shared directory files from `/var/nis`, and kill the cache manager. Then re-initialize the client with `nisinit -c`. The client's trusted server reloads new information into the client's existing coldstart file.
Passwd Table	A user's password or a workstation's superuser password.	Use the `nispasswd` 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 workstation's superuser password.	Use the `passwd` command, whether logged on as superuser or as yourself, whichever is appropriate.
Passwd Map (NIS)	A user's password or a workstation's superuser password.	Use `yppasswd.`

The `nisaddcred` *Command*

The `nisaddcred` command creates, updates, and removes LOCAL and DES credentials. To create a credential, 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, 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.

Syntax

To create or update credentials for another NIS+ principal:

```
nisaddcred -p uid -P principal-name local
nisaddcred -p rpc-netname -P principal-name des
```

To update your own credentials:

```
nisaddcred des
nisaddcred local
```

To remove credentials:

```
nisaddcred -r principal-name
```

Secure RPC Netname vs NIS+ Principal Name

When creating credentials, you will often have to enter a principal's *rpc-netname* and *principal-name*. It is easy to confuse their syntaxes, so here is an explanation of each.

A secure RPC netname is a name whose syntax is determined by the Secure RPC protocol. Therefore, it does not follow NIS+ naming conventions:

```
rpc-netname    ::= unix.uid@domain  |
                   unix.hostname@domain
```

It always begins with the `unix.` prefix and ends with a domain name. However, the domain name *does not* contain a trailing dot. If it identifies a user, it requires the user's UID. If it identifies a workstation, it requires the workstation's hostname. It is always preceded by the `-p` (lowercase) flag.

An NIS+ principal follows the normal NIS+ naming conventions, but it must always be fully-qualified:

```
principal-name ::= principal.domain.
```

Whether it identifies a client user or a client workstation, it begins with the principal's *name*, followed by a dot and the complete domain name, ending in a dot. When used to create credentials, it is always preceded by the `-P` (uppercase) flag. When used to remove credentials, it does not use the `-P` flag.

A Note About Creating Your Own Credentials

When you try to create your own credentials, you run into a problem of circularity: you cannot create your own credentials unless you have Create rights to your domain's Cred table, but you cannot have such rights until you have credentials. You have to step out of the loop somehow. You can do this in two ways:

- Creating your credentials while logged on as superuser to your domain's master server
- Having another administrator create your credentials using a dummy password, then changing your password with the `chkey` command.

In either case, you are creating credentials for "another" NIS+ principal. Therefore, to create your own credentials, follow the instructions in the task titled "How to Create Credentials for an NIS+ Principal" on page 127

▼ How to Create Credentials for an NIS+ Principal

To create credentials for an NIS+ principal, you must have Create rights to the Cred table of the principal's home domain. In addition, that principal must be recognized by the server. This means that if the principal is a user, he or she 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 workstation, it must have an entry either in the domain's NIS+ Hosts table or in the server's `/etc/hosts` file. Once those conditions are met, you can use the `nisaddcred` command with both the `-p` and `-P` options:

```
nisaddcred -p uid -P principal-name local
nisaddcred -p rpc.netname -P principal-name des
```

Remember that although you can create both LOCAL and DES credentials for a client user, you can only create DES credentials for a client workstation. Also, you can create DES credentials only in the client's home domain, but you can create LOCAL credentials for a client user in other domains, in addition to those of his or her home domain. Following are several examples.

This example creates both LOCAL and DES credentials for an NIS+ principal who is a client user named topadmin and has a UID of 11177. This principal belongs to the Sales.Wiz.Com. domain, so this example enters her credentials from another client of that domain:

```
salesclient# nisaddcred -p 11177 -P topadmin.Sales.Wiz.Com. local
salesclient# nisaddcred -p unix.11177@Sales.Wiz.Com \
                      -P topadmin.Sales.Wiz.Com. des
Adding key pair for unix.11177@Sales.Wiz.Com
        (topadmin.Sales.Wiz.Com.).
Enter login password: enter-login-password
```

The proper response to the login prompt is topadmin's login password or a dummy password that she can later change.

Below is a similar example, but it is more thoroughly commented and shows how another administrator, whose credentials you create, can then use chkey to change his own password. In this example, you create credentials for an administrator named "Bob," who has a UID of 11199. Bob belongs to the root domain, so you would enter his credentials from the root master server.

```
rootmaster# nisaddcred -p 11199 -P bob.Wiz.Com. local    # Create LOCAL credential for Bob.
rootmaster# nisaddcred -p unix.11199@Wiz.Com \           # Create DES credential for Bob.
                       -P bob.Wiz.Com. des               #
Adding key pair for unix.11199@Wiz.Com (bob.Wiz.Com.).   #
Enter bob's login password: enter-dummy-password         # Enter dummy password for Bob.
nisaddcred: WARNING: password differs from login passwd. #
Retype password: re-enter-dummy-password                 # Re-enter dummy password.

% rlogin rootmaster -l bob                               # Bob logs into rootmaster.
Password: enter-login-password                           # Bob enters real login password.
Password does not decrypt secret key for                 # Bob gets error message but is
unix.11199@Wiz.Com.                                      #   allowed to login anyway.
$ keylogin                                               # Bob does keylogin.
Password: enter-dummy-password                           # Bob enters dummy password.
$ chkey                                                  # Bob does chkey.
Updating nisplus publickey database                      #
Generating new key for 'unix.11199@Wiz.Com'.             #
Enter login password: enter-login-password               # Bob enters real login password.
Retype password: re-enter-login-password                 # Bob re-enters real login
Done.                                                    #    password.
```

First, you would create Bob's credentials in the usual way, but using a dummy login password. NIS+ would warn you and ask you to re-enter it. When you did, the operation would be complete. The domain's Cred table would contain Bob's credential information based on the dummy password. The domain's Passwd table (or `/etc/passwd` file), however, would still have the "correct" entry.

Then, Bob would log into the domain's master server, entering his *correct* login password (since the login operation checks the password entry in the Passwd table or `/etc/passwd` file). From there, Bob would first keylogin, using the dummy password (since a keylogin checks the Cred table), and then use the `chkey` command to change the Cred entry to the real thing.

The two previous examples created credentials for a client user while logged onto the master server of the client user's home domain. However, if you have the proper access rights, you can create credentials in another domain. Simply append the domain name to syntax:

```
nisaddcred -p uid -P principal-name local domain-name
nisaddcred -p rpc-netname -P principal-name des domain-name
```

This example first creates LOCAL and DES credentials for an administrator named "Betty" in her home domain, which happens to be the root domain, then adds her LOCAL credentials to the "Sales.Wiz.Com." domain. Betty's UID is 11155. This command is entered from the root master server. For simplicity, it assumes you are entering Betty's correct login password.

```
rootmaster# nisaddcred -p 11155 -P betty.Wiz.Com. local

rootmaster# nisaddcred -p unix.11155@Wiz.Com \
                       -P betty.Wiz.Com. des
Adding key pair for unix.11155@Wiz.Com (betty.Wiz.Com.).
Enter login password: enter-login-password

rootmaster# nisaddcred -p 11155 -P betty.Wiz.Com. local \
                       Sales.Wiz.Com.
```

LOCAL credentials map a UID to an NIS+ principal name. Although an NIS+ principal who 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 "betty" 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.

Here is an example that creates credentials for a client *workstation*. Its hostname is "rootclient1" and it belongs to the root domain. Therefore, its credentials are created from the root master server. In this example, you create them while logged on as root to the root master; however, if you already have valid credentials and the proper access rights, you could create them while logged on as yourself.

```
rootmaster# nisaddcred -p unix.rootclient1@Wiz.Com \
                      -P rootclient1.Wiz.Com. des
Adding key pair for unix.rootclient1@.Wiz.Com
        (rootclient1.Wiz.Com.).
Enter rootclient1.Wiz.Com.'s root login password: enter-password
Retype password: re-enter-password
```

The proper response to the password prompt is the client workstation's superuser password. Of course, you could use a dummy password that would later be changed by someone logged on as superuser to that client workstation.

▼ How to Update Your Own Credentials

Updating your own credentials is considerably simpler than creating them. Simply enter the simple versions of the `nisaddcred` command while logged on as yourself:

```
nisaddcred des
nisaddcred local
```

▼ How to Remove Credentials

The `nisaddcred` command removes a principal's credentials, but both at a time and only from the local domain. To use it, you must have Modify rights to the local domain's Cred table. Use the `-r` option and specify the principal with its NIS+ principal name:

```
nisaddcred -r principal-name
```

The following two examples remove the LOCAL and DES credentials of the administrator "topadmin.Wiz.Com." The first example removes both types of credentials from her home domain ("Wiz.Com."), the second removes her LOCAL credentials from the "Sales.Wiz.Com." domain. Note how they are each entered from the appropriate domain's master servers.

```
rootmaster# nisaddcred -r topadmin.Wiz.Com.

salesmaster# nisaddcred -r topadmin.Wiz.Com.
```

To verify that the credential was indeed removed, run `nismatch` on the Cred table, as shown below. For more information about `nismatch`, see Chapter 12, "Administering NIS+ Tables."

```
rootmaster# nismatch topadmin.Wiz.Com. cred.org_dir
salesmaster# nismatch topadmin.Wiz.Com. cred.org_dir
```

The `chkey` *Command*

The `chkey` command changes an NIS+ principal's public and secret keys stored in the Cred table. It does not affect the principal's entry either in the Passwd table or in the `/etc/passwd` file. In fact, this command is often used to make a principal's entry in the Cred table correspond to its entry in the Passwd table.

The `chkey` command interacts with the keyserver, the Cred table, and the Passwd table. When you invoke it, it tries to identify you with the keyserver. Therefore, you must keylogin before using it. However, the keyserver needs the password that was used to generate your keys in the Cred table. That may be different from your normal login password, depending on how your credentials were created.

For all this to work, you must also have an entry in the Passwd table. If you don't have an entry, or if you forget to keylogin, `chkey` will give you the following error message:

```
chkey: unable to locate password record for uid uid
```

Once it identifies you, it prompts you for a new password, which it then uses to generate a new set of public and private keys. It stores those keys in the Cred table, provided you have Modify rights to it.

▼ How to Change Your DES Keys

To change your DES keys with the `chkey` command, you need:

- Modify rights to your domain's Cred table
- The password from which your entry in the Cred table was formed
- An entry in the domain's Passwd table.
- Your login password.

First, keylogin, using your current password, then use `chkey`. Here is an example:

```
rootmaster% keylogin
Password:  enter-current-password
rootmaster% chkey
Updating nisplus publickey database
Generating new key for 'unix.11199@Wiz.Com'.
Enter login password: enter-new-password
Retype password: re-enter-new-password
Done.
```

If you want to use `chkey` again before you logout, you don't have to keylogin again.

The `nispasswd` *Command*

`nis_passwd` does not function in an NIS+ environment that is running at security level 0.

The `nispasswd` command changes or displays information stored in the NIS+ Passwd table. If you use it to change a principal's actual password, it tries to update the principal's secret key in the Cred table. If you have Modify rights to the Cred table and if the principal's login and network passwords are the same, it will update the keys in the Cred table. Otherwise, it changes the password, but does not change the secret keys. This means that the secret keys in the Cred table will have been formed with a password that is now different from the one stored in the Passwd table. In that case, you'll either have to change the keys with the `chkey` command, or keylogin after each login.

The Name Service Switch determines which processes obtain the new password. If the source for Password information in the Switch is `nisplus`, all processes will use the new password. However, if the source is `nis` or `files`, processes that use the standard `getpwnam` and `getspnam` interfaces (such as `rlogin` and `ftp`), will look elsewhere for the password information, and so will not get the new password stored in the NIS+ table. For more information about the Name Service Switch, see Chapter 6, “Setting Up the Name Service Switch.”

If you are the owner of the passwd table, provided you have the proper credentials, you can change password information at any time and without constraints. However, if you are not the owner, you must comply with aging and construction constraints.

When you attempt to change a password, if the old password has not aged sufficiently (i.e., number of days since last change is less than *min*), NIS+ will terminate and not carry out the change.

The new password must have at least six characters, but no more than eight. It must contain at least two letters and at least one number or special character. Make sure the password is not derived in any way from the user’s login name. Also make sure the new password has at least three characters that are different from the old password.

To use the `nispasswd` command, you must have the access rights appropriate for the operation:

Table 8-3 Access Rights for `nispasswd` Command

This Operation	Requires These Rights	To This Object
Displaying information	Read	The Passwd entry
Changing Information	Modify	The Passwd entry
Adding New Information	Modify	The Passwd Table

Note – `nis_passwd` is designed to function with credentials. If there are no credentials, `nis_passwd` cannot work. Systems running at security level 0 do not create or maintain credentials. Thus, in most cases you cannot use `nis_passwd` on a system running at security level 0. The exception to this rule concerns systems that have previously been run at a higher security level. If credentials from that period of higher level operation still exist in proper form and format, then you can run `nis_passwd` for principals holding those credentials even if the system is running at security level 0.

Related Commands

The `nispasswd` command provides capabilities that are similar to those offered by other commands. The table below summarizes their differences.

Table 8-4 `nispasswd` and Related Commands

Command	**Description**
passwd	Changes information in the workstation's `/etc/passwd` and `/etc/shadow` file.
yppasswd	Changes information in the NIS password map. Has no effect on the NIS+ Passwd table.
nispasswd	Changes and displays information in the NIS+ Passwd table. When a principal's password is changed, `nispasswd` tries to update principal's secret key in the Cred table. Its options are customized for the Passwd tables making the command easier to use for that table than the `nistbladm` command. It also allows an administrator to lock and force passwords, tasks that `nistbladm` doesn't allow.
nistbladm	Creates, changes, and displays information about any NIS+ table, including the Passwd table. Although the `nispasswd` command is easier to use for the Passwd table, `nistbladm` allows you to: - Create new entries - Delete an existing entry - Change the UID and GID fields in the Passwd table - Change the LastChanged, Inactive, and Expired fields in the Passwd table - Change access rights and other security-related attributes of the Passwd table.
niscat	Can be used to display the contents of the Passwd table.

Syntax

—To display information from the Passwd table

```
nispasswd -a
nispasswd -d username
```

—To change a password

```
nispasswd [username]
```

—To operate on the Passwd table of another domain

```
nispasswd [options] -D domainname
```

▼ How to Display Password Information

You can use the `nispasswd` command to display password information about all users in a domain or about one particular user. To display information about all users, use the `-a` option:

```
nispasswd -a
```

Only the entries and columns for which you have Read permission will be displayed. To display the entry for a particular user, use the `-d` option. Without a username, it displays your password entry:

```
nispasswd -d
nispasswd -d username
```

Entries are displayed with the following format:

```
username   status   mm/dd/yy   min   max   warn
```

Table 8-5 NIS+ Password Display Format

Field	Description
username	The user’s login name
status	The user’s password status. "PS" indicates the account has a password. "LK" indicates the account is locked. "NP" indicates the account has no password.
mm/dd/yy	The date, using Greenwich Mean Time, that the user’s password was last changed.

Field	Description
min	The minimum number of days since the last change that must pass before the password can be changed again.
max	The maximum number of days since the last change that can pass before the password must be changed.
warn	The number of days' notice that a user is given before his or her password expires.

To display entries from a passwd table in another domain, use the `-D` option:

```
nispasswd -a -D domainname
nispasswd -d username -D domainname
```

▼ How to Change Passwords

To change your own password, enter the `nispasswd` command without any arguments:

```
nispasswd
```

To change the password of another user in the local domain, add a *username*:

```
nispasswd username
```

To change the password of another user in a remote domain, also add the `-D` flag:

```
nispasswd username -D domainname
```

NIS+ prompts you once for the old password and twice for the new one. If you are the owner of the Passwd table, NIS+ does not prompt you for the old password. If you make a mistake and the two entries for the new password don't match, NIS+ will ask you again for the new password.

The `nisupdkeys` *Command*

The public keys of NIS+ servers are stored in several locations throughout the namespace, as listed in Table 8-2 on page 124. When new credentials are created for the server, a new public key is generated and stored in the Cred table. However, the directory object still has a copy of the old public key. The `nisupdkeys` command is used to update that copy.

The `nisupdkeys` command can update the key of one particular server, or of all the servers that support a directory. It can also update the server's IP address, if that has changed, and it can remove the server's public key from the directory object. However, it cannot update the NIS_COLD_START files on the client workstations. To update their copy, the clients should invoke the `nisinit` command. Or, if the NIS+ cache manager is running and more than one server is available in the coldstart file, they can wait until the time-to-live expires on the directory object. When that happens, the cache manager automatically updates the coldstart file. The default time-to-live is 12 hours.

To use the `nisupdkeys` command, you must have Modify rights to the directory object.

Syntax

Note that the `nisupdkeys` command is located in `/usr/lib/nis`.

—*To update keys:*

```
/usr/lib/nis/nisupdkeys [ directory ]
/usr/lib/nis/nisupdkeys -H server-name
```

—*To update IP addresses:*

```
/usr/lib/nis/nisupdkeys -a [ directory ]
/usr/lib/nis/nisupdkeys -a -H server-name
```

—*To clear keys:*

```
/usr/lib/nis/nisupdkeys -C [ directory ]
/usr/lib/nis/nisupdkeys -C -H server-name
```

—*To display a list of directories served by a server*

```
/usr/lib/nis/nisupdkeys -s [-a|-C] -H servername
```

When a directory name is supplied, `nisupdkeys` updates the public keys of all the servers that support that directory. If the name of a server is supplied instead, it updates the public keys of that server only. If no directory name is supplied, it updates the public keys of the local directory.

▼ How to Update All Public Keys in a Directory

To update the public keys of all servers that support a particular directory, simply provide the directory name:

```
nisupdkeys directory
```

If you don't supply a directory name, it uses the local directory. These two examples update the public keys of the root directory and a directory beneath it. They are both entered from the root master:

```
rootmaster# /usr/lib/nis/nisupdkeys
Fetch Public key for server rootmaster.Wiz.Com.
  netname= 'unix.rootmaster@Wiz.Com'
Updating rootmaster.Wiz.Com.'s public key.
   Public key : public-key
rootmaster# /usr/lib/nis/nisupdkeys Sales.Wiz.Com.
```

▼ How to Update Keys of a Particular Server

To update the public keys of a particular server in all the directories that store them, use the `-H` option and provide the server name:

```
nisupdkeys -H server-name
```

▼ How to Clear Public Keys

To clear all the public keys stored by a directory, use the `-C` option and specify the directory. To clear the public keys of only one server supporting that directory, use the `-C` and `-H` options.

```
nisupdkeys -C directory
nisupdkeys -C -H server-name
```

▼ How to Update IP Addresses

To update the IP addresses of one or more servers, use the `-a` option.

```
nisupdkeys -a directory
nisupdkeys -a -H server-name
```

The `keylogin` *Command*

The `keylogin` command helps authenticate an NIS+ principal. When a principal logs in, the login process prompts for a password, which is used to authenticate the principal. Normally, this is the only time the principal is asked to provide a password. However, if the principal’s DES credentials were created with a password that is different from the login password, the login password will no longer be able to authenticate the principal.

To remedy this problem, the principal must perform a keylogin, using the `keylogin` command, after every login. The `keylogin` command prompts the principal for its network password and stores it in the key server. From there, it is used by all NIS+ processes to authenticate the principal.

Syntax

```
keylogin
keylogin -r
```

The `-r` flag is used to keylogin a client workstation and to store the superuser’s key in `/etc/.rootkey`.

▼ How to Keylogin

To keylogin, simply enter the `keylogin` command and, when prompted, supply your network password; i.e., the password used to create your DES credentials. Before logging out, use the `keylogout` command. This example shows how to keylogin for a client user and a client workstation:

```
Client1% login: login-name
Password: login-password
Client1% keylogin
Password: network-password
Client1% su
Password: superuser-password
Client1# keylogin -r
Password: superuser-network-password
Wrote secret key into /etc/.rootkey
```

Administering NIS+ Access Rights 9

This chapter describes how to use the NIS+ access rights administration commands to perform the following tasks:

For a complete description of these commands, their syntax, and options, see the NIS+ man pages.

Specifying Access Rights in Commands

This chapter assumes an NIS+ environment running at security level 2 (the default).

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

Access rights, whether specified in an environment variable or a command, are identified with three types of arguments: *class*, *operator,* and *right*.

class:=	`n	o	g	w	a`
operator:=	`+	-	=`
right:=	`r	m	c	d`

The *class* refers to the type of NIS+ principal to which the *rights* will apply. The *operator* indicates the operation that will be performed with the *rights*. The *rights* are the access rights themselves. The accepted values for each are listed below.

Table 9-1 Access Rights Syntax

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	**Description**
+	Add the access rights specified by *right*.
-	Revoke the access rights specified by *right*.
=	Explicitly set the access rights specified by *right*; that is, revoke all existing rights and replace them with the new access rights.

Right	Description
r	Read the object definition or table entry
m	Modify the object definition or table entry
c	Create a table entry or column
d	Destroy a table entry or column

This example *adds* the *Read* access right to the *Owner.*

```
o+r
```

This example *adds* the *Modify* access right to the *Owner, Group,* and *World.*

```
a+m
```

This example *sets* the rights of the *Owner, Group,* and *World* to *Read.* This means that all three classes of principal now only have Read access, regardless of what access rights they had before:

```
a=r
```

This example *adds* the *Create* and *Destroy* rights to the *Owner.*

```
o+cd
```

This example *adds* the *Read* and *Modify* rights to the *World* and *Nobody.*

```
wn+rm
```

This example *removes* all four rights from the *Group, World,* and *Nobody.*

```
gwn-rmcd
```

This example combines the first two operations.

```
o+cd,wn+rm
```

Syntax for Owner and Group

To specify an owner, use an NIS+ principal name. To specify an NIS+ group, use an NIS+ group name with the domain name appended.

owner	`::=` *principal-name*`.`*domain-name*`.`
group	`::=` *group-name*`.`*domain-name*`.`

Syntax for Objects and Entries

Objects and entries use different syntaxes. Objects use simple object names, while table entries use indexed names.

object	`::=` *object-name*
entry	`::=` `[`*column-name*`=`*value*`, ...]`, *table-name*

For example:

```
hosts.org_dir.Sales.Wiz.Com.
[name=butler],hosts.org_dir.Eng.Wiz.Com.
[uid=33555],passwd.org_dir.Eng.Wiz.Com.
```

Note – In this case the brackets are part of the syntax, not just the "optional" symbol of the grammar.

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. This example uses two pairs to specify the entry:

```
[winner=yoyoma,year=1992],races.org_dir.Wiz.Com.
```

Columns use a special version of indexed names. Because you can only work on columns with the `nistbladm` command, see "The nistbladm Command" on page 190 for more information.

The `nisdefaults` *Command*

The `nisdefaults` command displays the seven defaults currently active in the namespace: domain, group, host, principal, access rights, directory search path, and time-to-live. NIS+ supplies preset values for these defaults. They are listed under "*Options*," below. In addition, you can specify your own security-related defaults (owner, group, access rights, and time-to-live) 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. This section describes how to perform tasks related to the `nisdefaults` command, the `NIS_DEFAULTS` environment variable, and the `-D` option.

Syntax

—To display individual defaults:

```
nisdefaults [-dghprst]
nisdefaults [-dghprst] [-v]
```

—To display all defaults:

```
nisdefaults [-a]
```

— Options:

Table 9-2 `nisdefaults` Options

-d	Domain. Displays the home domain of the workstation from which the command was entered.
-g	Group. Displays the group that would be assigned to the next object created from this shell. The group is taken from the NIS_GROUP environment variable.
-h	Hostname. Displays the workstation's hostname.
-p	Principal. Displays the user ID or hostname of the NIS+ principal who entered the `nisdefaults` command.
-r	Rights. Displays the access rights that will be assigned to the next object or entry created from this shell. They are: `----rmcdr---r---`
-s	Search Path. Displays the syntax of the search path, which indicate the domains that NIS+ will search through when looking for information.
-t	Time to live. Displays the time to live that will be assigned to the next object created from this shell. The default is 12 hours.

Displaying Default Values

You can display all default values or any subset of them. To display all values, enter the `nisdefaults` command without arguments. They are displayed in verbose format. To use terse format, add the `-a` option. Here is an example:

```
rootmaster% nisdefaults
Principal Name : topadmin.Wiz.Com.
Domain Name    : Wiz.Com.
Host Name      : rootmaster.Wiz.Com.
Group Name     :
Access Rights  : ----rmcdr---r---
Time to live   : 12:0:0
Search Path    : $Wiz.Com.
```

To display a subset of the values, use the appropriate options. The values are displayed in terse mode. To display them in verbose mode, add the `-v` flag.

```
rootmaster% nisdefaults -rs
----rmcdr---r---
$Wiz.Com.
```

Changing Defaults

You can change the default access rights, owner, and group by changing the value of the `NIS_DEFAULTS` environment variable. Use the command that is appropriate for your shell with the `access=`, `owner=`, and `group=` arguments:

`access=`*right* ...
`owner=`*principal-name*
`group=`*group-name*

You can combine two or more arguments into one line:

`owner=`*principal-name*`:group=`*group-name*

Here are some examples:

```
client% setenv NIS_DEFAULTS access=o+r
client% setenv NIS_DEFAULTS owner=abe.Wiz.Com.
client% setenv NIS_DEFAULTS access=o+r:owner=abe.Wiz.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; the column simply inherits the defaults of the table.

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
```

Resetting the Value of `NIS_DEFAULTS`

You can reset the NIS_DEFAULTS variable back to its original values (listed on page 145), by entering the name of the variable without arguments, using the format appropriate to your shell:

```
client# unsetenv NIS_DEFAULTS                        # for csh

client# NIS_DEFAULTS=; export NIS_DEFAULTS          # for sh/ksh
```

Overriding Defaults

You can override default access rights, owner, and group, any time that you create an NIS+ object or table entry with any of these NIS+ commands:

- `nismkdir` - creates NIS+ directory objects
- `nisaddent` - transfers entries into an NIS+ table
- `nistbladm` - creates entries in an NIS+ table

Insert the `-D` option into the syntax of those commands, as shown below:

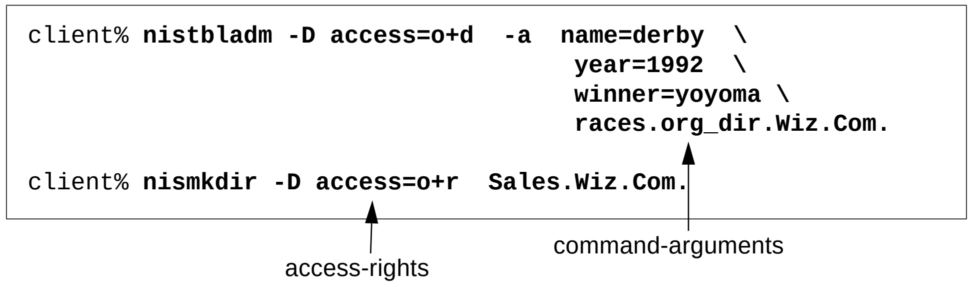
command `-D access=`*right* . . . *command-arguments*
command `-D owner=`*principal-name* *command-arguments*
command `-D group=`*group-name* *command-arguments*

As when setting defaults, you can combine two or more arguments into one line:

command `-D owner=`*principal-name*`:group=`*group-name* `\`
command-arguments

Remember that a column's owner and group are always the same as its table, so you cannot override them.

These two examples override the default access rights:



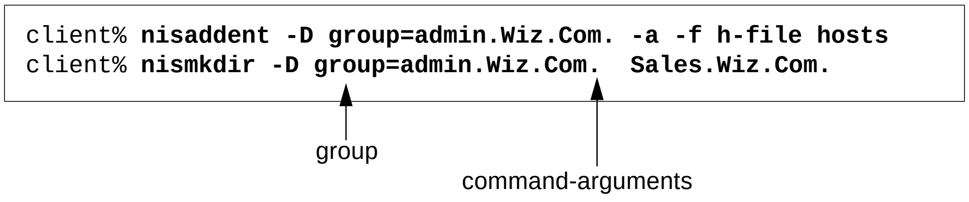
```
client% nistbladm -D access=o+d  -a  name=derby  \
                                      year=1992  \
                                      winner=yoyoma \
                                      races.org_dir.Wiz.Com.

client% nismkdir -D access=o+r  Sales.Wiz.Com.
```

Figure 9-1 Overriding Default Access Rights, Example

These two examples override the default owner:

```
Client% nistbladm -D owner=abe.Wiz.Com.  -a  name=derby  \
                                              year=1992  \
                                              winner=yoyoma \
                                             races.org_dir.Wiz.Com.

client% nismkdir -D owner=abe.Wiz.Com.  Sales.Wiz.Com.
```

These two examples override the default group:



```
client% nisaddent -D group=admin.Wiz.Com. -a -f h-file hosts
client% nismkdir -D group=admin.Wiz.Com.  Sales.Wiz.Com.
```

Figure 9-2 Overriding the Default Group, Example

This example overrides the default owner and group:

```
client% nismkdir -D owner=abe.Wiz.Com.:group=admin.Wiz.Com. \
                  Sales.Wiz.Com.
```

The `nischmod` *Command*

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.

Syntax

—*To add rights for an object or entry:*

`nischmod` *class*. . .+*right*. . . *object-name.*
`nischmod` *class*. . .+*right*. . . [*column-name=value*],*table-name*

—*To remove rights for an object or entry:*

`nischmod` *class*. . .-*right*. . . *object-name.*
`nischmod` *class*. . .-*right*. . . [*column-name=value*],*table-name*

Adding Rights to an Object

To add access rights to an NIS+ object, use the + operator:

`nischmod` *class*. . .+*right*. . . *object-name.*

This example adds Read and Modify rights to the Group of the Sales.Wiz.Com. directory object.

```
client% nischmod  g+rm   Sales.Wiz.Com.
```

Removing Rights to an Object

To remove access rights to an NIS+ object, use the `-` operator:

`nischmod` *class*`...`-*right*`...` *object-name*

This example removes Create and Destroy rights from the Group of the Sales.Wiz.Com. directory object.

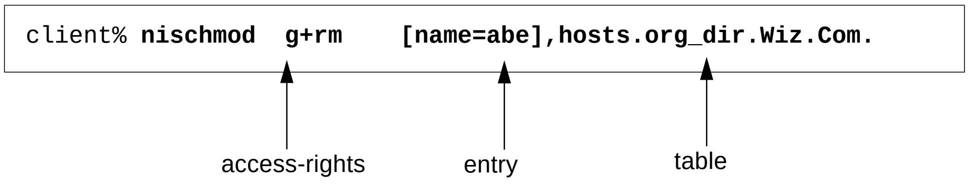
```
client% nischmod g-cd   Sales.Wiz.Com.
```

Adding Rights to a Table Entry

To add access rights to an entry in an NIS+ table, use the `+` operator and an indexed name:

`nischmod` *class*`...`+*right*`...` `[`*column-name*=*value*`],`*table-name*

This example adds Read and Modify rights to Group for an entry in the hosts.org_dir.Wiz.Com. table. The entry is the one whose `hostname` column has the value of `abe`:



```
client% nischmod  g+rm    [name=abe],hosts.org_dir.Wiz.Com.
```

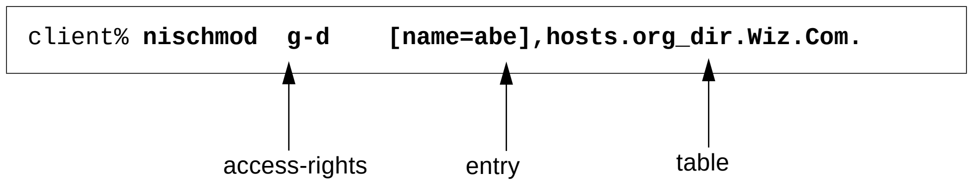
Figure 9-3 Adding Rights to a Table Entry, Example

Removing Rights to a Table Entry

To remove access rights to an entry in an NIS+ table, use the `-` operator and an indexed name:

```
nischmod class...-right...    [column-name=value],table-name
```

This example removes Destroy rights from Group for an entry in the “hosts.org_dir.Wiz.Com.” table. The entry is the one whose `hostname` column has the value of `abe`:



```
client% nischmod  g-d    [name=abe],hosts.org_dir.Wiz.Com.
```

Figure 9-4 Removing Rights to a Table Entry, Example

The `nistbladm` *Command*

The `nistbladm` command performs a variety of operations on NIS+ tables, as described in Chapter 12, “Administering NIS+ Tables.” However, two of its options, `-c` and `-u`, enable you to perform several security-related tasks. To use the `-c` option, you must have create rights to the directory under which you will create the table. To use the `-u` option, you must have Modify rights to the table column.

Syntax

—To set column rights when creating a table:

```
nistbladm -c type column=access-rights... table-name
```

—To change rights to a particular column:

```
nistbladm -u [column=access-rights,...],table-name
```

Setting Column Rights When Creating a Table

When a table is created, its columns are assigned the same rights as the table object. To assign a column its own set of rights, append *access-rights* to each column's equal sign and separate the columns with a space:

column=*access-rights*   *column*=*access-rights*   *column*=*access-rights*

Here is the full syntax:

`nistbladm -c` *type column*=*access-rights*`...`   *table-name*

This example creates a table with three columns and adds Modify rights for the World to the second and third columns:

```
client% nistbladm -c races.org_dir.Wiz.Com. \
                     name=S year=S,w+m winner=S,w+m races
```

For more information about the `nistbladm -c` option, see Chapter 12, "Administering NIS+ Tables."

Adding Rights to an Existing Table Column

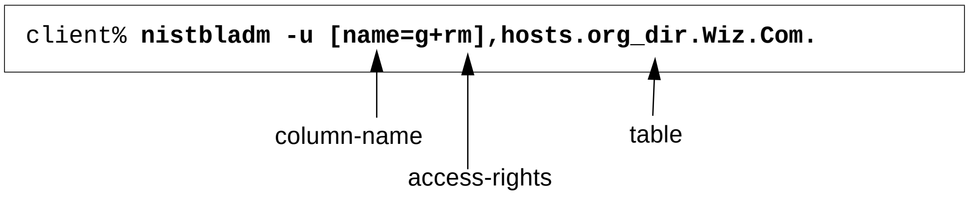
To add access rights to a column in an existing NIS+ table, use the `u` option (its full syntax is described in Chapter 12, "Administering NIS+ Tables"). Use one *column*=*access-rights* pair for each column whose rights you want to update. To update multiple columns, separate them with commas and enclose the entire set with square brackets:

[*column*=*access-rights*]
[*column*=*access-rights*, *column*=*access-rights*]

Here is the full syntax:

`nistbladm -u [`*column*=*class*`...`+*right*`...],` *table-name*

This example adds Read and Modify rights to Group for the “hostname” column in the “hosts.org_dir.Wiz.Com.” table.



```
client% nistbladm -u [name=g+rm],hosts.org_dir.Wiz.Com.
```

Figure 9-5 Adding Access Rights to a Table Column, Example

This example adds Read and Modify rights to Group for two columns in the Hosts table of the Wiz.Com. domain:

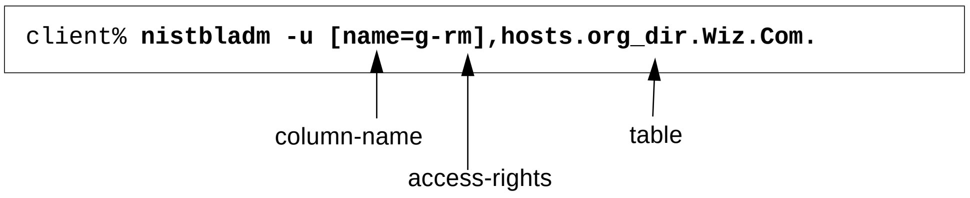
```
client% nistbladm -u  \
        [name=g+rm,addr=g+rm],hosts.org_dir.Wiz.Com.
```

Removing Rights to a Table Column

To remove access rights to a column in an NIS+ table, use the u option, the - operator, and an indexed name:

`nistbladm -u [`*column=class...-access-rights...*`],`*table-name*

This example removes Group’s Read and Modify rights to the “hostname” column in the hosts.org_dir.Wiz.Com. table.



```
client% nistbladm -u [name=g-rm],hosts.org_dir.Wiz.Com.
```

Figure 9-6 Removing Rights to a Table Column, Example

The `nischown` Command

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.

Syntax

—*To change an object's owner:*

`nischown` *new-owner object-name*`...`

—*To change an entry's owner:*

`nischown` *new-owner* `[`*column=value*`,...],`*table-name*`...`

Changing an Object's Owner

To change an object's owner, use the following syntax:

`nischown` *new-owner object-name*

Be sure to append the domain name to both the object name and new owner name. This example changes the owner of the Hosts table in the Wiz.Com. domain to grant.Wiz.Com.:

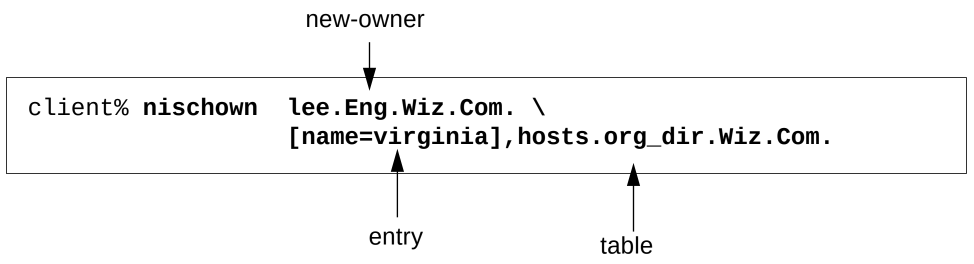
```
client% nischown  grant.Wiz.Com.  hosts.org_dir.Wiz.Com.
```

Changing a Table Entry's Owner

To change a table entry's owner, use an indexed name for the entry, as shown below (this syntax is fully described on page 144):

`nischown` *new-owner* `[`*column=value*`,...],`*table-name*

Be sure to append the domain name to both the new owner name and the table name. This example changes the owner of an entry in the Hosts table of the Wiz.Com. domain to lee.Eng.Wiz. The entry is the one whose value in the hostname column is virginia.



```
client% nischown  lee.Eng.Wiz.Com. \
                  [name=virginia],hosts.org_dir.Wiz.Com.
```

Figure 9-7 Changing a Table's Owner, Example

The `nischgrp` *Command*

The `nischgrp` command changes the group owner 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.

Syntax

—*To change an object's group:*

`nischgrp` *new-group object-name*`...`

—*To change an entry's group:*

`nischgrp` *new-group* `[`*column=value*`,...],`*table-name*`...`

Changing an Object's Group

To change an object's group, use the following syntax:

`nischgrp` *new-group object-name*

Be sure to append the domain name to both the object name and new group name. This example changes the group of the Hosts table in the Wiz.Com. domain to admins.Wiz.Com.:

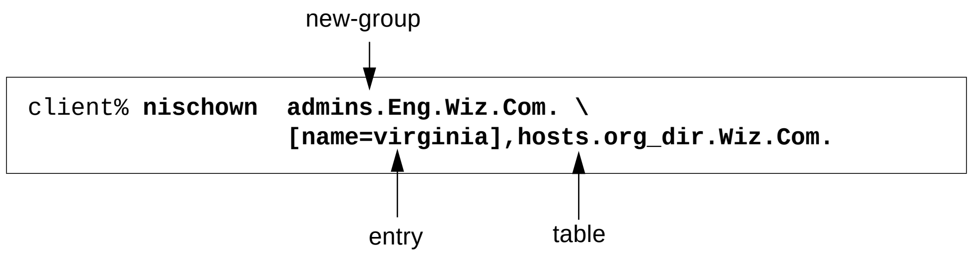
```
client% nischown  admins.Wiz.Com.  hosts.org_dir.Wiz.Com.
```

Changing a Table Entry's Group

To change a table entry's group, use an indexed name for the entry, as shown below (this syntax is fully described on page 144):

`nischgrp` *new-group* `[`*column=value*`, . . .],`*table-name*

Be sure to append the domain name to both the new group name and the table name. This example changes the group of an entry in the Hosts table of the Wiz.Com. domain to admins.Eng.Wiz.Com. The entry is the one whose value in the hostname column is "virginia."



```
client% nischown  admins.Eng.Wiz.Com. \
                  [name=virginia],hosts.org_dir.Wiz.Com.
```

Figure 9-8 Changing a Table Entry's Group, Example

Administering NIS+ Groups 10

This chapter describes how to use NIS+ group administration commands to perform the following tasks:

For a complete description of these commands, their syntax, and options, see the NIS+ man pages.

Related Commands

You can also use Administration Tool's Database Manager to add, modify, or delete groups, and to add or delete members of groups.

The `nisgrpadm` command performs most group administration tasks, but several other commands affect groups as well:

Table 10-1 Commands that Affect Groups

Command	**Description**	**See**
nissetup	Creates, among other things, the directory in which a domain's groups are stored: groups_dir	page 205
nisls	Lists the contents of the groups_dir directory; in other words, all the groups in a domain	page 169
nischgrp	Assigns a group to any NIS+ object	page 155
nisdefault s	Lists, among other things, the group that will be assigned to any new NIS+ object.	page 144

Specifying Group Members in All Commands

NIS+ groups can have three types of members: explicit, implicit, and recursive:

Figure 10-1 Types of Members in NIS+ Groups:

```
member    ::=explicit-member
             implicit-member
             recursive-member
explicit-member ::= principal-name
implicit-member::= *.domain-name
recursive-member::= @group-name.domain-name
```

Explicit members are individual NIS+ principals. They are identified, in all group administration commands, by their principal name. The name does not have to be fully qualified if entered from its default domain.

Implicit members are 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 members are 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.

Non-Members

NIS+ groups also accept non-members in all three categories, explicit, implicit, and recursive. Non-members are identified by a minus sign in front of their name:

Figure 10-2 Types of Non-Members in NIS+ Groups:

```
explicit-non-member ::= -principal-name
implicit-non-member ::= -*.domain-name
recursive-non-member::= -@group-name.domain-name
```

Using `niscat` *With Groups*

The `niscat -o` command can be used to list the object properties 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 misc.groups_dir.Wiz.Com.
Object Name   : misc
Owner         : rootmaster.Wiz.Com.
Group         : admin.Wiz.Com.
Domain        : groups_dir.Wiz.Com.
Access Rights : ----rmcdr---r---
Time to Live  : 1:0:0
Object Type   : GROUP
Group Flags   :
Group Members : rootmaster.Wiz.Com.
                topadmin.Wiz.Com.
                @.admin.Wiz.Com.
                *.Eng.Wiz.Com.
```

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. See the syntax below for an explanation. A better arranged list of members is provided by the `nisgrpadm -l` command, on page 164.

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 10-2 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

Syntax

The `nisgrpadm` has two main forms, one for working with groups, one for working with group members.

To create or delete a group, or to lists its members:

```
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:

```
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.domain-name
```

The example below creates three groups named admin. The first is in the Wiz.Com. domain, the second in Sales.Wiz.Com., and the third in Eng.Wiz.Com. All three are created from the master server of their respective domains.

```
rootmaster# nisgrpadm -c admin.Wiz.Com.
Group “admin.Wiz.Com.” created.
salesmaster# nisgrpadm -c admin.Sales.Wiz.Com.
Group “admin.Sales.Wiz.Com.” created.
engmaser# nisgrpadm -c admin.Eng.Wiz.Com.
Group “admin.Eng.Wiz.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 9, “Administering NIS+ Access Rights). Used without options, it provides this output:

```
rootmaster# nisdefaults
Principal Name : rootmaster.Wiz.Com.
Domain Name    : Wiz.Com.
Host Name      : rootmaster.WIz.Com.
Group Name     :
Access Rights  : ----rmcdr---r---
Time to live   : 12:0:0
Search Path    : Wiz.Com.
```

The owner is listed in the “Principal Name:” field. The owning group is listed only if you have set the NIS_GROUP environment variable.

Of course, 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.Wiz.Com. \
             -c admin.Sales.Wiz.Com.
Group “admin.Sales.Wiz.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.Wiz.Com. group.

```
salesmaster% nisgrpadm -d test.Sales.Wiz.Com.
Group "test.Sales.Wiz.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 earlier, 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 grace and beth, both from the default domain, Alma.Com., and the principals nahny and umpa, from the Villas.Com. domain, to the group diapers.Alma.Com.

```
                        group             explicit members
client% nisgrpadm -a diapers grace beth   \
                       nahny.Villas.Com. umpa.Villas.Com.
Added "grace.Alma.Com." to group "diapers.Alma.Com."
Added "beth.Alma.Com." to group "diapers.Alma.Com."
Added "nahny.Villas.Com." to group "diapers.Alma.Com."
Added "umpa.Villas.Com." to group "diapers.Alma.Com."
```

To verify the operation, use the `nisgrpadm -l` option. Look for the members under the Explicit members category.

This example adds all the NIS+ principals in the Wiz.Com. domain to the all.Wiz.Com. group. It is entered from a client in the Wiz.Com. domain. Note the * symbol *and the dot* in front of the domain name.

group   implicit member

```
client% nisgrpadm -a all  *.Wiz.Com.
Added “*.Wiz.Com.” to group “all.Eng.Wiz.Com.”
```

This example adds the NIS+ group admin.Wiz.Com. to the admin.Eng.Wiz.Com. group. It is entered from a client of the Eng.Wiz.Com. domain. Note the @ symbol in front of the group name.

group   recursive member

```
client% nisgrpadm -a admin  @admin.Wiz.Com.
Added “@admin.Wiz.Com.” to group “admin.Eng.Wiz.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.Eng.Wiz.Com.” group. It is entered from a client in the Eng.Wiz.Com. group:

```
client% nisgrpadm -l admin
Group entry for “admin.Eng.Wiz.Com.” group:
    No explicit members
    No implicit members:
    Recursive members:
         @admin.Wiz.Com.
    No explicit non-members
    No implicit non-members
    No recursive non-members
```

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 “grace” and “nahny.Villas.Home.Com.” from the “diapers.Alma.Home.Com.” group. It is entered from a client in the Alma.Home.Com. domain:

```
client% nisgrpadm -r diapers grace nahny.Villas.Home.Com.
Removed “grace.Alma.Home.Com.” from group
   “diapers.Alma.Home.Com.”.
```

This example removes the admin.Wiz.Com. group from the admin.Eng.Wiz.Com. group. It is entered from a client in the Eng.Wiz.Com. domain:

```
client% nisgrpadm -r admin @admin.Wiz.Com.
Removed “@admin.Wiz.Com.” from group “admin.Eng.Wiz.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.Wiz.Com. group. It is entered from a client in the Wiz.Com. domain.

```
client% nisgrpadm -t admin topadmin
“topadmin.Wiz.Com.” is a member of group “admin.Wiz.Com.”.
```

This example tests whether the NIS+ principal joe, from the Sales.Wiz.Com. domain, belongs to the admin.SalesWiz.Com. group. It is entered from a client in the Wiz.Com. domain.

```
client% nisgrpadm -t admin.Sales.Wiz.Com. joe.Sales.Wiz.Com.
“joe.Sales.Wiz.Com.” is a member of group “admin.Sales.Wiz.Com.”.
```

Administering NIS+ Directories 11

This chapter describes how to use the NIS+ directory administration commands to perform the following tasks:

For a complete description of these commands, their syntax, and options, see the NIS+ man pages.

Listing the Directories Servered by a Server

The `nisupdkeys -s` command can be used to list the directories that have been served by a particular master or replica server. The full syntax is:

```
/usr/lib/nis/nisupdkeys -s [-a|-C] -H servername
```

Using `niscat` *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 Wiz.Com.
Object Name   : Wiz
Owner         : rootmaster.Wiz.Com.
Group         :
Domain        : Com.
Access Rights : r---rmcdr---r---
Time to Live  : 24:0:0
Object Type   : DIRECTORY
.
.
.
```

The `nisls` *Command*

The `nisls` command lists the contents of an NIS+ directory. To use it, you must have Read rights to the directory object.

Syntax

—To display in terse format:

```
nisls
nisls [-dgLmMR] directory-name
```

—To display in verbose format:

```
nisls -l [-gm] [-dLMR] directory-name
```

— *Options:*

Table 11-1 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.
-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]
nisls [-dLMR] directory-name
```

For example, this instance of `nisls` is entered from the root master server of the root domain "Wiz.Com.":

```
rootmaster% nisls
Wiz.Com.:
org_dir
groups_dir
```

Here is another example entered from the root master server:

```
rootmaster% nisls -R Sales.Wiz.Com.
Sales.Wiz.Com.:
org_dir
groups_dir

groups_dir.Sales.Wiz.Com.:
admin

org_dir.Sales.Wiz.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]
nisls -l [-gm] [-dLMR] directory-name
```

Here is an example, entered from the master server of the root domain "Wiz.Com.":

```
rootmaster% nisls -l
Wiz.Com.:
D r---rmcdr---r--- rootmaster.Wiz.Com. date org_dir
D r---rmcdr---r--- rootmaster.Wiz.Com. date groups_dir
```

The `nismkdir` Command

The `nismkdir` command creates a non-root NIS+ directory and associates it with a master server. (To create a root directory, use the `nisinit -r` command, described on page 178.) 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. For a complete description, see Chapter 4, "Setting Up a Non-Root Domain."

Syntax

—*To create a directory:*

`nismkdir [-m` *master-server*`]` *directory-name*

— *To add a replica to an existing directory:*

nismkdir -s *replica-server directory-name*

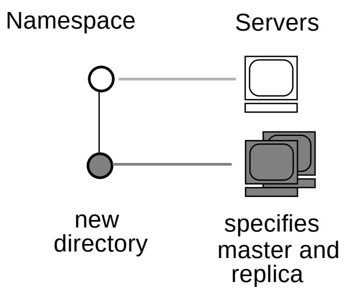
nismkdir -s *replica-server org_dir.directory-name*

nismkdir -s *replica-server groups_dir.directory-name*

Creating a Directory

To create a directory, you must have Create rights to its parent directory. First use the `-m` option to identify the master server and then the `-s` option to identify the replica:

```
nismkdir -m master directory
nismkdir -s replica directory
```

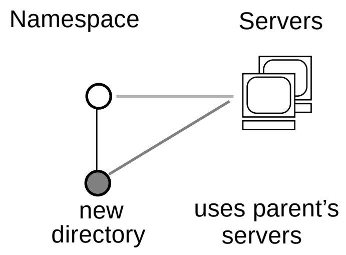


This example creates the Sales.Wiz.Com. directory and specifies its master server, "salesmaster.Wiz.Com." and its replica, "rep1.Wiz.Com." It is entered from the root master server.

```
rootmaster% nismkdir -m salesmaster.Wiz.Com. Sales.Wiz.Com.
rootmaster% nismkdir -s rep1.Wiz.Com. Sales.Wiz.Com.
rootmaster% nismkdir -s rep1.Wiz.Com. org_dir.Sales.Wiz.Com.
rootmaster% nismkdir -s rep1.Wiz.Com. groups_dir.Sales.Wiz.Com.
```

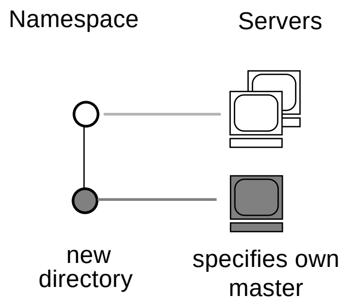
The `nismkdir` command allows you to use the parent directory's servers for the new directory, instead of specifying its own. However, this should not be done except in the case of small networks. Here are two examples:

```
rootmaster% nismkdir Sales.Wiz.Com.
```



The first example creates the "Sales.Wiz.Com." directory and associates it with its parent directory's master and replica servers.

```
rootmaster% nismkdir -m salesmaster.Wiz.Com. Sales.Wiz.Com.
```



The second example creates the "Sales.Wiz.Com." directory and specifies its own master server, "salesmaster.Wiz.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.Wiz.Com." domain already existed, the `nismkdir` command as shown above would have made "salesmaster.Wiz.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 sysem using the `nismkdir` command. An easier way to do this is with the `nisserver` script as described in *Name Services Configuration Guide*.

To assign a new replica server to an existing directory, use the `-s` option and the name of the existing directory:

`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:

```
rootmaster% nismkdir -s rep1.Wiz.Com. Wiz.Com.
rootmaster% nismkdir -s rep1.Wiz.Com. org_dir.Wiz.Com.
rootmaster% nismkdir -s rep1.Wiz.Com. groups_dir.Wiz.Com.
```

Note that you cannot assign a server to support its parent domain —unless, of course, it belongs to the root domain.

The `nisrmdir` *Command*

The `nisrmdir` command can remove a directory or simply disassociate a replica server from a directory. 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 disassociate a replica server from a directory, you must have Modify rights to the directory.

Removing a Directory

To remove an entire directory and disassociate its master and replica servers, use the `nisrmdir` command without any options:

`nisrmdir` *directory-name*

This example removes the Eng.Wiz.Com. directory from beneath the Wiz.Com. directory:

```
rootmaster% nisrmdir Eng.Wiz.Com.
```

Disassociating a Replica From a Directory

To disassociate a replica server from a directory, use the `nisrmdir` command with the `-s` option:

`nisrmdir -s` *servername directory*

This example disassociates the engreplica1 server from the Eng.Wiz.Com. directory:

```
rootmaster% nisrmdir -s engreplica1 Eng.Wiz.Com.
```

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 non-empty 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 page 163), and you can empty tables with `nistbladm -r` or `nistbladm -R` (see page 192).

Syntax

— *To remove a non-directory object:*

`nisrmdir [-if]` *object-name*

— *Options:*

Table 11-2 `nisrmdr` Syntax Options

-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 Non-Directory Objects

To remove non-directory 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.Wiz.Com. groups.org_dir.Wiz.Com.
Remove admins.Wiz.Com.? y
Remove groups.org_dir.Wiz.Com.? y
```

The `rpc.nisd` *Command*

The `rpc.nisd` command starts the NIS+ daemon. The daemon can run in NIS-compatibility mode, which enables it to answer requests from NIS clients as well. 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 Chapter 1, "Setting Up the Root Domain," and Chapter 3, "Setting Up NIS+ Servers."

By default, the NIS+ daemon starts with security level 2.

Syntax

— *To start the daemon:*

```
rpc.nisd [-r]
```

— *To start the daemon in NIS-compatibility mode:*

```
rpc.nisd [-r] -Y [-B]
```

— *To start an NIS-compatible daemon with DNS forwarding capabilities:*

```
rpc.nisd [-r] -Y -B
```

— *Options:*

Table 11-3 `rpc.nisd` Syntax Options

`-S` *security-level*	Specifies a security level.
-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.

Starting the NIS+ Daemon

To start the NIS+ daemon on any server except the root master, use the command without options:

```
rpc.nisd
```

The daemon starts with security level 2, which is the default.

To start the daemon with security level 0 or 1, use the `-S` flag:

```
rpc.nisd -S level
```

Starting a NIS-Compatible Daemon

You can start the NIS+ daemon in NIS-compatibility mode in any server, including the root master. Use the `-Y` (uppercase) option:

```
rpc.nisd -Y
```

If the server is rebooted, the daemon will not restart in NIS-compatibility mode unless you also uncomment the line that contains ‘`EMULYP=”Y”`’ in the server’s `/etc/init.d/rpc` file.

To start the daemon with a security level 0 or 1, use the `-S` flag:

```
rpc.nisd -Y -S level
```

Start a DNS-Forwarding NIS-Compatible Daemon

You can add DNS forwarding capabilities to an NIS+ daemon running in NIS-compatibility mode by adding the `-B` option to `rpc.nisd`:

```
rpc.nisd -Y -B
```

If the server is rebooted, the daemon will not restart in DNS-forwarding NIS-compatibility mode unless you also uncomment the line that contains ‘EMULYP=”-Y”’ in the server’s `/etc/init.d/rpc` file and change it to:

```
EMULYP=”-Y -B”
```

Stopping the NIS+ Daemon

To stop the NIS+ daemon, whether it is running in normal or NIS-compatibility mode, kill it like you would any other daemon. First find its process ID, then kill it. Here is an example:

```
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
rootmaster# kill 1081
```

The `nisinit` Command

The `nisinit` command initializes a workstation to be an NIS+ client. 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. They are described in Chapter 1, "Setting Up the Root Domain," and Chapter 2, "Setting Up NIS+ Clients."

Syntax

—To initialize a client:

```
nisinit -c -B
nisinit -c -H hostname
nisinit -c -C filename
```

—To initialize a root master server:

```
nisinit -r
```

Initializing a Client

You can initialize a client in three different ways:

- By hostname
- By broadcast
- By coldstart file

Each way has different prerequisites and associated tasks. For instance, before you can initialize a client by hostname, the client's `/etc/hosts` file must list the hostname you will use. Complete instructions for each method, including prerequisites and associated tasks, are provided in Chapter 2, "Setting Up NIS+ Clients." Following is a summary of the steps that use the `nisinit` command.

To initialize a client by hostname, use the `-c` and `-H` options, and include the name of the server from which the client will obtain its coldstart file:

```
nisinit -c -H hostname
```

To initialize a client by coldstart file, use the `-c` and `-C` options, and provide the name of the coldstart 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
```

The `nis_cachemgr` Command

The `nis_cachemgr` command starts the NIS+ cache manager program, which 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.

When started, the cache manager obtains its initial information from the client's coldstart file, and downloads it into the `/var/nis/NIS_SHARED_DIRCACHE` file.

The cache manager makes requests as a client workstation. Make sure the client workstation has the proper credentials, or instead of improving performance, the cache manager will degrade it.

Starting the Cache Manager

To start the cache manager, simply enter the `nis_cachemgr` command:

```
client% nis_cachemgr
client% nis_cachemgr -i
```

Without the `-i` option, the cache manager is restarted, but it retains the information in the `/var/nis/NIS_SHARED_DIRCACHE` file. The information in the coldstart file is simply appended to the existing information in the file. The `-i` option clears the cache file and re-initializes it from the contents of the client's coldstart file.

To stop the cache manager, kill it as you would any other process.

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

Cold Start directory:
Name : 'Wiz.Com.'
Type : NIS
Master Server :
         Name       : rootmaster.Wiz.Com.
         Public Key : Diffie-Hellman (196 bits)
         Universal addresses (6)
         .
         .
         .
Replicate:
         Name       : rootreplica1.Wiz.Com.
         Public Key : Diffie-Hellman (196 bits)
         Universal addresses (6)
         .
         .
         .
Time to live : 12:0:0
Default Access Rights :
```

The `nisping` Command

The `nisping` command sends a ping to replica servers, telling them to ask the master server for updates immediately. (The replicas normally wait a couple of minutes before executing this request.) Before pinging, the command checks the time of the last update received by each replica. If it is the same as the last update sent by the master, it does not send the ping to the replica.

The `nisping` command can also checkpoint a directory. This consists of telling each server in the directory, including the master, to update its information on disk from the domain's transaction log.

Syntax

To display the time of the last update:

```
/usr/lib/nis/nisping -u [domain]
```

To ping replicas:

```
/usr/lib/nis/nisping [domain]
/usr/lib/nis/nisping -H hostname [domain]
```

To checkpoint a directory:

```
/usr/lib/nis/nisping -C hostname [domain]
```

Displaying the Time of the Last Update

Use the `-u` option. It displays the update times for the master and replicas of the local domain, unless you specify a different domain name.

```
/usr/lib/nis/nisping -u [domain]
```

Here is an example:

```
rootmaster# /usr/lib/nisping -u
Last updates for directory Wiz.Com.:
Master server is rootmaster.Wiz.Com.
        Last update occurred at Wed Nov 25 10:53:37 1992

Replica server is rootreplica1.Wiz.Com.
        Last update seen was Wed Nov 18 11:24:32 1992
```

Pinging Replicas

You can ping all the replicas in a domain, or one in particular. To ping all the replicas, use the command without options:

```
/usr/lib/nis/nisping
```

To ping all the replicas in a domain other than the local domain, append a domain name:

```
/usr/lib/nis/nisping domainname
```

Here is an example that pings all the replicas of the local domain, Wiz.Com.:

```
rootmaster# /usr/lib/nis/nisping
Pinging replicas serving directory Wiz.Com. :
Master server is rootmaster.Wiz.Com.
        Last update occurred at Wed Nov 25 10:53:37 1992

Replica server is rootreplica1.Wiz.Com.
        Last update seen was Wed Nov 18 11:24:32 1992

        Pinging ... rootreplica1.Wiz.Com.
```

Since the update times were different, it proceeds with the ping. If the times had been identical, it would not have sent a ping.

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

To checkpoint a directory, use the `-C` option:

```
/usr/lib/nis/nisping -C directory-name
```

All the servers that support a domain, including the master, transfer their information from their `.log` files to disk. This erases the log files and frees more disk space. While a server is checkpointing, it will still answer requests for service, but it is unavailable for updates.

Here is an example of `nisping` output:

```
rootmaster# /usr/lib/nis/nisping -C
Checkpointing replicas serving directory Wiz.Com. :
Master server is rootmaster.Wiz.Com.
        Last update occurred at Wed Nov 25 10:53:37 1992

Master server is rootmaster.Wiz.Com.
checkpoint has been scheduled with rootmaster.Wiz.Com.
Replica server is rootreplica1.Wiz.Com.
        Last update seen was Wed Nov 18 11:24:32 1992

Replica server is rootreplica1.Wiz.Com.
checkpoint has been scheduled with rootmaster.Wiz.Com.
```

The `nislog` *Command*

The `nislog` command displays the contents of the transaction log.

Syntax

```
/usr/sbin/nislog
/usr/sbin/nislog -h [number]
/usr/sbin/nislog -t [number]
```

Displaying the Contents of the Transaction Log

To display the entire contents of the transaction log, use the `nislog` command without options:

```
/usr/sbin/nislog
```

To display the first (head) or last (tail) entry in the log, use the `-h` or `-t` options:

```
/usr/sbin/nislog -h
/usr/sbin/nislog -t
```

To display the first or last *n* entries, use the `-h` and `-t` options, but specify a *number*:

```
/usr/sbin/nislog -h number
/usr/sbin/nislog -t number
```

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 Wiz.Com. directory was first created. "XID" refers to the transaction ID.

```
rootmaster# /usr/sbin/nislog -h 2
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   : Wiz.Com.
Etry type   : ADD Name
Entry timestamp : Wed Nov 25 10:50:59 1992
Principal       : rootmaster.Wiz.Com.
Object name     : org_dir.Wiz.Com.
...................Object.....................
Object Name   : org_dir
Owner         : rootmaster.Wiz.Com.
Group         : admin.Wiz.Com.
Domain        : Wiz.Com.
Access Rights : r---rmcdr---r---
Time to Live  : 24:0:0
Object Type   : DIRECTORY
Name : 'org_dir.Wiz.Com.'
Type: NIS
Master Server : rootmaster.Wiz.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; failing that, to the entry; failing that, to the 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 9, "Administering NIS+ Access Rights."

Syntax

— *To change the time-to-live value of objects:*

```
nischttl time-to-live object-name
nischttl [-L] time-to-live object-name
```

— *To change the time-to-live value of entries:*

```
nischttl time-to-live [column=value,...],table-name
nischttl [-ALP] time-to-live [column=value,...],table-name
```

```
time-to-live ::=   seconds | days d hours h minutes m seconds s
```

— *Options:*

Table 11-4 `nischttl` Syntax Options

-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 or entry 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, enter 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 or a combination of days, hours, minutes, and seconds. For the former, just enter the number of seconds. For the latter, add the suffixes "`d`, `h`, `m`, and `s`" to the number of days, hours, minutes, and seconds. Here are two pairs of examples that accomplish the same thing:

```
client% nischttl 86400 Sales.Wiz.Com.
client% nischttl 24h Sales.Wiz.Com.

client% nischttl 176461 hosts.org_dir.Sales.Wiz.Com.
client% nischttl 2d1h1m1s hosts.org_dir.Sales.Wiz.Com.
```

The first pair changes the time-to-live of the Sales.Wiz.Com. directory to 86,400 seconds, or 24 hours. The second pair changes the time-to-live of all the entries in a Hosts table to 176,461 seconds, or 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
```

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.Wiz.Com.
client% nischttl 24h [uid=99],passwd.org_dir.Wiz.Com.

client% nischttl 176461 [name=fred],hosts.org_dir.Wiz.Com.
client% nischttl 2d1h1m1s [name=fred],hosts.org_dir.Wiz.Com.
```

Administering NIS+ Tables 12

This chapter describes how to use the NIS+ table administration commands to perform the following tasks:

For a complete description of these commands, their syntax, and options, see the NIS+ man pages.

The `nistbladm` *Command*

The `nistbladm` command is the primary NIS+ table administration command. 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.

Syntax

— *To create or delete a table:*

```
nistbladm -c table-type columnspec... tablename
nistbladm -d tablename

        columnspec ::=  column=[CSI,rights]
```

— *To add, modify, or remove entries:*

```
nistbladm -a
nistbladm -A entry
nistbladm -m new-entry old-entry
nistbladm -r
nistbladm -R entry

     entry ::=   column=value  ...  tablename  |
                 [column=value, ...],tablename
```

The *columnspec* syntax is explained under the task “Creating a Table” on page 191. The *entry* syntax is explained under the task “Adding an Entry to a Table” on page 193.

Creating a 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 table-type columnspec... tablename
```

The *table-type* is simply a string that identifies the table as belonging to a class of tables. It can be any string you choose.

The *columnspec* argument describes how to specify the characteristics of each column. To use the default column characteristics, simply provide the column names followed by equal signs, and separate the columns with spaces:

```
column=  column=
```

To assign the column some non-default characteristics, such as access rights that are different from those of the table as a whole, append each column's characteristics to the equal sign:

```
column=[CSI,rights]  column=[CSI,rights]
```

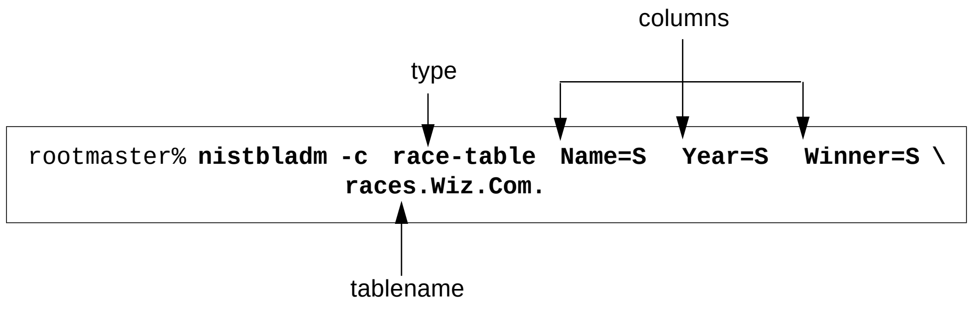
A column can have any of the following characteristics:

Table 12-1 Column Characteristics

S	Searchable. The `nismatch` command can search through the column.
I	Case-insensitive. When `nismatch` searches through the column, it will ignore case.
C	Encrypted.
rights	Access Rights. These access rights are over and above those granted to the table as a whole or to specific entries.

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. The syntax for access rights is described in Chapter 9, "Administering NIS+ Access Rights."

This example creates a table named “Races” in the Wiz.Com. directory (the “org_dir” directory is reserved for system tables). The table has three searchable columns, Name, Year, and Winner. (Within any table there should be no two entries with the same values for all searchable columns.)



```
rootmaster% nistbladm -c  race-table  Name=S   Year=S   Winner=S \
                       races.Wiz.Com.
```

Name	Year	Winner

This is the same example as the above, but access rights are added to each column:

```
rootmaster% nistbladm -c  race-table  Name=S,w+m  Year=S,w+m \
                           Winner=S,w+m   races.Wiz.Com.
```

For more information about specifying column access rights when creating a table, see “Setting Column Rights When Creating a Table” on page 152.

Deleting a Table

To delete a table, simply use the `-d` option and enter the table name:

`nistbladm -d` *tablename*

The table must be empty before you can delete it (see “Removing a Single Entry From a Table” on page 195). This example deletes the Races table from the “Wiz.Com.” directory:

```
rootmaster% nistbladm -d  races.Wiz.Com.
```

Adding an Entry to a Table

You can also use Administration Tool's Database Manager to add, delete, or modify entries to most NIS+ tables.

You can add an entry to a table in two ways:

- With the `-a` option
- With the `-A` option

The `-a` option is recommended for administrators. It adds an entry to a table unless the entry already exists, in which case it returns an error. To use it, you must specify a value for every column in the table:

```
nistbladm -a entry
```

To find the name of a particular column, use the `niscat -o` command. You can use two different syntaxes to specify an entry:

```
entry ::=    column=value ... tablename |
             [column=value,...],tablename
```

The first consists of one or more *column*=*value* pairs, separated by spaces and followed by the table name. The second consists of one or more *column*=*value* pairs, separated by commas and enclosed in square brackets, followed by a comma and the table name. The second syntax is referred to as an *indexed-name.*

These two examples add the same entry to the `races` table, but they each use a different form:

```
rootmaster% nistbladm -a Name=derby  Year=1992  Winner=yoyoma \
                        races.Wiz.Com.
```

```
rootmaster% nistbladm -a [Name=derby,Year=1992,Winner=yoyoma],\
                       races.Wiz.Com.
```

Name	Year	Winner
derby	1992	yoyoma

You can only add one entry with each instance of the `nistbladm` command. This example adds three more entries to the `races` table:

```
rootmaster% nistbladm -a [Name=derby,Year=1991,\
                         Winner=elvis],  races.Wiz.Com.
rootmaster% nistbladm -a [Name=derby,Year=1990,\
                         Winner=LittleFeat], races.Wiz.Com.
rootmaster% nistbladm -a [Name=derby,Year=1989, \
                         Winner=bocefus], races.Wiz.Com.
```

Name	Year	Winner
derby	1992	yoyoma
derby	1991	elvis
derby	1990	LittleFeat
derby	1989	bocefus

The `-A` option is designed for applications. Like the `-a` option, it adds a new entry to a table. However, if the entry already exists, instead of exiting with an error, it changes the operation to "modify," as if the `-m` option had been used instead. Unlike the `-m` option, however, with the `-A` option, you must specify all columns in the entry.

This example demonstrates how `-A` overwrites an existing entry:

Name	Year	Winner
derby	1992	yoyoma
derby	1991	elvis

```
rootmaster% nistbladm -A Name=derby  Year=1992  \
                         Winner=chilipepper  races.Wiz.Com.
```

Name	Year	Winner
derby	1992	chilipepper
derby	1991	elvis

The `-a` option would have returned an error, since the entry specified by `Name=derby Year=1992` already exists.

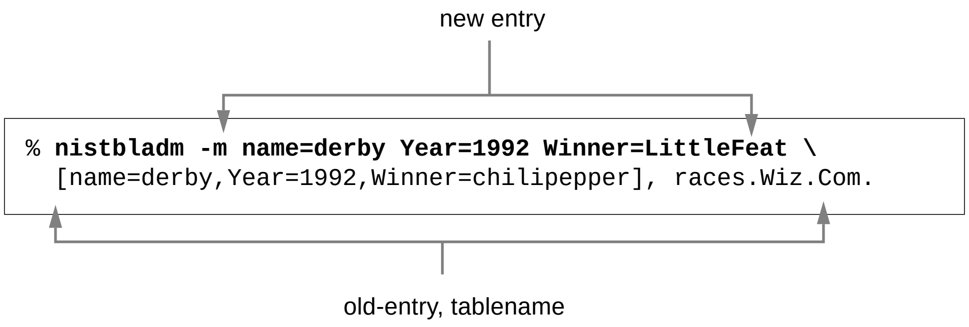
Modifying a Table Entry

You can also use Administration Tool's Database Manager to add, delete, or modify entries to most NIS+ tables.

To modify a table entry, use the `-m` option:

```
nistbladm -m new-entry old-entry
```

Specify the *new-entry* with a set of *column*=*value* pairs. Use an indexed name to specify the *old-entry* and the table name. This example modifies an entry in the `races` table:



```
% nistbladm -m name=derby Year=1992 Winner=LittleFeat \
  [name=derby,Year=1992,Winner=chilipepper], races.Wiz.Com.
```

Name	**Year**	**Winner**
derby	1992	LittleFeat
derby	1991	elvis
derby	1990	LittleFeat
derby	1989	bocefus

Removing a Single Entry From a Table

You can also use Administration Tool's Database Manager to add, delete, or modify entries to most NIS+ tables.

To remove a single entry from a table, use the `-r` option:

```
nistbladm -r indexed-name
```

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. This example removes the Year 1990 entry from the `races` table:

```
rootmaster% nistbladm -r \
        [Name=derby,Year=1990,Winner=LittleFeat],races.Wiz.Com.
```

You could have removed the same entry by specifying only the Year column value, as in this example:

```
rootmaster% nistbladm -r [Year=1990],races.Wiz.Com.
```

However, you could *not* have removed the 1990 entry by specifying only the Winner column value (LittleFeat), because two entries have that same value (1992 and 1990):

Name	Year	Winner
derby	1992	LittleFeat
derby	1991	elvis
derby	1990	LittleFeat
derby	1989	bocefus

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 Winner is LittleFeat:

```
rootmaster% nistbladm -R [Winner=littlefeat],races.Wiz.Com.
```

Name	Year	Winner
derby	1992	LittleFeat
derby	1991	elvis
derby	1990	LittleFeat
derby	1989	bocefus

You can use the `-R` option to remove all the entries from a table. Simply do not specify any column values, as in this example:

```
rootmaster% nistbladm -R [],races.Wiz.Com.
```

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.

Syntax

— *To display the contents of a table:*

`niscat [-hM]` *tablename*

— *To display the object properties of a table:*

`niscat -o` *tablename*
`niscat -o` *entry*

— Options:

Table 12-2 `niscat` Syntax Options

-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 & 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 *tablename*:

`niscat` *tablename*

This example displays the contents of the table named `races`.

```
rootmaster% niscat -h races.Wiz.Com.
#Name:Year:Winner
Derby:1992:LittleFeat
Derby:1991:Elvis
Derby:1990:LittleFeat
Stakes:1992:DaBulls
Stakes:1991:DaBulls
Stakes:1990:LittleFeat
AlmadenOpen:1992:LittleFeat
AlmadenOpen:1991:Buckaroo
AlmadenOpen:1990:Trigger
```

▼ **How to Display 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:

```
rootmaster# niscat -o hosts.org_dir.Wiz.Com.
Object Name   : hosts
Owner         : rootmaster.Wiz.Com.
Group         : admin.Wiz.Com.
Domain        : org_dir.Wiz.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: ----------------
```

```
rootmaster# niscat -o [name=rootmaster],hosts.org_dir.Wiz.Com.
Object Name     : hosts
Owner           : rootmaster.Wiz.Com.
Group           : admin.Wiz.Com.
Domain          : org_dir.Wiz.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 the table below.

Table 12-3 Comparison of `nisgrep` and `nismatch`

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 races.Wiz.Com. Only the first two columns are searchable.

Table 12-4 races.Wiz.Com. Example Table

Name (S)	**Year** (S)	**Winner**
Derby	1992	LittleFeat
Derby	1991	Elvis
Derby	1990	LittleFeat
Stakes	1992	Dabulls
Stakes	1991	Dabulls
Stakes	1990	LittleFeat
AlmadenOpen	1992	LittleFeat
AlmadenOpen	1991	Buckarro
AlmadenOpen	1990	Trigger

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.Wiz.Com.
```

The regular expression symbols are summarized in Table 12-5, below.

Table 12-5 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

Table 12-5 Regular Expression Symbols

Symbol	**Description**
`[`*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
?	Find any value
`\'`*s-char*`'`	Find a special character, such as ? or $.
`x	y`	Find a character that is either x or y

Syntax

— *To search through the first column:*

```
nismatch string tablename
nisgrep reg-exp tablename
```

— *To search through a particular column:*

```
nismatch column=string  tablename
nisgrep column=reg-exp  tablename
```

— *To search through multiple columns:*

```
nismatch column=string ... tablename
nismatch [column=string,...],tablename
nisgrep column=reg-exp ... tablename
```

— *Options:*

Table 12-6 `nismatch` and `nisgrep` Syntax Options

-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 & 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 "races" table for all the entries whose first column has a value of "derby":

```
rootmaster% nismatch  -h  derby  races.Wiz.Com.

--OR--

rootmaster% nisgrep  -h  derby  races.Wiz.Com.
Name**Year**Winner
Derby**1992**LittleFeat
Derby**1991**Elvis
Derby**1990**LittleFeat
```

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 `races` table for all the entries whose third column has a value of “LittleFeat”:

```
rootmaster% nismatch -h  Winner=LittleFeat  races.Wiz.Com.

--OR--

rootmaster% nisgrep  -h  Winner=LittleFeat \
                         races.org_dir.Wiz.Com.
Name**Year**Winner
Derby**1990**LittleFeat
Derby**1992**LittleFeat
Stakes**1990**LittleFeat
AlmadenOpen**1992**LittleFeat
```

Searching Multiple Columns

To search for entries with matches in two or more columns, use the following syntax:

```
nismatch [-h] column=string ... tablename |
nismatch [-h] [column=string,...],tablename

nisgrep  [-h] column=reg-exp ... tablename
```

This example searches for entries whose second column has a value of “1992” and whose third column has a value of “LittleFeat”:

```
rootmaster% nismatch -h [Year=1992,Winner=LittleFeat], \
                 races.Wiz.Com.

--OR--

rootmaster% nisgrep -h Year=1992 Winner=LittleFeat \
                 races.Wiz.Com.
Name**Year**Winner
Derby**1992**LittleFeat
AlmadenOpen**1992**LittleFeat
```

The `nisln` *Command*

The `nisln` command creates symbolic links between NIS+ objects and table entries. You can use it to link objects to objects or objects to table entries. (You cannot create a link that originates with a table entry.) All NIS+ administration commands accept the `-L` flag, which directs them to follow links between NIS+ objects.

To create a link to another object or entry, you must have Modify rights to the source object; that is, the one that will point to the other object or entry.

Syntax

— *To create a link:*

`nisln` *source target*

— *Options*:

Table 12-7 `nisln` Syntax Options

-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 "Overriding Defaults" on page 147.

Creating a Link

To create a link between objects, simply specify both object names, first the *source*, then the *target*. To create links between objects and entries use indexed names.

`nisln` *source-object   target-object*
`nisln [`*column*=*value*`,...],`*tablename   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'll need the `nisaddent` command, described in "The nisaddent

Command" on page 207. Expanding a directory into a domain is part of the process of setting up a domain. For a complete description of the prerequisites and required operations, see Part II.

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.

Syntax

— *To expand a directory into an NIS+ domain:*

```
/usr/lib/nis/nissetup
/usr/lib/nis/nissetup directory-name
```

— *To expand a directory into an NIS-compatible NIS+ domain::*

```
/usr/lib/nis/nissetup -Y
/usr/lib/nis/nissetup -Y directory-name
```

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 Wiz.Com.
org_dir.Wiz.Com. created
groups_dir.Wiz.Com. created
auto_master.org_dir.Wiz.Com. created
auto_home.org_dir.Wiz.Com. created
bootparams.org_dir.Wiz.Com. created
cred.org_dir.Wiz.Com. created
ethers.org_dir.Wiz.Com. created
group.org_dir.Wiz.Com. created
hosts.org_dir.Wiz.Com. created
mail_aliases.org_dir.Wiz.Com. created
sendmailvars.org_dir.Wiz.Com. created
netmasks.org_dir.Wiz.Com. created
netgroup.org_dir.Wiz.Com. created
networks.org_dir.Wiz.Com. created
passwd.org_dir.Wiz.Com. created
protocols.org_dir.Wiz.Com. created
rpc.org_dir.Wiz.Com. created
services.org_dir.Wiz.Com. created
timezone.org_dir.Wiz.Com. created
```

▼ Expand 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.Wiz.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 Chapter 5, “Setting Up NIS+ Tables.” 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, then load the file into the other table. Be sure, though, that the information in the file is formatted properly. Appendix A 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 vastly different from those of the table, the merge option can spare the server a great many operations. Because it only deletes 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 existing information in the tables, you must have Modify rights to the tables.

Syntax

— *To load information from text files:*

```
/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:*

```
/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:*

```
/usr/lib/nis/nisaddent -d [-t tablename] tabletype > filename
```

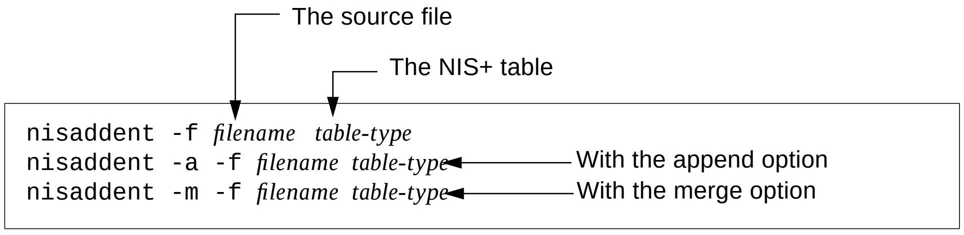
— *Options:*

Table 12-8 `nisaddent` Syntax Options

-a	Append. Contents of the source are appended to contents of the table.
-r	Replace. Contents of the source replace contents of the table.
-m	Merge. Contents of the source are merged with contents of the table.
-d	Dump. Contents of the NIS+ table are dumped to `stdout`.
-v	Verbose. The command prints verbose status messages.
-P	Follow path. If the command was unable to find a table, follow the search paths specified in the environment variable `NIS_PATH`.
-A	All data. Apply the operation to all the tables in the search path.
-M	Master server. Use the tables only in the Master server of the domain.
-D	Override defaults. For the new data being loaded into the tables, override existing defaults. For syntax, see "Overriding Defaults" on page 147.

Loading Information From a File

You can transfer the contents of a file into an NIS+ table in several different ways. One way is to use the `-f` option:



```
nisaddent -f filename table-type
nisaddent -a -f filename table-type
nisaddent -m -f filename table-type
```

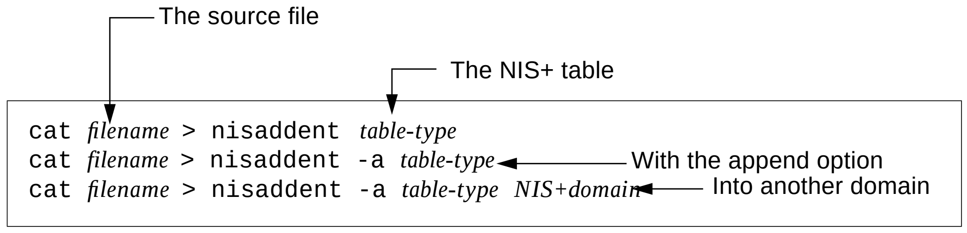
By default, `-f` *replaces* the contents of *table-type* in the local domain with the contents of *filename.* With the `-a` option, it appends the contents of *filename* to *table-type*. With the `-m` option, it merges the contents of *filename* into the contents of *table-type.*

When creating a NIS+ passwd table from `/etc` files, you must run `nisaddent` twice; once on the `/etc/passwd` file and once on the `/etc/shadow` file.

These 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:

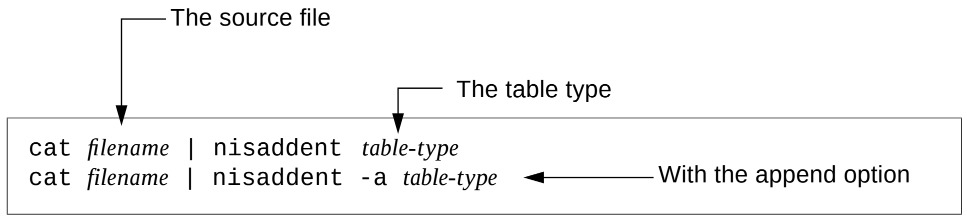
```
rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow

rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd \
            Sales.Wiz.Com.
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow \
            Sales.Wiz.Com.
```

Another way is to use `stdin` as the source. However, you cannot use the `-m` option with `stdin`. Here is an example:



```
cat filename > nisaddent  table-type
cat filename > nisaddent -a  table-type
cat filename > nisaddent -a  table-type  NIS+domain
```

Here is the output of `cat` being piped into `nisaddent`:



```
cat filename | nisaddent  table-type
cat filename | nisaddent -a table-type
```

If the NIS+ table is one of the automounter tables or a nonstandard table, add the `-t` option and the complete name of the NIS+ table. To make it easier to find, it is highlighted in this example:

```
rootmaster# /usr/lib/nis/nisaddent -f /etc/auto_home.xfr \
            -t auto_home.org_dir.Wiz.Com. key-value

rootmaster# /usr/lib/nis/nisaddent -f /etc/auto_home.xfr \
            -t auto_home.org_dir.Sales.Wiz.Com. \
            key-value Sales.Wiz.Com.
```

Loading Data From an NIS Map

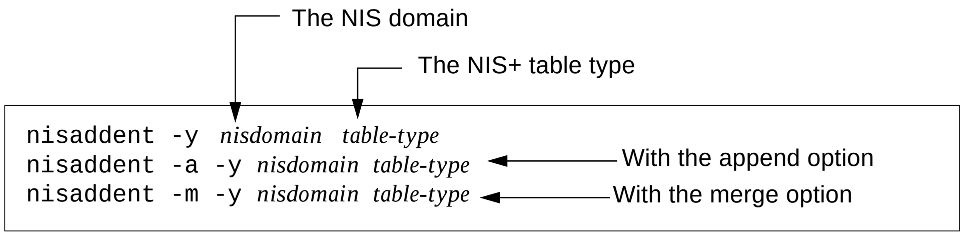
Note that `/var/yp/`*nisdomain* must be local files

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*::

Table 12-9 NIS+ Table Types and Matching NIS Map Names

NIS+ Table Type	NIS Map Name
Hosts	hosts.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.



```
nisaddent -y  nisdomain  table-type
nisaddent -a -y nisdomain  table-type
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 OldWiz domain:

```
rootmaster# /usr/lib/nis/nisaddent -y OldWiz passwd

rootmaster# /usr/lib/nis/nisaddent -y OldWiz passwd \
            Sales.Wiz.Com.
```

The second example does the same thing, but for the “Sales.Wiz.Com.” domain instead of the local domain, “Wiz.Com.”

If the NIS+ table is one of the automounter tables 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. To make the option easier to find, it is highlighted in these examples:

```
rootmaster# /usr/lib/nis/nisaddent -y OldWiz \
            -t auto_home.org_dir.Wiz.Com. key-value

rootmaster# /usr/lib/nis/nisaddent -y OldWiz \
          -t auto_home.org_dir.Wiz.Com. key-value  Sales.Wiz.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. To make the option easier to find, it is highlighted in these examples:

```
rootmaster# /usr/lib/nis/nisaddent -Y hosts.byname hosts

rootmaster# /usr/lib/nis/nisaddent -Y hosts.byname hosts \
            Sales.Wiz.Com.
```

If the NIS map is one of the automounter maps or a nonstandard map, combine the -Y option with the -t option:

```
rootmaster# /usr/lib/nis/nisaddent -Y auto_home \
            -t auto_home.org_dir.Wiz.Com. key-value

rootmaster# /usr/lib/nis/nisaddent -Y auto_home \
          -t auto_home.org_dir.Wiz.Com. key-value Sales.Wiz.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:

```
/usr/lib/nis/nisaddent -d [-t tablename] tabletype > filename
```

Problems and Solutions 13

This chapter describes some of the problems you may encounter while administering an NIS+ namespace. 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 to this chapter, there is an appendix containing an alphabetic listing of the more common NIS+ error messages. If you are responding to a specific error message, check Appendix A, “Error Messages” first. If the problem is simple, or specific to a single error message, its solution is usually described in the error message appendix.

This chapter covers the following types of NIS+ problems:

13

Namespace Administration Problems

This section describes problems that may be encountered in the course of routine namespace administration work.

Symptoms

- `Illegal object type for operation` message.
- Other “object problem” type 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” type error messages

Possible causes

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.
- 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

Check the following:

- That the NIS+ server that you specified with the `-H` option is a valid server and that it is running.
- That you typed the correct DNS domain name with the `-D` option, or set it with `domainname`.
- That `rpc.nisd` is running on the server
- That "`nobody`" has `read` permission for this domain
- That the netmask is properly set up on this machine.

Checkpoint Keeps Failing

If checkpoint operations (for example, 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 with `nisrmdir -s`.

Lack of Disk Space

Lack of sufficient disk space will cause a variety of different error messages. (See "Insufficient Disk Space" on page 238 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 `nisls -l` and `niscat` to check for existence, permissions, and readability. Then check `syslog` 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*`:`*domain name* error messages.

For example, the machine `alladin` might be a client of the `subwiz.wiz.com.` domain. If the master server of the `subwiz.wiz.com.` subdomain is the machine `merlin`, then `merlin` is a client of the `wiz.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 `alladin` machine is `alladin.subwiz.wiz.com.` The fully qualified name of the `merlin` machine is `merlin.wiz.com.` The name `merlin.subwiz.wiz.com.` is wrong and will cause an error because `merlin` is a client of `wiz.com.`, not `subwiz.wiz.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 sub-directories.

Namespace Database Problems

This section covers problems related to the namespace database and tables.

Symptoms

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"

(See also "Ownership and Permission Problems" on page 226.)

Multiple `rpc.nisd` *Parent Processes*

Symptoms

Various Database and transaction log corruption error messages containing the terms:
- "corrupt log"
- "log corrupted"

- “log entry corrupt”
- “corrupt database”
- “database corrupted”

Possible Causes

You have multiple *independent* `rpc.nisd` daemons running. In normal operation, `rpc.nisd` may 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 -ef | grep rpc.nisd`. Make sure that you have no more than one parent `rpc.nisd` process.

Solution

If you started `rpc.nisd` with the `-Y` or `-B` option, you must also kill the `rpc.nisd_resolv` daemon.

If you have more than one ‘parent’ `rpc.nisd` entries, you must kill all but one of them. Use `kill -9` *process-id*, then run the `ps` command again to make sure it has 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.

NIS Compatibility Problems

This section describes problems having to do with NIS compatibility with NIS+ and earlier systems and the Switch configuration file.

Symptoms

The `nsswitch.conf` file fails to perform correctly.

Error messages with operative clauses such as:

- "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 containing phrases 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 2.x 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.

As a general rule, in domains running in NIS-compatibility mode (domains that contain both NIS+ and NIS machines), always use `nispasswd` on an NIS+ machine to change passwords.

Diagnosis

Check to see if the user's old password is still valid on another NIS+ machine.

Solution

Use `nispasswd` 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.

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.

[Object] Not Found Problems

This section describes problem in which NIS+ was unable to find some object or principal.

Symptoms

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 your 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+ will assume it is a partially qualified name.

  An exception to this rule is that 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

Unlike previous systems such as NIS, there is no binding with NIS+.

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/`*hostname* directory has become corrupted or erased. Restore these files from your most recent backup.

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 Problems

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.)

Ownership and Permission Problems

This section describes problems related to user ownership and permissions.

Symptoms

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"

Other common 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, or the owner of the object, or the system administrator, can either change the permission requirements of the object as described in Chapter 9, "Administering NIS+ Access Rights," or add you to a group that does have the required permissions as described in Chapter 10, "Administering NIS+ Groups."

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 229 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 `nispasswd` 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` *name*.

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 `pine` is granted namespace read-only permissions. Then a machine named `pine` is added to the domain. The user `pine` will no longer be able to perform any namespace operations requiring any sort of permission, and the root user of the machine `pine` will only have Read-only permission in the namespace.

Symptoms

- The user or machine gets "permission denied" 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 `niscat` to find the host and user in the hosts and passwd tables and pipe through `grep` to see if there are identical host names and usernames in the respective tables. (For example, `niscat passwd.org_dir | grep` *username*.)

Then run `niscat` on the domain's cred table to see what type of credentials are provided for the duplicate host/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 then 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 229.

Security Problems

This section describes common password, credential, encryption, and other security-related problems.

Symptoms

Error messages with operative clauses such as:
- "authentication error"
- "authentication denied"
- "keyserv fails to encrypt".
- "cannot get public key"
- "chkey failed"
- "insufficient permission to"
- "no public key"
- "permission denied"
- "password [problems]"

User or root unable to perform any namespace operations or tasks. (See also "Ownership and Permission Problems" on page 226.)

Password Entered Incorrectly

The most common security problem is mistyping the password. Try it again, and make sure you correctly enter the password. If this is the first time you are using the machine, contact the network administrator to verify the password.

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

Cause Corrupted keys, 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/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.
- 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 "How to Create Credentials for an NIS+ Principal" on page 127.
- The `keyserv` daemon is not running. Use the `ps` command to see if `keyserv` is running. If it is not running, restart it 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:

`nismatch` *fully-qualified principal name* `cred.org`.*domain_name*

Changed Domain Name

Domain names should not be changed.

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. *Do not change a domain name.*

If you have *already* changed a domain name and are experiencing authentication problems, or error messages containing terms '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+ Password and Login Password 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.) However, if you change your NIS+ password with `nispasswd` that does not affect the password stored in `/etc/passwd`. T o change a password in an `/etc/passwd` file you must run the `passwd` command.

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.

NIS+ Password Different From Login Password

While it is possible for NIS+ clients to have different login and NIS+ passwords, doing so requires that the user to perform a `keylogin` before performing any NIS+ operations. When your login password and NIS+ are different, failure to perform a `keylogin` before performing NIS+ operations will result in typical "permission denied" type error messages.

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 is unable to log in as superuser.

Note that you should not have UserID 0 in the passwd table.

Possible Cause

You changed the root password using `nispasswd` instead of `passwd`. Root passwords must always be changed with `passwd` followed by the `chkey -p` command.

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 `nispasswd` to restore the old password.

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.

Slow Performance and System Hang Problems

This section describes common slow performance and system hang problems.

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.

Checkpointing

Do not reboot! Do not ping!

Someone has issued an `nisping` or `nisping -C` command. Or the `rpc.nisd` daemon is performing a checkpoint operation. 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.

Non-default table paths—Do not use `nistbladm` to set non-default table paths. Non-default 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 of 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 *hostname* 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 it has died, restart it. If `rpc.nisd` frequently dies, contact the Sun Solution Center.

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.

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 hostname information.
- The host name must end in a period.

Solution

If you started `rpc.nisd` with the `-Y` or `-B` option, you must also kill the `rpc.nisd_resolv` daemon.

Kill the NIS+ processes that are hanging and then kill `rpc.nisd` on that host/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.)

System Resource Problems

This section describes problems having to do with lack of system resources such as memory, disk space, and so forth.

Symptoms

Error messages with operative clauses such as:
- “no memory”
- “out of disk space”
- “cannot [do something] with log”
- “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.

User Problems

This section describes NIS+ problems that a typical user might encounter.

Symptoms

Common Symptoms

- User cannot log in
- User cannot rlogin to other domain

User Forgot Password and Cannot Log In

To set up a new password for a user who has forgotten the previous one, run `nispasswd` for that user on another machine (naturally you have to be the NIS+ administrator to do this.)

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 them 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 239), or `nispasswd` was not used to create the new password.

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` 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*`.`*domain_name*. `cred.org_dir | grep` *username* 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.

Other 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.

You can assume that NIS+ is running if:
- `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 Cause s

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
- 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`:.)

These messages indicate that there was some other problem:

- `replica_update: error result was`
- `replica_update: nis dump result nis_perror` *error string*
- `root_replica_update: update failed 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.

Appendices

This part of the manual provides reference material. It has two appendices.

Error Messages A

This section lists alphabetically the more common NIS+ error messages. For each message there is an explanation and, where appropriate, a solution or a cross reference to some other portion of this manual.

The "Problems and Solutions" chapter describes various type of problems and their solutions. Where appropriate, error messages in this appendix are cross-referenced to the corresponding section in the "Problems and Solutions" chapter.

About NIS+ 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 may appear in pop-up windows, shell tool command lines, user console window, the syslog file, or in 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 (table or directory) your command is addressing. However, in some cases an error message may 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.

If you cannot trace the cause of an error message to your command or machine, consider the possibility that the message may have been generated by a server in response to your command or in response to some other NIS+ function.

Similarly, some commands cause a number of different NIS+ objects to be searched or queried. Some of these objects may not be obvious. Any one of these objects could return an error message regarding permissions, read-only state, not-available, and so forth. In such cases the message may or may not be able to inform you of which object the problem occurred in.

In normal operation, the NIS+ software and servers make routine NIS+ 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, some other NIS+ 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 entirely.

Note – When working with an NIS+ namespace you may 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 NIS+ error message may have slightly different meanings depending on which part of the NIS+ software generated the message. For example, when the message `Not found` 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

Some error messages may be preceded by a character string (a name or a number) or by the name of the routine that generated the error message. In some cases a character string may also follow an error message. In this appendix, these variable strings are indicated by an *italic typeface*.

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".
- Many messages contain variable strings such as user IDs, domain names, host names, and so forth. Because variables could be anything, they are not included in the sorting of the messages listed in this appendix. For example, the message *Sales*`: is not a table` would be listed in this appendix as: *name*`: is not a table` and would be alphabetized under the letter 'I' for the first non-variable letter.
- Error messages that begin with asterisks, such as `**ERROR:` *domainname* `does not exist` are generated by the NIS+ installation and setup scripts. They are alphabetized according to their first letter, ignoring the asterisks.

Some of the error messages documented in this chapter are documented more fully in the appropriate `man` pages.

Common NIS+ 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 "Namespace Database Problems" on page 219 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 "Namespace Database Problems" on page 219 for additional information.)

`add_cleanup: Cant allocate more rags.`

This message indicates that your system is running low on available memory. (See "Insufficient Memory" on page 238 for information on insufficient memory problems.)

`add_pingitem: Couldn’t add` *directoryname* `to pinglist (no memory)`

(See “Insufficient Memory” on page 238 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!`

If you started `rpc.nisd` with the `-Y` or `-B` option, you must also kill the `rpc.nisd_resolv` daemon.

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, kill it and restart it with the same options as previously used. If it is not running, restart it 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.

`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 directory, use `nisls -lR` to check for existing files or subdirectories and delete them first. If you are trying to delete a table, use `niscat` to check the contents of the table and `nistbladm` to delete any existing contents.

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 Sun Solution 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 "Keyserv Failure" on page 230 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 "Security Problems" on page 228 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 timestamp-related message, then you need 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 `keyserve` was unable to generate a random DES key for this session. This requires some action on your part:

- Check to make sure that `keyserve` 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 `automound`, `rpc.nisd` and `sendmail`. Then do a keylogin.
- If `keyserve` 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 name service is specified and then make sure that service is responding. If it is not responding, restart it.

`authdes_seccreate: out of memory`

(See "System Resource Problems" on page 237 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 then run `keylogin`.

If restarting `keyserv` fails to correct the problem, it may 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 "Security Problems" on page 228 for additional information regarding security key problems.)

`authdes_validate: DES decryption failure`

See `authdes_marshal: DES decryption failure` on page 248.

`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
- The 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 "Insufficient Memory" on page 238 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 may 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 number*

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-variable*

The system cannot add to the log file. The reason is indicated by the *variable*. The most common cause of this message is lack of disk space. (See "Insufficient Disk Space" on page 238.)

`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 "Namespace Database Problems" on page 219.)

`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* which has been returned by the function. Additional information may 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 find` *name*`'s secret key`

Possible causes:

- You may have incorrectly types the password.

- There may be no entry for *name* in the cred table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt).
- The `nsswitch.conf` file may 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 "Security Problems" on page 228 for information on diagnosing and solving these type of problem.)

`checkpoint_log: Called from read only child ignored.`

This is simply 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 simply 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-variable*`'s secret key`

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for *name* in the `cred` table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt)

- The `nsswitch.conf` file may have the wrong publickey policy. It may 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.

(See “Security Problems” on page 228 for information on diagnosing and solving these type 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-variable`: could not get secret key for '`*string-variable*`'`

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for *name* in the `cred` table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt)
- The `nsswitch.conf` file may have the wrong publickey policy. It may 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.

(See “Security Problems” on page 228 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 “Insufficient Processes” on page 239 for additional information.)

`Couldn't parse access rights for column` *string-variable*

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 “[Object] Not Found Problems” on page 222 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 non-parent 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-variable*

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 “Insufficient Disk Space” on page 238 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 some kind of 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 then 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+ coldstart 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 then 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 the server *NIS+_server* into the `/etc/hosts` 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 non-root 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 may 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` *hostname* `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+ non-root 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 coldstart 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 non-root 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 non-root master with the `nissever` 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 the *client-name* was spelled incorrectly, rerun `nisclient -c` with the correct *client-name*.

- If the *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 messages 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 a NIS+ server:
1. become a 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 that 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 may 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: table` *tablename*`.org_dir.`*domain* `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 may have been caused by the NIS+ server being temporarily busy. Rerun the script to populate this *tablename* table.

`**ERROR: this name “`*client-name*`” is in both the passwd and hosts tables.`
`You cannot have an username same as the hostname.`

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 hostname (-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 may 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 “Domain Name Confusion” on page 218 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 “Illegal Object Problems” on page 216 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 as described in Chapter 9, “Administering NIS+ Access Rights”, or add you to a group that does have the required permissions as described in Chapter 10, “Administering NIS+ Groups”.

See “Ownership and Permission Problems” on page 226 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.)

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.

`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.

`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.

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` *numeric-variable* `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.

`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 "Namespace Database Problems" on page 219.)

`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 that someone changed an existing domain name. Existing domain names should not be changed. (See "Changed Domain Name" on page 230.)

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 "Slow Performance and System Hang Problems" on page 233 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 "Replica Update Failure" on page 241 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 equal (=) 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 may see this if your mechanism is broken.

Other possible causes are:

- Coldstart 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` and `/var/nis/NIS_SHARD_DIR_CACHE`, and then reboot. (If the problem is not in the root directory, you may be able to simply 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 below:

```
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
```

`NisDirCacheEntry::write: xdr_directory_obj failed`

The most likely causes for this message are:

- An attempt to allocate system memory failed. See “Insufficient Memory” on page 238 for a discussion of memory problems.
- If your system does not seem to be short of memory, contact the Sun Solutions Center.

`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-variable`: NIS+ server busy try again later.`

See “Slow Performance and System Hang Problems” on page 233 for possible causes.

`NIS+ server busy try again later.`

Self explanatory. Try the command later.

See also “Slow Performance and System Hang Problems” on page 233 for possible causes.

`NIS+ server for` *string-variable* `not responding still trying`

See “Slow Performance and System Hang Problems” on page 233 for possible causes.

`NIS+ server not responding`

See “Slow Performance and System Hang Problems” on page 233 for possible causes.

`NIS+ server needs to be checkpointed. Use nisping -C` *domainname*

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 “Logs Grow Too Large” on page 217 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 may 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`. (See the `nis_tables and nis_names man` pages for additional information.)

`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 “Insufficient Memory” on page 238 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 simply 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, ignored`

This is simply 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 “Replica Update Failure” on page 241 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: readonly child called to mkdir, ignored`

This is simply 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: readonly child was pung ignored.`

This is simply 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 simply 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 below:

```
# 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 "System Resource Problems" on page 237 for additional information on memory problems.)

`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.

`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.)

`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 may 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 “[Object] Not Found Problems” on page 222 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 “[Object] Not Found Problems” on page 222 for general information on this type of problem.)

`Not master server for this domain`

This message may mean that an attempt was made to directly update the database on a replica server.

This message may 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_SHARD_DIR_CACHE` file should run `ps` to obtain the process ID of the `nis_cachemgr`, kill the `nis_cachemgr` process, remove the `/var/nis/NIS_SHARD_DIR_CACHE` file, and then restart `nis_cachemgr`.

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-variable* `(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.

`Password does not decrypt secret key for` *name*

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for *name* in the `cred` table.
- NIS+ could not decrypt the key (possibly because the entry might be corrupt).
- The `nsswitch.conf` file may 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 "Security Problems" on page 228 for information on diagnosing and solving these type of problem.)

`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 may 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.)

`Permission denied`

Returned when you do not have the permissions required to perform the operation you attempted. (See "Ownership and Permission Problems" on page 226 for additional information.)

This message is generated by the NIS+ error code constant: `NIS_PERMISSION`.

`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 wish 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 may 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 may 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.)

`replica_update: Child process attempting update, aborted`

This is simply 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 the *string*) in carrying out a dump to a replica. See “Replica Update Failure” on page 241 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` *error string*

This message indicates a problem (identified by the *error string*) in carrying out a dump to a replica. See "Replica Update Failure" on page 241 for further information.

`replica_update:` *number* `updates` *number* `errors`

A status message indicating a successful update.

`replica_update: WARNING: last_update(`*directoryname*`) returned 0!`

A NIS+ process could not find the last update timestamp 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 simply 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-variable*`: could not fetch object from master.`

This message indicates a problem in carrying out a dump to a replica. See "Replica Update Failure" on page 241 for further information.

`Security exception on local system. UNABLE TO MAKE REQUEST.`

This message may be displayed if a user has the same login ID as a machine name. (See "User Login Same as Machine Name" on page 227 for additional information.)

`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.)

`Success`

The request was successful.

This message is generated by the NIS+ error code constant: `NIS_SUCCESS`. (See the `nis_tables` `man` page for additional 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 may have to try the command again later.

`_svcauth_des: corrupted window from` *principal-name*

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 may 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 that 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 for` *principal-name*

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 may have to call the Sun Solution Center for assistance. If the problem appears to be related to a DES encryption chip, call the Sun Solution Center.

`svcauth_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.

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 may have to call Sun Solution Center for assistance.

`_svcauth_des: invalid timestamp received from` *principal-name*

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` *principal-name*

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 -ef` to check if the `keyserv` process is running on the `keyserv` host. If it is not, then restart it 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` may 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` *principal-name*

The server cannot get the client's public key. Possible causes are:

- The principal has no public key. Run `niscat` 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 name service specified by a `nsswitch.conf` file is not responding.

`_svcauth_des: replayed credential from` *principal-name*

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 may have to take corrective action as described in the bullets 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` *principal-name*

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 may 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` *principal-name*

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 may 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.)

`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 may indicate one of the following problems:

- Corrupted `/var/nis/NIS_SHARD_DIR_CACHE` file. Kill the cache manager, remove this file, and restart the cache manager. \
- 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 may indicate a corrupted `/var/nis/NIS_COLD_START`, `/var/nis/NIS_SHARD_DIR_CACHE`, 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-variable*`'`

(See “[Object] Not Found Problems” on page 222 for information on this type of problem.) This particular message may 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-variable`: Unable to decrypt secret key for` *string-variable*`.`

Possible causes:

- You may have incorrectly typed the password.
- There may be no entry for *name* in the `cred` table.
- NIS+ could not decrypt the key because the entry might be corrupt.
- The `nsswitch.conf` file may 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 “Security Problems” on page 228 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` and `nis_names` man pages for additional information.)

`update_directory:` *number* `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.

`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 "System Resource Problems" on page 237).
- 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 non-root 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. They 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-variable*`) do not have secure RPC credentials in NIS+ domain '`*string-variable*`'`

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 "Ownership and Permission Problems" on page 226 for additional information on credential, ownership, and permission problems.

`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 may not have the directory object. Run `nisping -C` to checkpoint.
- You are out of disk space. (See “Insufficient Disk Space” on page 238.)
- Database corruption,
- Some other type of software error. Contact the Sun Solution Center.

Information in NIS+ Tables B

This appendix summarizes the information stored in the following NIS+ tables:

The Cred table, because it contains only information related to NIS+ security, is described in Chapter 7, “Administering NIS+ Security.”

As explained in Chapter 1, without the name service, this information would be stored in /etc files. In fact, most NIS+ tables have corresponding /etc files. With the NIS service, you could combine the information in the NIS maps with the information in their corresponding /etc maps by using the +/- syntax. However, the Name Service Switch provides a better method.

The Name Service Switch allows you to specify one or more sources for different types of information. In addition to NIS+ tables, that source can be NIS maps, DNS maps, or /etc tables. The order in which you specify them determines how the information from different sources is combined.

If you are creating input files for any of these tables, most tables share two formatting requirements: you must use one line per entry, and 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 a heading called "*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, and happens to be, by default, /home. The second part of the mount point (i.e., the sub directory 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 13-1 Auto_Home Table

Column	Description
Mount Point	The login name of every user in the domain.
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 simply 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 13-2 Auto_Master Table

Column	**Description**
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 `/—`.
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 pathname, 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, Solaris 2.X 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. For more information about the automounter, consult books about the Automounter or books that describe the NFS filesystem.

Bootparams Table

The bootparams table stores configuration information about every diskless workstation in a domain. A diskless workstation is a workstation that is connected to a network, but has no hard disk. Since it has no internal storage capacity, a diskless workstation stores its files and programs in the filesystem of a server on the network. It also stores its configuration information — or *boot parameters* — on a server.

Because of this arrangement, every diskless workstation 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 workstations. 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 workstation's root, swap, and dump partitions.

The default bootparams table has only two columns but uses them to provide the following four items of information:

Table 13-3 Bootparams Table

Column	Description
Hostname	The diskless workstation's official hostname, as specified in the hosts table
Configuration	Root Partition: the location (server name and path) of the workstation's root partition
	Swap Partition: the location (server name and path) of the workstation's swap partition
	Dump Partition: the location (server name and path) of the workstation's dump partition

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
```

Here is an example:

```
buckaroo      root=bigriver:/export/root1/buckaroo\
              swap=bigriver:/export/swap1/buckaroo\
              dump=bigriver:/export/dump/buckaroo
```

Ethers Table

The ethers table stores information about the 48-bit Ethernet addresses of workstations on the Internet. It has two columns:

Table 13-4 Ethers Table

Column	**Description**
Ethernet-address	The 48-bit Ethernet address of the workstation.
Official-host-name	The name of the workstation, as specified in the hosts table.

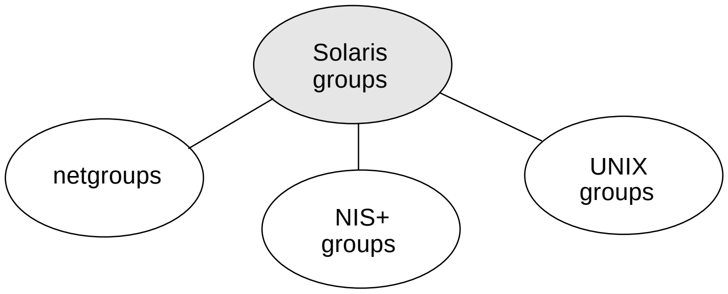
An Ethernet address has the form:

```
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.

Group Table

The group table stores information about workstation user groups. Solaris 2.X supports three kinds of groups: netgroups, NIS+ groups, and UNIX groups.



A netgroup is a group of workstations and users that have permission to perform remote operations on other workstations in the group. An NIS+ group is a set of NIS+ users that can be assigned access rights to an NIS+ object. They are described in Chapter 7, “Administering NIS+ Security.” A UNIX group is simply a collection of users who are given additional UNIX access permissions.

UNIX groups allow a set of users on the network to access a set of files on several workstations or servers without making those files available to everyone. For example, the engineering and marketing staff working on a particular project could form a workstation user group.

The group table has four columns:

Table 13-5 Group Table

Field	**Description**
Name	The group’s name
Password	The group’s password.
GID	The group’s numerical ID.
members	The names of the group members, separated by commas.

Previous releases of SunOS used a `+/-` syntax in local `/etc/group` files to incorporate or overwrite entries in the NIS group maps. Since Solaris 2.X uses the Name Service Switch to specify a workstation’s sources of information, this is no longer necessary. All you have to do in Solaris 2.X 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 workstations in a domain with their IP addresses. The workstations 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 workstations. The hosts table has four columns:

Table 13-6 Hosts Table

Column	Description
IP Address	The workstation’s IP address (network number plus workstation ID number)
Hostname	The workstation’s official name
Nickname	An optional name used in place of the hostname to identify the workstation
Comment	An optional comment about the record

Mail Aliases Table

The mail aliases table lists the domain’s mail aliases recognized by `sendmail`. It has two columns:

Table 13-7 Mail Aliases Table

Column	Description
Alias Name	The name of the alias
Members	A list containing the members that receive mail sent to this alias. Members can be users, workstations, or other aliases.

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 netgroups used for remote mounts are workstations; for remote logins and shells, they are users.

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

The netgroup table has two columns:

Table 13-8 Netgroup Table

Column	Description
Group Name	The name of the network group.
List of Members	A list of the members in the group.

Input File Format

The input file consists of a group name and any number of members:

groupname member-specification...

A member specification can be the name of another netgroup or an ordered list with three fields:

```
member-spec ::= group-name |
                ( hostname, username, domainname )
```

The first field specifies the name of a workstation. The second field specifies the name of a user. The third field specifies the domain in which the member specification is valid.

A missing field indicates a wildcard. For example, this netgroup includes all workstations and users in all domains:

```
everybody (,,)
```

A dash in a field is the opposite of a wildcard; it indicates that no workstations or users belong to the group. Here are two examples:

```
(host1, -,Wiz.Com.)
(-,joe,Wiz.Com.)
```

The first specification includes one workstation, `host1`, in the Wiz.Com. domain, but excludes all users. The second specification includes one user in the Wiz.Com. domain, but excludes all workstations.

Netmasks Table

The netmasks table contains the network masks used to implement standard Internet subnetting. The table has two columns:

Table 13-9 Netmasks Table

Column	**Description**
Network Number	The IP number of the network.
Subnet Mask	The network mask to use on the network.

For network numbers, you can use the conventional IP dot notation used by workstation addresses, but leave zeroes in place of the workstation addresses. For example, this entry

```
128.32.0.0          255.255.255.0
```

means that class B network 128.32.0.0 should have 16 bits in its network field, eight bits in its subnet field, and eight 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 may need to add your local networks to it. It has three columns:

Table 13-10 Networks Table

Column	Description
Network Name	The official name of the network, supplied by the Internet
Network Number	The official IP number of the network
Aliases	An unofficial name for the network

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).

The information in the passwd table is added when users' accounts are created. The passwd table contains the following columns:

Table 13-11 Passwd Table

Column	Description
User 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.
Password	The user's encrypted password.
UID	The user's numerical ID, assigned when the user's account is created.
Group ID	The numerical ID of the user's group.

Table 13-11 Passwd Table

Column	Description
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 Directory	The pathname of the user's home directory; that is, the directory the user is placed in after logging in.
Login Shell	The user's initial shell program. The default is the C-shell: `/usr/bin/csh`.

The passwd table has an additional column: the Shadow column. It stores restricted information about user accounts. It includes the following information:

Table 13-12 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.
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.

Previous releases of SunOS used a `+/-` syntax in local `/etc/passwd` files to incorporate or overwrite entries in the NIS password maps. Since Solaris 2.X uses the Name Service Switch to specify a workstation's sources of information, this is no longer necessary. All you have to do in Solaris 2.X 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 specify "compat" for the `passwd` source.

Protocols Table

The protocols table lists the protocols used by the Internet. It has four columns:

Table 13-13 Protocols Table

Column	**Description**
Protocol Number	The name of the protocol
Protocol Name	The protocol number
Aliases	An unofficial alias used to identify the protocol
Comments	Comments about the protocol

Here is an example of an input file for the protocols table:

```
#
# Internet (IP) Protocols
#
ip    0      IP     # internet protocol, pseudo protocol number
icmp  1      ICMP   # internet control message protocol
ggp   3      GGP    # gateway-gateway protocol
tcp   6      TCP    # transmission control protocol
pup   12     PUP    # PARC universal packet
udp   17     UDP    # user datagram protocol
#
```

RPC Table

The RPC table lists the names of RPC programs. It has four columns:

Table 13-14 RPC Table

Column	**Description**
RPC program name	The name of the program

Table 13-14 RPC Table

Column	Description
RPC program number	The program number
Aliases	Other names that can be used to invoke the program
Comments	Comments about the RPC program

Here is an example of an input file for the RPC table:

```
#
# rpc file
#
rpcbind    100000    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
#
```

Services Table

The services table stores information about the Internet services available on the Internet. It has four columns:

Table 13-15 Services Table

Column	Description
Service Name	The official Internet name of the service.
Port/Protocol	The port number and protocol through which the service is provided (for instance, 512/tcp)
Aliases	The list of alternate names by which the service can be requested.
Comments	Comments about the service.

Timezone Table

The timezone table, lists the default timezone of every workstation in the domain. The default timezone is used during installation, but can be overridden by the installer. The table has three columns:

Table 13-16 Timezone Table

Field	**Description**
Timezone name	The name of the timezone (e.g., US/Pacific)
Workstation or Domain Name	The name of the workstation or, if using only one line in the entire table, the name of the domain
Comments	Comments about the timezone

Index