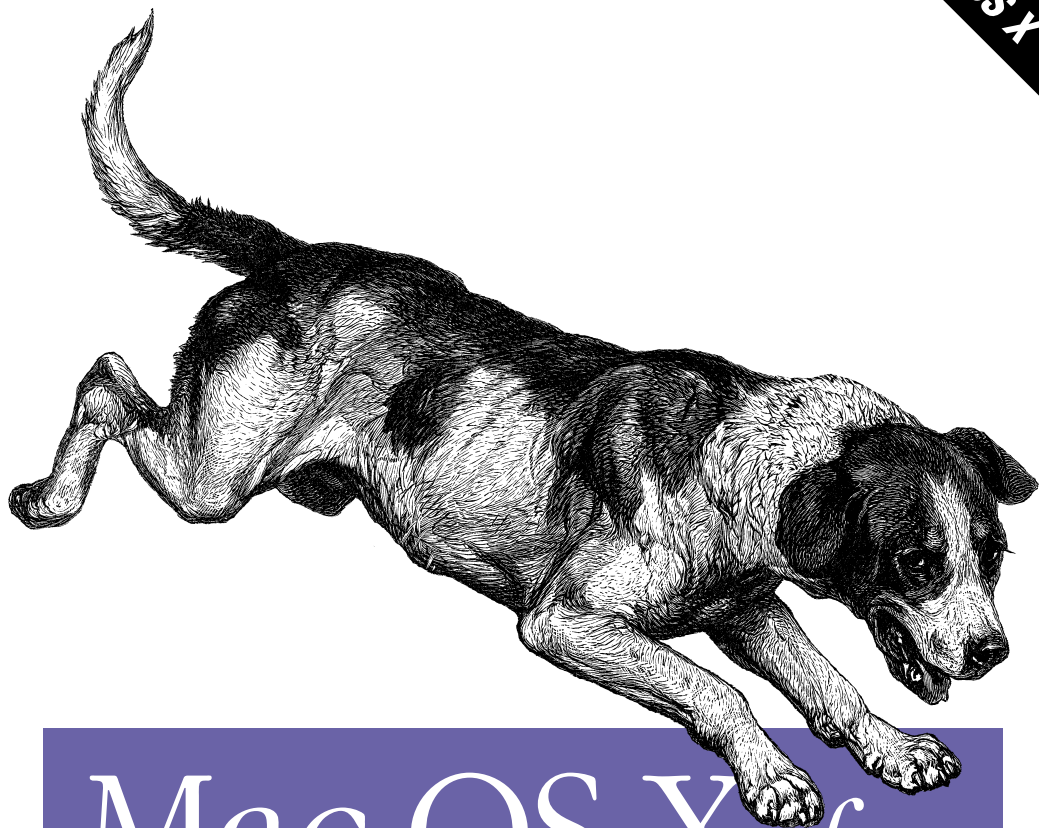




Developer
Connection

Recommended Title

Switching
to Mac OS X



Mac OS X *for* Unix Geeks

O'REILLY®

Brian Jepson & Ernest E. Rothman

Mac OS X for Unix Geeks

Brian Jepson and Ernest E. Rothman

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Directory Services

A *directory service* manages information about users and resources, such as printers and servers. It can manage this information for anything from a single machine to an entire corporate network. The Directory Service architecture in Mac OS X is called Open Directory. Open Directory encompasses flat files (such as */etc/hosts*), NetInfo (the legacy directory service brought over from earlier versions of Mac OS X and NeXTSTEP), LDAPv3, and other services through third-party plug-ins.

This chapter describes how to perform common configuration tasks, such as adding a user or host on Mac OS X with the default configuration. If your system administrator has configured your Macintosh to consult an external directory server, some of these instructions may not work. If that's the case, you should ask your system administrator to make these kinds of changes anyhow!

Understanding Directory Services

In Mac OS X 10.1.x and earlier, the system was configured to consult the NetInfo database for all directory information. If you needed to do something simple, such as adding a host, you couldn't just add it to */etc/hosts* and be done with it. Instead, you had to use the NetInfo Manager (or NetInfo's command-line utilities) to add the host.

However, in Mac OS X 10.2 (Jaguar), NetInfo functions more as a legacy protocol. Instead of being a major player in the directory services world, NetInfo's role has been reduced to that of the local directory database for machines that are not participating in a network-wide directory, such as Active Directory or OpenLDAP. NetInfo is still present on Mac OS X systems, but you can perform most configuration tasks by editing the standard

Unix flat files. By default, Mac OS X 10.2 is configured to consult the local directory (also known as the NetInfo database) for authentication, which corresponds to */etc/passwd** and */etc/group* on other Unix systems. You can override this setting with the Directory Access application. For more information, see “Configuring Directory Services,” later in this chapter.

For users whose network configuration consists of an IP address, a default gateway, and some DNS addresses, this default configuration should be fine. You’ll need to tap into Open Directory’s features for more advanced configurations, such as determining how a user can log into a workstation and find their home directory, even when that directory is hosted on a shared server.

In order to work with Mac OS X’s Directory Services, you must first understand the overall architecture, which is known as Open Directory. Directory Services is the part of Mac OS X (and the open source Darwin operating system) that implements this architecture. Figure 3-1 shows the relationship of Directory Services to the rest of the operating system. On the top, server processes, as well as the user’s desktop and applications, act as clients to Directory Services, which delegates requests to a directory service plug-in (see the “Configuring Directory Services” section, later in this chapter, for a description of each plug-in).

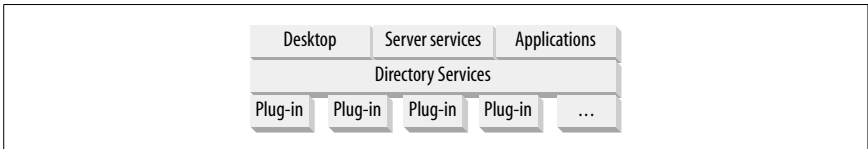


Figure 3-1. The Directory Services architecture

Programming with Directory Services

As a programmer, you frequently need to deal with directory information, whether you realize it or not. Your application uses Directory Services each time it looks up a host entry, authenticates a password, or uses a printer. The Open Directory architecture unifies what used to be a random collection of flat files in */etc*. The good news is that the flat files still work. The other good news is that there is a brave new world just beyond those flat files. So, while all your old Unix code should work with the Open Directory architecture, you should look for new ways to accomplish old tasks, especially if you can continue writing portable code.

* */etc/master.passwd* is the shadow password file that actually contains the encrypted passwords.

To get at directory information, Unix applications typically go through the C library using such functions as `gethostent()`. Higher-level APIs, such as Pluggable Authentication Modules (PAM) and Common Data Security Architecture (CDSA), also use the C library. Figure 3-2 shows how this works. The C library connects to *lookupd*, a thin shim that is the doorway to the *DirectoryService* daemon. The *DirectoryService* daemon consults the available plug-ins until it finds the one that can answer the directory query.

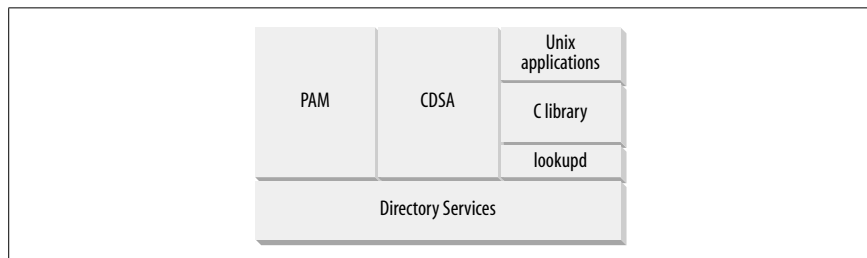


Figure 3-2. Accessing Directory Services

Working with Passwords

One possible route to user and password information is through the `getpw*` family of functions. However, those functions are not ideal for working with systems like Mac OS X that support multiple directories (flat files, NetInfo, LDAP, etc.). In particular, `getpwnam()` is not guaranteed to return a crypted password if the system has been configured to use another scheme, such as MD5 passwords. You should use the PAM API instead. PAM is included with, or available for, many flavors of Unix, so you can use it to write portable code. For more information on PAM, see the *pam(8)* manpage.

Configuring Directory Services

In order to configure Directory Services, use the Directory Access application (*/Application/Utilities*), shown in Figure 3-3. You can enable or disable various directory service plug-ins, or change their configuration.

Directory Access supports the following plug-ins:

AppleTalk

This is the ultimate Mac OS legacy protocol. AppleTalk was the original networking protocol supported by Mac OS versions prior to Mac OS X. Linux and the server editions of Windows also support AppleTalk.

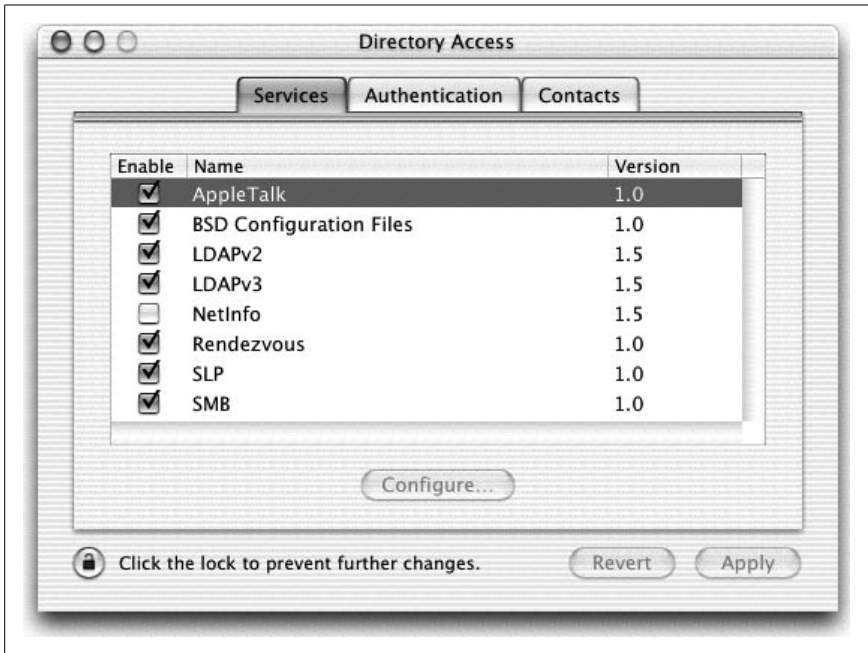
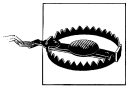


Figure 3-3. The Directory Access application shows the available plug-ins

BSD configuration files

These are flat files located in the */etc* directory, such as *hosts*, *exports*, and *services*.



By default, the checkboxes for NetInfo and BSD Configuration Files are off. For the BSD Configuration Files, the checkbox controls whether the files are consulted for Directory Service lookups. NetInfo is a little more complicated. If the checkbox is off, NetInfo uses the local domain but does not consult network-based NetInfo domains. If the checkbox is on, NetInfo will also look for and potentially use any network-based domains that it finds.

LDAPv2

This is a version of LDAP that Mac OS X can access (read-only).

LDAPv3

This is a newer version of LDAP, which Mac OS X fully supports (read-write). This is the same version of LDAP used by Microsoft's Active Directory and Novell's NDS. Mac OS X Server includes both the client

and server components of OpenLDAP (<http://www.openldap.org>), an Open Source LDAPv3 implementation. The client version of Jaguar includes only the OpenLDAP client components.

NetInfo

This is a legacy Directory Services protocol introduced in NeXTSTEP.



NetInfo and LDAP both use the same data store, which is contained in `/var/db/netinfo/`. The data store is a collection of embedded database files.

Rendezvous

This is Apple's zero-configuration protocol for discovering file sharing, printers, and other network services. It uses a peer-to-peer approach to announce and discover services automatically as devices join a network.

SLP

This is the Service Location Protocol, which supports file and print services over IP.

SMB

This is the Server Message Block protocol, which is Microsoft's protocol for file and print services.

Under the Services tab, everything except NetInfo and BSD Configuration Files is enabled by default. However, if you go to the Authentication tab (Figure 3-4), you'll see that NetInfo is the sole service in charge of authentication (which is handled by `/etc/passwd` and `/etc/group` on other Unix systems).

By default, the Authentication tab is set to Automatic. You can set the Search popup to any of the following:

Automatic

This is the default, which searches (in order): the local NetInfo directory, a shared NetInfo domain, and a shared LDAPv3 domain.

Local directory

This searches only the local NetInfo directory.

Custom path

This allows you to use BSD flat files (`/etc/passwd` and `/etc/group`). After you select Custom path from the pop up, click Add and select `/BSD Configuration Files/Local`.



Figure 3-4. The Directory Access Authentication tab

After you have changed the Search setting, click Apply. The Contact tab is set up identically to the Authentication tab and is used by programs that search Directory Services for contact information (office locations, phone numbers, full names, etc.).



Note that enabling BSD flat files does not copy or change the information in the local directory (the NetInfo database). If you want to rely only on flat files, you would need to remove all the entries from the local directory and add them to */etc/master.passwd*. This would mean you could no longer use the GUI tools to manage those accounts.

NetInfo

The NetInfo directory is organized hierarchically, starting from the *root*, which, like a filesystem's *root*, is called */*. However, this is not meant to suggest that there is a corresponding directory or file for each NetInfo entry. Instead, the NetInfo data is stored in a collection of files under */var/db/netinfo*.

You can browse or modify the NetInfo database using NetInfo Manager, which is located in */Applications/Utilities*. Figure 3-5 shows NetInfo Manager displaying the properties of the *mysql* user.

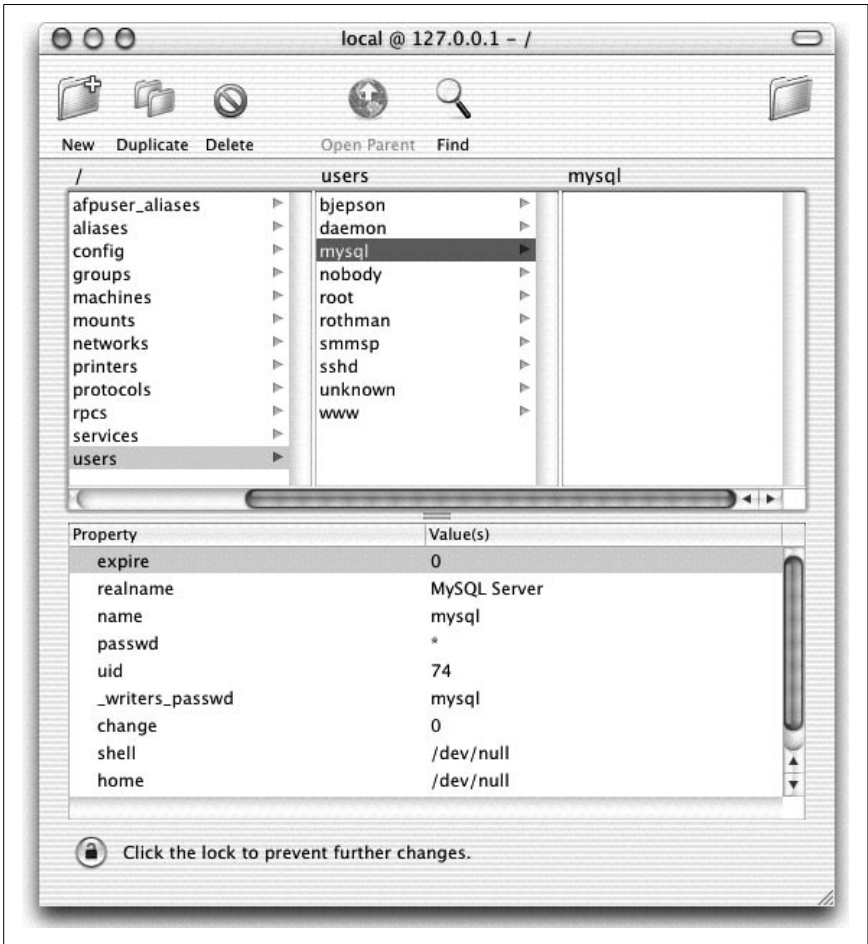


Figure 3-5. Browsing NetInfo

NetInfo Utilities

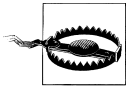
This chapter demonstrates four NetInfo utilities: *nipl*, *nireport*, *nidump*, and *niload*. Table 3-1 describes these and other NetInfo utilities.

Table 3-1. NetInfo tools

Tool	Description
<i>nicl</i>	Provides a command-line interface to NetInfo.
<i>nidump</i>	Extracts flat file format data (such as <i>/etc/passwd</i>) from NetInfo.
<i>nifind</i>	Finds a NetInfo directory.
<i>nigrep</i>	Performs a regular expression search on NetInfo.
<i>niload</i>	Loads flat file format data (such as <i>/etc/passwd</i>) into NetInfo.
<i>nireport</i>	Prints tables from NetInfo.

The *nidump* and *nireport* utilities display the contents of the NetInfo database. *niload* loads the contents of flat files—such as */etc/passwd* or */etc/hosts*—into NetInfo. *niutil* directly manipulates the NetInfo database; it's the command-line equivalent of NetInfo Manager. To modify the NetInfo database, use *sudo* with these commands or first log in as the *root* user. NetInfo commands that can be performed as a normal user are shown with the % prompt. If you need superuser privileges, the # prompt is shown. (Because the user can modify the shell prompt, be careful using this as a visual cue on a real system.)

Unlike other *ni** utilities, *nicl* acts directly on the database files. Consequently, you can use *nicl* to modify the local directory even when Directory Services is not running (such as when you boot into single-user mode).



When you use *niload*, *nicl*, or *niutil*, you are making potentially dangerous changes to your system. But even if you trash the NetInfo database with reckless usage of *niutil* and *niload*, you can restore the NetInfo database from your last backup. For more details, see the “Restoring the NetInfo Database” section, later in this chapter. To back up the local NetInfo database, use the command:

```
nidump -r / -t localhost/local > backup.nidump
```

NetInfo and Mac OS X Server

Mac OS X Server includes a graphical utility, Server Settings, shown in Figure 3-6, which handles the tasks described in this chapter.

Unfortunately, Mac OS X (the client version) does not include this utility, but the instructions in this chapter will help you accomplish the same things. You can also use the instructions in this chapter if you are connecting remotely through SSH to a Mac OS X Server.



Figure 3-6. Mac OS X Server's Server Settings utility

NetInfo Command Reference

This section provides an overview of the NetInfo command-line utilities used in this chapter. The following sections, “Managing Groups” and “Managing Users and Passwords,” demonstrate how to use *niql*, *nireport*, *nidump*, and *niload*. For more details on these or other NetInfo utilities, see their respective manpages.

nidump

Syntax

```
nidump [-T timeout] (-r directory|format) [-t] domain
```

Description

You can dump NetInfo information in a flat file format (such as the */etc/hosts* format) or in a raw format that uses a C-like syntax:

```
{  
  "name" = ( "localhost" );  
  "ip_address" = ( "127.0.0.1" );  
  "serves" = ( "./local" );  
}
```

Options

-T *timeout*

Specifies a timeout in seconds.

-t

Treats the domain as a tagged domain, which includes a machine name and a tagged NetInfo database. For example, *abbot/local* refers to the local NetInfo domain of the machine named *abbot*.

-r *directory*

Dumps the directory in raw format. Directory should be a path to a NetInfo directory, such as */users/root* or */machines*.

format

Specifies a format corresponding to a Unix flat file of the same name. Can be one of: *aliases*, *bootptab*, *bootparams*, *ethers*, *exports*, *fstab*, *group*, *hosts*, *networks*, *passwd*, *printcap*, *protocols*, *resolv.conf*, *rpc*, *services*, or *mountmaps*.

domain

Specifies a NetInfo domain. For standalone machines, use a dot (*.*), which refers to the local domain.

nireport

Syntax

```
nireport [-T timeout] [-t] domain directory [property ...]
```

Description

You can list all NetInfo groups by using the *nireport* utility. To use *nireport*, use the following syntax:

Options

-T *timeout*

Specifies a timeout in seconds.

-t

Treats the domain as a tagged domain, which includes a machine name and a tagged NetInfo database.

domain

Specifies a NetInfo domain.

directory

Denotes a path to a NetInfo directory.

property ...

Specifies one or more NetInfo properties. For example, each user listed in the */users* directory has *name*, *passwd*, *uid*, and *gid* properties (as well as a few other properties). Every directory has a *name* property that corresponds to the directory name. For example, the */machines* directory's *name* property is *machines*.

You can use *nireport* to list any portion of the NetInfo directory. For example, to list the top-level directory, specify the local domain, the */* directory, and the *name* property, as in `nireport . / name`.

niload

Syntax

```
niload [-v] [-T timeout] [(-d|-m)] [(-p|-P password)]  
        [-u user] {-r directory|format} [-t] domain
```

Description

niload reads the Unix flat file format from standard input and loads it into the NetInfo database.

Options

-v

Selects verbose mode.

-T *timeout*

Specifies a timeout in seconds.

-d

Specifies that if a duplicate entry already exists, NetInfo deletes that entry before adding the new one. This can cause you to lose data if NetInfo is tracking information that isn't represented in the flat file. For example, if you dump the */users* directory to a flat *passwd* file format and load it back in with *niload -d*, you will lose the *picture*, *hint*, and *sharedDir* properties for every user on your system, because the *passwd* file does not have a field for those properties. Most of the time, the *-m* option is what you want.

-m

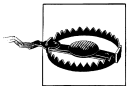
Specifies that if a duplicate entry already exists, *niload* will merge the changes. So, if you dump the */users* directory to a flat *passwd* file format, change a user's shell, and load that file back in with *niload*, NetInfo will keep the old shell. If you use the *-m* option, NetInfo will accept the new shell without the destructive side effects of the *-d* option.

-p

Prompt for a password. You can use this instead of prefixing the command with *sudo*.

-P *password*

Use the specified password.



If your shell history file is enabled, the *-P* option presents a security risk, since the password will be stored, along with the history of other shell commands. It is best to avoid using this option.

-u *user*

Use the specified user's identity when running the command. You'll be prompted for a password.

-t

Treats the domain as a tagged domain, which includes a machine name and a tagged NetInfo database.

domain

Specifies a NetInfo domain.

directory

Denotes a path to a NetInfo directory.

format

Specifies a format corresponding to a Unix flat file of the same name. Can be one of the following: *aliases*, *bootptab*, *bootparams*, *exports*, *fstab*, *group*, *hosts*, *networks*, *passwd*, *printcap*, *protocols*, *rpc*, or *services*.

nicl

Syntax

```
nicl [options] datasource [command]
```

Description

Use *nicl* to modify entries in the NetInfo database. You can manipulate directories and properties with *nicl*. The *datasource* may be the path to a NetInfo directory (such as */*) or the filesystem path of a NetInfo database (you must use the *-raw* option for this). Use *-raw* to work directly with the NetInfo database, such as */var/db/netinfo/local.nidb*. This is useful in cases when the NetInfo daemon is down (such as when you boot into single-user mode).

Options

-c

Create a new data source.

-p

Prompt for a password. You can use this instead of prefixing the command with *sudo*.

-P password

Use the specified password.

-q

Be quiet.

-raw

Indicates that the *datasource* is a filesystem path to a NetInfo database.

-ro

Open *datasource* as read-only.

-t

Treats the domain as a tagged domain, which includes a machine name and a tagged NetInfo database.

-u user

Use the specified user's identity when running the command. You'll be prompted for a password.

-v

Be verbose.

-x500

Use X.500 names (see the *niel* manpage for more details).

Commands

-append path key val ...

Appends a value to an existing property. The property is created if it does not already exist.

-copy path newparent

Copies the specified *path* to a new parent path.

-create path [key [val ...]]

Creates a NetInfo directory specified by *path*. See the "Creating a User with *niel*" section, later in this chapter, for a complete example.

-delete path [key [val ...]]

Destroys the specified path and all its contents. If you specify a key and/or value, only the specified key is deleted. For an example, see the "Deleting a Group" section, later in this chapter.

-domainname

Prints the NetInfo domain name of *datasource*.

-flush

Flushes the directory cache.

-insert path key val index

Operates like *-append*, but instead of placing the value at the end, it inserts it at the specified index.

-list path [key ...]

Lists all the NetInfo directories in the specified path. For example, to list all users, use *niel / -list /users*.

-merge path key val ...

Operates like *-append*, but if the value already exists, it is not duplicated. See the "Adding Users to a Group" section, later in this chapter.

-move path newparent

Moves the specified *path* to a new parent path.

-read path [key ...]

Displays all the properties of the specified path. For example, to see *root*'s properties, use *niel / -read /users/root*.

-search arguments

Performs a search within the NetInfo database. For complete details, see the *niel* manpage.

- rename *path oldkey newkey*
Renames a property.
- resync
Resynchronizes NetInfo.
- rparent
Prints the NetInfo parent of *datasource*.
- statistics
Displays NetInfo server statistics.

Managing Groups

NetInfo stores information about groups in its */groups* directory. This is different from the */etc/group* file, which is consulted only in single-user mode.

To list all of the group IDs (GIDs) and group names for the local domain, invoke *nireport* with the NetInfo domain (., the local domain), the directory (*/groups*), and the properties you want to inspect—in this case, *gid* and *name*:

```
% nireport . /groups gid name
-2      nobody
-1      nogroup
0       wheel
1       daemon
2       kmem
3       sys
4       tty
5       operator
6       mail
7       bin
20      staff
25      smmsp
31      guest
45      utmp
66      uucp
68      dialer
69      network
70      www
74      mysql
75      sshd
80      admin
99      unknown
```



Although the flat file format is called *group* (after the */etc/group* file), the NetInfo group directory is */groups*. If you forget that last *s*, NetInfo will look for the wrong directory.

Creating a Group with niload

The *niload* utility can be used to read the flat file format used by */etc/group* (name:password:gid:members). To add a new group, you can create a file that adheres to that format, and load it with *niload*. For ad hoc work, you can use a here document rather than a separate file:

```
# niload group . <<EOF
? writers*:1001:
? EOF
```

Here Documents

A here document is a shell quoting syntax that allows you to send data to standard input as though it had come in from a file. You can use this syntax interactively from the command line or in a shell script. The EOF tag, shown in the previous example, can be any text string. The here document starts with <<STRING and ends when STRING appears on a line by itself. For example, you can sort a bunch of words with here documents. (The ? character is supplied by the shell to let you know it is expecting input.)

```
% sort <<WORDS
? gamma
? beta
? alpha
? omega
? WORDS
alpha
beta
gamma
omega
```

Creating a Group with nidl

To create a group with *nidl*, you'll need to create a directory under */groups* and set the *gid* and *passwd* properties. If you want a password, it must be encrypted with *crypt()*. If you don't want a group password, use an asterisk instead. (Be sure to quote the * so that the shell does not attempt to expand it.) The following creates a group named *writers* as GID 5005 with no password and no members:

```
# nidl / create /groups/writers gid 5005
# nidl / create /groups/writers passwd '*'
```

Adding Users to a Group

You can add users to the group by appending values to the *users* property with *nicl*'s *-merge* switch at the command line (or by using the *merge* command interactively). If the *users* property does not exist, *nicl* creates it. If the users are already part of the group, they are not added to the list (contrast this with the *-append* command, which can result in the same user being added more than once if the command is invoked multiple times).

```
# nicl / -merge /groups/writers users bjepson rothman
```



To give someone administrative privileges, add that user to the *admin* group (*/groups/admin*). This gives him or her the ability to use *sudo* and install software that requires such privileges.

Listing Groups with *nidump*

Use *nidump* to confirm that the new group was created correctly. To list groups with *nidump*, pass in the format (in this case, the *group* file) and the domain (., the local domain).

```
% nidump group . | grep writers
writers:*:5005:bjepson,rothman
```

Because you can use *nireport* to dump any NetInfo directory, you could also use it to see this information:

```
% nireport . /groups name passwd gid users | grep writers
writers *          5005      bjepson,rothman
```

Deleting a Group

To delete a group, use *nicl*'s *-delete* switch. Be careful with this switch, since it will delete everything in and below the specified NetInfo directory:

```
# nicl / -delete /groups/writers
```

Managing Users and Passwords

The NetInfo equivalent of the *passwd* file resides under the */users* portion of the NetInfo directory. Mac OS X uses */etc/master.passwd* for storing encrypted passwords. That *master.passwd* file is called the *shadow password file* because it shadows the other, nonsensitive information contained in the */etc/passwd* file. Only the *root* user can read the shadow password file. Non-privileged users can use the regular *passwd* file to discover other information, such as a user's full name or home directory.

The */etc/passwd* and */etc/master.passwd* files are consulted only while the system is in single-user mode, or if the system has been reconfigured to use BSD Configuration Files (see the “Configuring Directory Services” section, earlier in this chapter). To add a normal user to your system, you should use System Preferences → Accounts. However, if you want to bulk-load NetInfo with many users or create a user while logged in over *ssh*, you can use *nicl* or *niload*.

You can list all users with the *nireport* utility. Supply the NetInfo domain (., the local domain), the directory (*/users*), and the properties you want to inspect (*uid*, *name*, *home*, *realname*, and *shell*):

```
% nireport . /users uid name home realname shell
-2  nobody  /dev/null          Unprivileged User   /dev/null
0   root     /var/root          System Administrator /bin/tcsh
1   daemon  /var/root          System Services     /dev/null
99  unknown /dev/null          Unknown User         /dev/null
70  www      /Library/WebServer World Wide Web Server /dev/null
```

Creating a User with niload

The *niload* utility understands the flat file format used by */etc/passwd* (name:password:uid:gid:class:change:expire:gecos:home_dir:shell). See the *passwd(5)* manpage for a description of each field. To add a new user, create a file that adheres to that format and load it with *niload*. You can use a here document rather than a separate file. This example creates a user for Ernest Rothman with a UID of 701 and membership in the *staff* (GID 20) group:

```
# niload passwd . <<EOF
? rothman:*:701:20::0:0:Ernest Rothman:/Users/rothman:/bin/tcsh
? EOF
```

As you can see from this example, we set the password field to *, which disables logins for that account. To set the password, we’ll use the *passwd* command to set it:

```
# passwd rothman
Changing password for rothman.
New password: *****
Retype new password: *****
```

If you *niload* a user that already exists, that user will not be overwritten or changed in any way. You should delete the user with *nicl* first (see the “Deleting a Group” section, earlier in this chapter). Before the user can log in, you must create his home directory (see the “Creating a User’s Home Directory” section, later in this chapter).

Creating a User with *nicl*

To create a user with *nicl*, you'll need to create a directory under */users*, and set the *uid*, *gid*, *shell*, *realname*, and *home* properties. The following code creates the same user shown in the previous section, "Creating a User with *niload*."

```
# nicl / -create /users/rothman uid 701
# nicl / -create /users/rothman gid 20
# nicl / -create /users/rothman shell /bin/tcsh
# nicl / -create /users/rothman home /Users/rothman
# nicl / -create /users/rothman realname "Ernest Rothman"
# nicl / -create /users/rothman passwd \*
```

After you create the user, you should set the password as shown in the previous section.

Creating a User's Home Directory

One thing that NetInfo can't do for you is create the user's home directory. Mac OS X keeps a skeleton directory under the */System/Library/User Template* directory. If you look in this directory, you'll see localized versions of a user's home directory. To copy the localized English version of the home directory, use the *ditto* command:

```
# ditto /System/Library/User\ Template/English.lproj /Users/rothman
```

Then, use *chown* to recursively set the ownership of the home directory and all its contents (make sure you set the group to a group of which the user is a member):

```
# chown -R rothman:staff /Users/rothman
```

This change makes the new user the owner of his home directory and all its contents.

Modifying a User

You can change a user's properties by deleting a property with *-delete* and adding it again with *-create*. For example, to change *rothman*'s shell to *bash*, use:

```
# nicl / -delete /users/rothman shell
# nicl / -create /users/rothman shell /bin/bash
```



You can also modify most user settings with System Preferences → Accounts.

Listing Users with *nidump*

Use *nidump* to confirm that *rothman* was added successfully. To list users with *nidump*, pass in the format (in this case, the *passwd* file) and the domain (use *.* for the local domain):

```
% nidump passwd . | grep rothman  
rothman:nIhowm2p0BGsE:701:20::0:0:Ernest Rothman:/Users/rothman:/bin/tcsh
```

Deleting a User

To delete a user, use *nicl*'s *-delete* switch. Since *-delete* recursively deletes everything under the specified directory, use this option with caution:

```
# nicl / -delete /users/rothman
```

If you want to also delete that user's home directory, you will have to do it manually, using *rm -r*.

Managing Hostnames and IP Addresses

Mac OS X 10.1 (and earlier versions of Mac OS X) only consulted */etc/hosts* in single-user mode and stored hostname/IP address mappings in the */machines* portion of the NetInfo database. As of Mac OS X 10.2, you can use the */etc/hosts* file to map hostnames to IP addresses. For example, the following entry would map the hostname *xyzzzy* to 192.168.0.1:

```
192.168.0.1    xyzzzy
```

Creating a Host with *niload*

If you are using Mac OS X 10.1.5 (or an earlier version of Mac OS X), you'll need to use NetInfo to maintain the hosts database. The *niload* utility understands the flat file format used by */etc/hosts* (*ip_address:name*). See the *hosts(5)* manpage for a description of each field. To add a new host, create a file using that format and load it with *niload*. This example uses a here document instead of a separate file to add the host *xyzzzy*:

```
# niload hosts . <<EOF  
? 192.168.0.1 xyzzzy  
? EOF
```

If you add an entry that already exists, it will not be changed or overwritten. Instead, *niload* will silently ignore your command. (To override this behavior, see the *-m* option under "niload" in the "NetInfo Command Reference" section, earlier in this chapter.)

Exporting Directories with NFS

Mac OS X 10.1 (and earlier versions of Mac OS X) stored NFS exports in the */exports* portion of the NetInfo directory. As of Mac OS X 10.2, however, you can use the */etc/exports* file. For example, the following line exports the */Users* directory to two hosts (192.168.0.134 and 192.168.0.106):

```
/Users -ro -mapall=nobody 192.168.0.134 192.168.0.106
```

The NFS server will start automatically at boot time if there are any exports in the NetInfo database. After you've set up your exports, you can reboot, and NFS should start automatically. NFS options supported by Mac OS X include the following:

-maproot=user

Specifies that the remote *root* user should be mapped to the specified user. You may specify either a username or numeric user ID.

-maproot=user:[group[:group...]]

Specifies that the remote *root* user should be mapped to the specified user with the specified group credentials. If you include the colon with no groups, as in *-maproot=username:*, it means the remote user should have no group credentials. You may specify a username or numeric user ID for *user* and a group name or numeric group ID for *group*.

-mapall=user

Specifies that all remote users should be mapped to the specified user.

-mapall=user:[group[:group...]]

Specifies that all remote users should be mapped to the specified user with the specified group credentials. If you include the colon with no groups, as in *mapall=username:*, it specifies that the remote user should be given no group credentials.

-kerb

Uses a Kerberos authentication server to authenticate and map client credentials.

-ro

Exports the file system read-only. The synonym *-o* is also supported.

Flat Files and Their NetInfo Counterparts

As was mentioned earlier, NetInfo managed information for several flat files in earlier releases of Mac OS X, including */etc/printcap*, */etc/mail/aliases*, */etc/protocols*, and */etc/services*. For a complete list of known flat file formats, see the *nidump* and *niload* manpages.

Although you can edit these flat files directly as you would on any other Unix system, you can also use NetInfo to manage this information. You can use *niload* with a supported flat file format to add entries to NetInfo, or you can use *nicl* or NetInfo Manager to directly manipulate the entries. Table 3-2 lists each flat file, the corresponding portion of the NetInfo directory, and important properties associated with each entry. See the *netinfo(5)* manpage for complete details. Properties marked with (list) can take multiple values. (For an example, see the “Adding Users to a Group” section, earlier in this chapter).

The “Wired into Directory Services?” column in Table 3-2 indicates whether Directory Services will consult the flat file when the BSD configuration files plug-in is enabled. In some cases, an operating system daemon may bypass Directory Services and consult a flat file, and Directory Services then has no knowledge of the flat file. For example, *nfsd* can consult either Directory Services or */etc/exports*, but if it uses the flat file, it is consulting it directly and is not going through Directory Services.

Table 3-2. Flat files and their NetInfo counterparts

Flat file	NetInfo directory	Important properties	Wired into Directory Services?
<i>/etc/exports</i>	<i>/exports</i>	<i>name, clients (list), opts (list)</i>	No
<i>/etc/fstab</i>	<i>/mounts</i>	<i>name, dir, type, opts (list), passno, freq</i>	Yes
<i>/etc/group</i>	<i>/groups</i>	<i>name, passwd, gid, users (list)</i>	Yes
<i>/etc/hosts</i>	<i>/machines</i>	<i>ip_address, name (list)</i>	Yes
<i>/etc/mail/aliases</i>	<i>/aliases</i>	<i>name, members (list)</i>	No
<i>/etc/networks</i>	<i>/networks</i>	<i>name (list), address</i>	No
<i>/etc/passwd, /etc/master.passwd</i>	<i>/users</i>	<i>name, passwd, uid, gid, realname, home, shell</i>	Yes
<i>/etc/printcap</i>	<i>/printers</i>	<i>name, and various printcap properties (see the printcap(5) manpage)</i>	No
<i>/etc/protocols</i>	<i>/protocols</i>	<i>name (list), number</i>	No
<i>/etc/rpc</i>	<i>/rpcs</i>	<i>name (list), number</i>	No
<i>/etc/services</i>	<i>/services</i>	<i>name (list), port, protocol (list)</i>	No

Restoring the NetInfo Database

If the NetInfo database is damaged, boot into single-user mode by holding down **⌘-S** as the system starts up. Next, check to see if you have a backup of the NetInfo database. The */etc/daily cron* job backs up the NetInfo database each time it is run. You can find the backup in */var/backups/local.nidump*. If you don’t have a backup, you won’t be able to restore the NetInfo settings.

The *local.nidump* file is overwritten each time the *cron* job runs, so make sure you back it up regularly (preferably to some form of removable media).



If your computer is generally not turned on at 3:15 a.m. (the default time for the *daily cron* job), you'll never get a backup of your NetInfo database. You can solve this problem by editing */etc/crontab* to run this job at a different time, or to run the job periodically with the command *sudo periodic daily*. See the “Default cron Jobs” section in Chapter 2 for more details.

After the system boots in single-user mode, you should:

1. Log in as the *root* user.
2. Fix any filesystem errors:

```
# /sbin/fsck -y
```
3. Mount the *root* filesystem as read/write:

```
# /sbin/mount -uw /
```
4. Change directories and go to the NetInfo database directory:

```
# cd /var/db/netinfo/
```
5. Move the database out of the way and give it a different name:

```
# mv local.nidb/ local.nidb.broken
```
6. Switch directories and go to the *StartupItems* directory:

```
# cd /System/Library/StartupItems
```
7. Start the network:

```
# ./Network/Network start
```
8. Start the port mapper, since NetInfo depends on it. Ignore any warnings about a missing NetInfo database:

```
# ./Portmap/Portmap start
```
9. Start NetInfo. Since it has to rebuild the NetInfo database, this may take several minutes:

```
# ./DirectoryServices/DirectoryServices start
```
10. Load the backup into NetInfo:

```
# /usr/bin/niload -d -r / . < /var/backups/local.nidump
```
11. Create the *.AppleSetupDone* file so that the Setup Assistant does not appear when you reboot:

```
# touch /var/db/.AppleSetupDone
```

After you have restored the NetInfo database, reboot the system with the *reboot* command.

