Part 4 Administering NIS

This part describes the *Network Information Service* (NIS) and how to administer it.

- ***CHAPTER 1, Network Information Service (NIS)***
- ***CHAPTER 2, Administering NIS***

CHAPTER 1

Network Information Service (NIS)

This chapter describes NIS, the Network Information Service.

See *Solaris Naming Setup and Configuration Guide* for information on how to initially setup and configure NIS.

NIS Introduction

NIS is a distributed name service. It is a mechanism for identifying and locating network objects and resources. It provides a uniform storage and retrieval method for network−wide information in a transport−protocol and media−independent fashion.

By running the service, the system administrator can distribute administrative databases, called *maps*, among a variety of servers (*master* and *slaves*), and update those databases from a centralized location in an automatic and reliable fashion to ensure that all clients share the same name service information in a consistent manner throughout the network. For additional overview and background information on NIS, see *NIS @ 1−3*.

NIS was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of

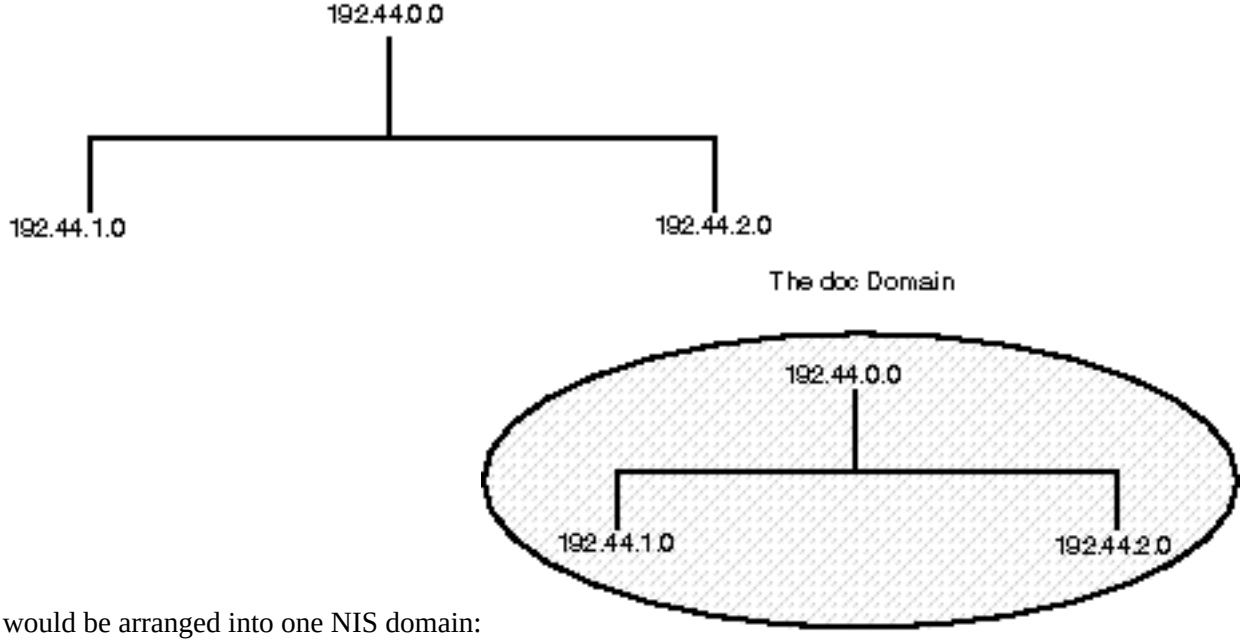
network information. NIS stores information not only about machine names and addresses, but also about users, the network itself, and network services. This collection of network *information* is referred to as the NIS *namespace*.

Note − In some contexts *machine* names are referred to has *host* names or *workstation* names. This discussion uses *machine*, but some screen messages or NIS map names may use *host* or *workstation*.

NIS Architecture

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

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



would be arranged into one NIS domain:

A NIS domain cannot be connected directly to the Internet using just NIS. However, organizations that want to use NIS and also be connected to the Internet can combine NIS with DNS. You can use NIS to manage all local information and use DNS for Internet host lookup. NIS provides a forwarding service that forwards host lookups to DNS if the information cannot be found in a NIS map. The Solaris operating system also allows you to set up the nsswitch.conf file so that hosts lookup requests go only to DNS, or to DNS and then NIS if not found by DNS, or to NIS and then DNS if not found by NIS. (See *CHAPTER 2, The Name Service Switch*, for details.)

NIS and NIS+

Both NIS and NIS+ perform some of the same tasks. NIS+, however, allows for hierarchical domains, namespace security, and other features that NIS does not provide. For a more detailed comparison between NIS and NIS+, see *How NIS+ Differs From NIS @ 1−3*.

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

- Servers within a domain. While you can have both NIS and NIS+ servers operating in the same domain, doing so is not recommended for long periods. As a general rule, using both services in the same domain should be limited to a relatively short transition period from NIS to NIS+. If some clients need NIS service, you can run NIS+ in NIS compatibility mode as explained *Solaris 1.x Releases and NIS−Compatibility Mode @ 1−6*.
- Subdomains. If the master server of your root domain is running NIS+, you can set up subdomains whose servers are all running NIS. (If your root domain master server is running NIS, you cannot have subdomains.) This might be useful in situations where you are moving from NIS to NIS+. For example, suppose your enterprise had separate, multiple NIS domains, possibly at different sites. Now you need to link them all together into a single, hierarchical multi−domain namespace under NIS+. By first setting up the root domain under NIS+, you can then designate the legacy NIS domains as sub−domains that continue to run NIS until it is convenient to switch them over to NIS+
- Machines within a domain.
    - If a domain's servers are running NIS+, individual machines within that domain can be set up to use either NIS+, NIS, or /etc files for their name service information. In order for an NIS+ server to supply the needs of an NIS client, the NIS+ server must be running in NIS−Compatibility mode as described in *Solaris Naming Setup and Configuration Guide*.
    - If a domain's servers are running NIS, individual machines within that domain can be set up to use either NIS or /etc files for name services (they cannot use NIS+).

Which service a machine uses for various name services is controlled by the machine's nsswitch.conf file. This file is called the *switch* file. See *CHAPTER 2, The Name Service Switch* for further information.

NIS and FNS

Under certain conditions, FNS commands can be used by NIS clients to update naming information that pertains to them such as file systems and printers. (See *NIS Clients Can Update Contexts With FNS if SKI is Running @ 2−1* for details.)

NIS Machine Types

There are three types of NIS machines:

- Master server
- Slave servers
- Clients of NIS servers

Any machine can be an NIS client, but only machines with disks should be NIS servers, either master or slave. Servers are also clients, typically of themselves.

NIS Servers

By definition, an NIS server is a machine storing a set of maps that are available to network machines and applications. The NIS server does not have to be the same machine as the NFS file server.

NIS servers come in two varieties, master and slave. The machine designated as master server contains the set of maps that you, the NIS administrator, create and update as necessary. Each NIS domain must have one, and only one, master server. This should be a machine that can propagate NIS updates with the least performance degradation.

You can designate additional NIS servers in the domain as slave servers. A slave server has a complete copy of the master set of NIS maps. Whenever the master server maps are updated, the updates are propagated among the slave servers. The existence of slave servers allows the system administrator to evenly distribute the load resulting from answering NIS requests. It also minimizes the impact of a server becoming unavailable.

Normal practice is to designate one master server for all NIS maps. However, because each individual NIS map has the machine name of the master server encoded within it, you could designate different servers to act as master and slave servers for different maps. Note, however, that randomly designating a server as master of one map and another server as master of another map can cause a great deal of administrative confusion. For that reason it is best to have a single server be the master for all the maps you create within a single domain. The examples in this chapter assume that one server is the master for all maps in the domain.

NIS Clients

NIS clients run processes that request data from maps on the servers. Clients do not make a distinction between master and slave servers, since all NIS servers should have the same information.

NIS servers are also clients, typically though not necessarily, of themselves. For information on how to create NIS clients, refer to the ypbind man page.

NIS Elements

The NIS service is composed of the following elements:

- Domains (see *The NIS Domain @ 1−1*)
- Maps (see *NIS Maps @ 1−4*)
- Daemons (see *NIS Daemons @ 1−2*)
- Utilities (see *NIS Utilities @ 1−3*)

- NIS Command Set (see *Summary of NIS−Related Commands @ 1−5*)

The NIS Domain

An NIS *domain* is a collection of machines that share a common set of NIS maps. Each domain has a domain name and each machine sharing the common set of maps belongs to that domain. Domain names are case−sensitive.

Any machine can belong to a given domain, as long as there is a server for that domain's maps in the same network. Solaris Release 2 machines do not require the server to be on the same subnet, but earlier implementations of NIS historically have required that a server exist on every subnet using NIS. A NIS client machine obtains its domain name and binds to a NIS server as part of its boot process.

NIS Daemons

NIS service is provided by five daemons as shown in *1−1*.

1−1 NIS Daemons

Daemon	Function
`ypserv`	Server process
`ypbind`	Binding process
`ypxfr`	High speed map transfer
`rpc.yppasswdd`	NIS password update daemon
`rpc.ypupdated`	Modifies other maps such as publickey

NIS Utilities

NIS service is supported by nine utilities as shown in *1−1*.

1−1 NIS Utilities

Utility	Function
`makedbm`	Creates dbm file for an NIS map
`ypcat`	Lists data in a map

`ypinit`	Builds and installs an NIS database and initializes NIS client's ypservers list.
`yppmatch`	Finds a specific entry in a map
`yppoll`	Gets a map order number from a server
`yppush`	Propagates data from NIS master to NIS slave server
`ypset`	Sets binding to a particular server
`ypwhich`	Lists name of the NIS server and nickname translation table
`ypxfr`	Transfers data from master to slave NIS server

NIS Maps

NIS stores information in a set of files called maps.

NIS maps were designed to replace UNIX /etc files, as well as other configuration files, so they store much more than names and addresses. On a network running NIS, the NIS master server for each NIS domain maintains a set of NIS maps for other machines in the domain to query. NIS slave servers also maintain duplicates of the master server's maps. NIS client machines can obtain name space information from either master or slave servers.

NIS maps are one type of implementation of Solaris databases. (Other types, not necessarily overlapping, are the files generally found in the /etc directory, the DNS resource records (RRs), and NIS+ tables.)

NIS Maps Overview

NIS maps are essentially two−column tables. One column is the *key* and the other column is information value related to the key. NIS finds information for a client by searching through the keys. Some information is stored in several maps because each map uses a different key. For example, the names and addresses of machines are stored in two maps: `hosts.byname` and hosts.byaddr. When a server has a machine's name and needs to find its address, it looks in the hosts.byname map. When it has the address and needs to find the name, it looks in the hosts.byaddr map.

Maps for a domain are located in each server's /var/yp/*domainname* directory. For example, the maps that belong to the domain test.com are located in each server's /var/yp/test.com directory.

An NIS Makefile is stored in the /var/yp directory of machines designated as a NIS server at installation time. Running `make` in that directory causes `makedbm` to create or modify the default NIS maps from the input files. (See *Solaris Naming Setup and Configuration Guide* for a description of using this process to initially set up your NIS name space.)

Note − Never make the maps on a slave server. Always run `make` on the master server.

The information in NIS maps is stored in `ndbm` format. The ypfiles and ndbm man pages explain the format of the map file.

Default NIS Maps

A default set of NIS maps are provided for you. You may want to use all these maps or only some of them. NIS can also use whatever maps you create or add when you install other software products.

1−1 describes the default NIS maps, information they contain, and whether the operating system consults the corresponding administrative files when NIS is running.

1−1 NIS Map Descriptions

Map Name	Corresponding NIS Admin File	Description
bootparams	bootparams	Contains path names of files clients need during boot: root, swap, possibly others.
ethers.byaddr	ethers	Contains machine names and Ethernet addresses. The Ethernet address is the key in the map.
ethers.byname	ethers	Same as ethers.byaddr, except the key is machine name instead of the Ethernet address.
group.bygid	group	Contains group security information with group ID as key.
group.byname	group	Contains group security information with group name as key.
hosts.byaddr	hosts	Contains machine name, and IP address, with IP address as key.
hosts.byname	hosts	Contains machine name and IP address, with machine (host) name as key.
mail.aliases	aliases	Contains aliases and mail addresses, with aliases as key.
mail.byaddr	aliases	Contains mail address and alias, with mail address as key.
netgroup.byhost	netgroup	Contains group name, user name and machine name.

netgroup.byuser	netgroup	Same as netgroup.byhost, except that key is user name.
netgroup	netgroup	Same as netgroup.byhost, except that key is group name.
netid.byname	passwd, hosts group	Used for UNIX−style authentication. Contains machine name and mail address (including domain name). If there is a netid file available it is consulted in addition to the data available through the other files.
netmasks.byaddr	netmasks	Contains network mask to be used with IP submitting, with the address as the key.
networks.byaddr	networks	Contains names of networks known to your system and their IP addresses, with the address as the key.
networks.byname	networks	Same as networks.byaddr, except key is name of network.
passwd.adjunct. byname	passwd and shadow	Contains auditing information and the hidden password information for C2 clients.
passwd.byname	passwd and shadow	Contains password information with user name as key.
passwd.byuid	passwd and shadow	Same as passwd.byname, except that key is user ID.
protocols.byname	protocols	Contains network protocols known to your network.
protocols.bynumber	protocols	Same as protocols.byname, except that key is protocol number.
rpc.bynumber	rpc	Contains program number and name of RPCs known to your system. Key is RPC program number.
services.byname	services	Lists Internet services known to your network. Key is port or protocol.
services.byservice	services	Lists Internet services known to your network. Key is service name.
ypservers	N/A	Lists NIS servers known to your network.

Using NIS Maps

NIS makes updating network databases much simpler than with the /etc files system. You no longer have to change the administrative /etc files on every machine each time you modify the network environment.

For example, when you add a new machine to a network running NIS, you only have to update the input file in the master server and run `make`. This automatically updates the hosts.byname and hosts.byaddr maps. These maps are then transferred to any slave servers and are made available to all of the domain's client machines and their programs. When a client machine or application requests a machine name or address, the NIS server refers to the hosts.byname or hosts.byaddr map as appropriate and sends the requested information to the client.

You can use the `ypcat` command to display the values in a map. The `ypcat` basic format is:
`% ypcat` *`mapname`*

Where *mapname* is the name of the map you want to examine or its *nickname*. If a map is composed only of keys, as in the case of ypservers, use `ypcat '-' k`; otherwise, `ypcat` prints blank lines. The ypcat man page describes more options for `ypcat`.

You can use the `ypwhich` command to determine which server is the master of a particular map. Type the following:
`% ypwhich -m` *`mapname`*

Where *mapname* is the name or the nickname of the map whose master you want to find. `ypwhich` responds by displaying the name of the master server. For complete information, refer to the ypwhich man page.

NIS Map Nicknames

Nicknames are aliases for full map names. To obtain a list of available map nicknames, such as `passwd` for passwd.byname, type `ypcat '-' x` or `ypwhich '-' x`.

Nicknames are stored in the /var/yp/nicknames file, which contains a map nickname followed by the fully specified name for the map, separated by a space. This list may be added to or modified. Currently, there is a limit of 500 nicknames.

Summary of NIS-Related Commands

The NIS service includes specialized daemons, system programs, and commands, which are summarized in *1-1*. Refer to their man pages for details about how to use them.

1-1 NIS Command Summary

Command	Description

`ypserv`	Services NIS clients' requests for information from a NIS map. `ypserv` is a daemon that runs on NIS servers with a complete set of maps. At least one `ypserv` daemon must be present on the network for NIS service to function.
`ypbind`	Provides NIS server binding information to clients. It provides binding by finding a `ypserv` process that serves maps within the domain of the requesting client. `ypbind` must run on all servers and clients.
`ypinit`	Automatically creates maps for an NIS server from the input files. It is also used to construct the initial /var/yp/binding/*domain*/ypservers file on the clients. Use `ypinit` to set up the master NIS server and the slave NIS servers for the first time.
`make`	Updates NIS maps by reading the Makefile (when run in the /var/yp directory). You can use `make` to update all maps based on the input files or to update individual maps. The ypmake(1M) man page describes the functionality of `make` for NIS.
`makedbm`	`makedbm` takes an input file and converts it into dbm.dir and dbm.pag files—valid `dbm` files that NIS can use as maps. You can also use `makedbm` '−' u to disassemble a map, so that you can see the key−value pairs that comprise it.
`ypxfr`	Pulls an NIS map from a remote server to the local /var/yp/*domain* directory, using NIS itself as the transport medium. You can run `ypxfr` interactively, or periodically from a crontab file. It is also called by `ypserv` to initiate a transfer.
`ypxfrd`	Provides map transfers service for `ypxfr` requests (generally slave servers). It is run only on the master server.
`yppush`	Copies a new version of an NIS map from the NIS master server to its slaves. You run it on the master NIS server.
`ypset`	Tells a `ypbind` process to bind to a named NIS server. This is not for casual use and its use is discouraged because of security implications. See the ypset(1M) and ypbind(1M) man pages for information about the '−' ypset and '−' ypsetme options to the `ypbind` process.
`yppoll`	Tells which version of an NIS map is running on a server that you specify. It also lists the master server for the map.
`ypcat`	Displays the contents of an NIS map.
`ypmatch`	Prints the value for one or more specified keys in an NIS map. You cannot specify which version of the NIS server map you are seeing.
`ypwhich`	Shows which NIS server a client is using at the moment for NIS services, or, if invoked with the '−' m *mapname* option, which NIS server is master of each of the maps. If only '−' m is used, it displays the names of all the maps available and their respective master servers.

NIS Binding

NIS clients get information from a NIS server through the binding process, which can work in one of two modes: server−list or broadcast.

- Server−list. In the server−list mode, the `ypbind` process queries the /var/yp/binding/*domain*/ypservers list for the names of all of the NIS servers in the domain. The `ypbind` process binds only to servers in this file. The file is created by running `ypinit '−' c`.
- Broadcast. The `ypbind` process can also use an RPC broadcast to initiate a binding. Since broadcasts are only local subnet events that are not routed further, there must be at least one server (master or slave) on the same subnet as the client. The servers themselves may exist throughout different subnets since map propagation works across subnet boundaries. In a subnet environment, one common method is to make the subnet router an NIS server. This allows the domain server to serve clients on either subnet interface.

Server−List Mode

The binding process in server−list mode works as follows:

1. Any program, running on the NIS client machine that needs information provided by an NIS map, asks `ypbind` for the name of a server.
2. `ypbind` looks in the /var/yp/binding/*domainname*/ypservers file for a list of NIS servers for the domain.
3. `ypbind` initiates binding to the first server in the list. If the server does not respond, `ypbind` tries the second, and so on, until it finds a server or exhausts the list.
4. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
5. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
6. `ypserv` sends the requested information back to the client.

Broadcast Mode

The broadcast mode binding process works as follows:

1. `ypbind` must be started with the broadcast option set ('−' broadcast).
2. `ypbind` issues an RPC broadcast in search of an NIS server.

Note − In order to support such clients, it is necessary to have an NIS server on each subnet requiring NIS service.

---

1. `ypbind` initiates binding to the first server that responds to the broadcast.

2. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
3. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
4. `ypserv` sends the requested information back to the client.

Normally, once a client is bound to a server it stays bound to that server until something causes it to change. For example, if a server goes out of service, the clients it served will then bind to new servers.

To find out which NIS server is currently providing service to a specific client, use the following command:

```
% ypwhich machinename
```

Where *machinename* is the name of the client. If no machine name is mentioned, `ypwhich` defaults to the local machine (that is, the machine on which the command is run).

Differences Between This and Earlier NIS Versions

The following features are new or different in Solaris Release 2.6 NIS.

NSKit Discontinued

The most recent Solaris releases have not included NIS service. Up to now, NIS service had to be installed from the unbundled NSKit. NIS has now been included in the Solaris Release 2.6 and there is no 2.6 Release NSKit.

Because NIS service is now part of the Solaris 2.6 Release, the SUNWnsktu and SUNWnsktr packages no longer exist. Instead, NIS is now installed via the NIS Server cluster (containing the SUNWypu and SUNWypr packages).

NIS service is no longer started with the /etc/init.d/yp script which no longer exists. With the Solaris 2.6 Release, you first configure your master server NIS maps with the `ypinit` script, and then start NIS with `ypstart`. NIS service is stopped with the `ypstop` command.

The `ypupdated` Daemon

The most recent versions of NSKit did not include the `ypupdated` daemon. The `ypupdated` daemon is now included in this Solaris release.

/var/yp/securenets

As with the previous NSKit release, the /var/yp/securenets file is now used to limit access to NIS services. If such a file exists on an NIS server, the server only answers queries or supplies maps to machines and

networks listed in the file. For the file format, see the securenets man page.

The following is an example of a securenets file.

```
255.255.255.0 13.13.13.255
host   13.13.14.1
host   13.13.14.2
```

Multihomed Machine Support

As with the previous NSKit release, the `ypserv` process provides support for machines which have more than one network address. When the machine maps are created, the Makefile creates a YP_MULTI_HOSTNAME entry in the map for any machine that has more than one address. This entry lists all the addresses for that machine. When the machine address is needed, an attempt is made to use the closest address on the list. See the ypserv man page for more details.

The determination of closest address is an arithmetic one and as such there is no check for address validity. For example, suppose that a multihomed machine has six IP addresses and only five of the interfaces on the machine are operating normally. Machines on a network that is not directly connected to this multihomed machine can receive the IP address for the down interface from `ypserv`. Thus, this hypothetical client can not reach the multihomed machine.

Note − All addresses for a multihomed machine should normally be active. If a particular address or machine is going to be out of service, remove it from the NIS maps.

Sun Operating System 4.X Compatibility Mode

Solaris 2.6 release NIS supports password configuration files in both the Sun Operating System 4.x (Solaris release 1) password format and the Solaris Release 2 password and shadow file formats.

The mode of operation is determined by the existence of the file $PWDIR/shadow, where $PWDIR is the Makefile macro set in the /var/yp/Makefile file. If the shadow file exists, NIS operates in the Solaris Release 2 mode. If this file does not exist, NIS operates in the SunOS 4.x mode.

In the SunOS 4.x mode, all password information is kept in the passwd file. In the Solaris Release 2 mode, password information is kept in the shadow file and the user account information is kept in the passwd file.

If the `make` macro PWDIR is set to the /etc directory, NIS can operate only in the Solaris Release 2 mode because of the Solaris Release 2 `passwd` processing requirements. However, if PWDIR points to any directory other than /etc, the user has the option of keeping `passwd` configuration files in either the SunOS 4.x format or in the Solaris Release 2 format. The `rpc.yppasswdd` daemon understands both password formats. The Solaris Release 2 format is recommended.

Using the Name Service Switch

The name service switch is designed to simplify name service administration. Client machines and applications use this switch to select a name service. The switch mechanism is implemented using the /etc/nsswitch.conf file, which specifies the source(s) used to resolve references for each information type.

This section discusses only those elements that are needed to properly configure the name service switch for NIS operation. For a more detailed description of the nsswitch.conf file, see *CHAPTER 2, The Name Service Switch.*

An nsswitch.conf file is automatically loaded into every machine's /etc directory by the Solaris 2.6 release software, along with three alternate (template) versions:

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

These alternate template files contain the default switch configurations used by the NIS+ service, NIS, and local files. (See *The nsswitch.conf Template Files @ 2−2*.) No default file is provided for DNS, but you can edit any of these files to use DNS (see *DNS Forwarding for NIS Clients @ 2−2*).

Note that this switch functionality does not exist under SunOS 4.x. Thus, DNS forwarding for 4.x clients must be done on the NIS server. In this situation, if a 4.x client requests information for a host that is not listed in the NIS server's NIS map, the NIS server forwards the client's host request to a DNS server on the client's behalf.

When Solaris 2.6 release software is first installed on a machine, the installer selects the machine's default name service: NIS+, NIS, or local files. During installation, the corresponding template file is copied to /etc/nsswitch.conf. For a machine client using NIS, the installation process copies nsswitch.nis to nsswitch.conf. Unless you have an unusual NIS database setup, the default /etc/nsswitch.nis template file as copied to nsswitch.conf should be sufficient for NIS operation.

When changing a machine client from naming system (/etc, NIS or NIS+) to another, you copy the corresponding template file to nsswitch.conf. You can also change the sources of particular types of network information used by the client by editing the appropriate lines of the /etc/nsswitch.conf file. See *Solaris Naming Setup and Configuration Guide*, and *CHAPTER 2, The Name Service Switch*.

Caution − If the /etc/nsswitch.conf file is set to files and not nis for host information, and the server is not included in the /etc/hosts file, then the `ypcat` command generates the following error message: RPC failure: "RPC failure on yp operation"

CHAPTER 2

Administering NIS

This chapter describes how to administer NIS.

See *CHAPTER 1, Network Information Service (NIS),* for a general description of NIS.

See *Solaris Naming Setup and Configuration Guide* for information on how to initially set up and configure NIS.

Password Files and Namespace Security

For security reasons:

- It is best to limit access to the NIS maps on the master server.
- The files used to build the NIS password maps should not contain an entry for root to protect against unauthorized access. To accomplish this, the password files used to build the password maps should have the root entry removed from them and be located in a directory other than the master server's

/etc directory. This directory should be secured against unauthorized access.

For example, the master server password input files could be stored in a directory such as /var/yp, or any directory of your choice, as long as the file itself is not a link to another file and is specified in the Makefile. The /usr/lib/netsvc/yp/ypstart script automatically sets the correct directory option according to the configuration specified in your Makefile.

Note − In addition to the older Solaris 1.x version passwd file format, this implementation of NIS accepts the Solaris Release 2 passwd and shadow file formats as input for building the NIS password maps.

Administering NIS Users

This section includes information about setting user passwords, adding new users to an NIS domain, and assigning users to netgroups.

Adding a New User to an NIS Domain

To add a new NIS user:

1. **Log in as root on the master NIS server.**
2. **Create the new user's login ID with the** `useradd` **command.**

   For Solaris Release 2 systems, type the following:
   ```
   # useradd userID
   ```

   Where *userID* is the login ID of the new user. This command creates entries in the /etc/passwd and /etc/shadow files on the master NIS server.
3. **Create the new user's initial password.**

   To create an initial password that the new user can use to log in, run the `passwd` command in the form:
   ```
   # passwd userID
   ```

   Where *userID* is the login ID of the new user. You will be prompted for the password to assign to this user.

   This step is necessary because the password entry created by the `useradd` command is locked, which means that the new user cannot log in. By specifying an initial password, you unlock the entry.
4. **If necessary, copy the new entry into the server's passwd map input files.**

   If the map source files on your master server are in a directory other than /etc (as they should be), you have to copy and paste the new lines from the /etc/passwd and /etc/shadow files into the passwd map input files on the server. (See *Password Files and Namespace Security @ 2−1* for additional information on this matter.)

   For example, if you added the new user baruch, the line from /etc/passwd that you would copy to your passwd input file would look like:

```
baruch:x:123:10:User baruch:/home/baruch:/bin/csh:
```

The line for baruch that you would copy from /etc/shadow would look like:

```
baruch:W12345GkHic:6445::::::
```

Note − If you are using a Solaris Release 1 passwd file format as input for your NIS maps, you must use a text editor to add the new user to your passwd file, manually.

---

5. **Make sure that the Makefile correctly specifies the directory where the password input file resides.**

6. **If appropriate, delete the new user's entries from /etc/passwd and /etc/shadow input files.**

   For security reasons, it is not good practice to maintain user entries in the NIS master server /etc/passwd and /etc/shadow files. After copying the entries for the new user to the NIS map source files that are stored in some other directory, use the `userdel` command on the master server to delete the new user.

   For example, to delete the new user baruch from the master server's /etc files, you would enter:

   ```
   # userdel baruch
   ```

   For more information about `userdel`, see the userdel man page.

7. **Update the NIS passwd maps.**

   After you have updated the passwd input file on the master server, update the passwd maps by running `make` in the directory containing the source file.

   ```
   # userdel baruch
   # cd /var/yp
   # /usr/ccs/bin/make passwd
   ```

8. **Tell the new user the initial password you have assigned to his or her login ID.**

   After logging in, the new user can run passwd at any time to establish a different password.

User Passwords

Users run `passwd` to change their passwords.

```
% passwd username
```

(See *Using Passwords @ 4−1* for a complete description of password matters from the users point of view.)

Before users can change their passwords, you must start the `rpc.yppasswdd` daemon on the master server to update the password file. The commands for starting the daemon are already present in the /usr/lib/netsvc/yp/ypstart file.

The `rpc.yppasswdd` daemon is started automatically by `ypstart` on the master server. Notice that when the '−' m option is given to `rpc.yppasswd`, a `make` is forced in /var/yp immediately following a modification of the file. If you want to avoid having this `make` take place each time the passwd file is changed, remove the '−' m option from the `rpc.yppasswd` command in the `ypstart` script and

control the pushing of the passwd maps through the crontab file.

Note − No arguments should follow the `rpc.yppasswd '−' m` command. Although you can edit the ypstart script file to achieve a different action, it is not recommended that you modify this file other than optionally removing the `'−' m` option. All commands and daemons invoked by this file with the proper set of command line parameters. If you choose to edit this file, be especially careful when editing the `rpc.yppasswdd` command. If you add an explicit call to the passwd.adjunct file, the exact $PWDIR/security/passwd.adjunct path must be used; otherwise, incorrect processing results.

Netgroups

NIS netgroups are groups (sets) of users or machines that you define for your administrative purposes. For example, you can create netgroups that:

- Define a set of users who can access a specific machine
- Define a set of NFS client machines to be given some specific filesystem access.
- Define a set of users who are to have administrator privileges on all the machines in a particular NIS domain.

Each netgroup is given a netgroup name. Netgroups do not directly set permissions or access rights. Instead, the netgroup names are used by other NIS maps in places where a user name or machine name would normally be used. For example, suppose you created a netgroup of network administrators called netadmins. To grant all members of the netadmins group access to a given machine, you need only add a netadmin entry to that machine's /etc/passwd file. Netgroup names can also be added to the NIS group map. See the netgroup man page for more detailed information on using netgroups.

On a network using NIS, the netgroup input file on the master NIS server is used for generating three maps: netgroup, netgroup.byuser, and netgroup.byhost. The netgroup map contains the basic information in the netgroup input file. The two other NIS maps contain information in a format that speeds lookups of netgroup information, given the machine or user.

Entries in the netgroup input file are in the format: *name ID*, where *name* is the name you give to a netgroup, and *ID* identifies a machine and/or user who belongs to the netgroup. You can specify as many ids (members) to a netgroup as you want, separated by commas. For example, to create a netgroup with three members, the netgroup input file entry would be in the format: *name ID, ID, ID.* The member IDs in a netgroup input file entry are in the format:

`([−|`*`machine`*`], [−|`*`user`*`], [`*`domain`*`])`

Where *machine* is a machine name, *user* is a user ID, and *domain* is the machine or user's NIS domain with each element separated by a comma. The domain element is optional and should only be used to identify machines or users in some other NIS domain. The *machine* and *user* element of each member's entry are required, but a dash (−) is used to denote a null. There is no necessary relationship between the machine and user elements in an entry.

For example, below are two sample netgroup input file entries, each of which create a netgroup named admins composed of the users hauri and juanita who is in the remote domain sales and the machines altair and sirius.

```
admins (altair, hauri), (sirius,juanita,sales)
admins (altair,-), (sirius,-), (-,hauri), (-,juanita,sales)
```

Various programs use the netgroup NIS maps for permission checking during login, remote mount, remote login, and remote shell creation. These programs include: `mountd`, `login`, `rlogin`, and `rsh`. The `login` command consults the netgroup maps for user classifications if it encounters netgroup names in the `passwd` database. The `mountd` daemon consults the netgroup maps for machine classifications if it encounters netgroup names in the /etc/dfs/dfstab file. `rlogin` and `rsh` (in fact, any program that uses the `ruserok` interface) consults the netgroup maps for both machine and user classifications if they encounter netgroup names in the /etc/hosts.equiv or .rhosts files.

If you add a new NIS user or machine to your network, be sure to add them to appropriate netgroups in the netgroup input file. Then use the `make` and `yppush` commands to create the netgroup maps and push them to all of your NIS servers. See the netgroup man page for detailed information on using netgroups and netgroup input file syntax.

Working With NIS Maps

The following sections describe how to administer NIS maps.

Obtaining Map Information

Users can obtain information from and about the maps at any time by using the `ypcat`, `ypwhich`, and `ypmatch` commands. In the examples that follow, mapname refers both to the official name of a map and to its nickname, if any.

To list all the values in a map, type:
```
% ypcat mapname
```

To list both the keys and the values (if any) in a map, type:
```
% ypcat -k mapname
```

To list all the map nicknames, type any of the following commands:
```
% ypcat -x
% ypwhich -x
% ypmatch -x
```

To list all the available maps and their master(s), type:
```
% ypwhich -m
```

To list the master server for a particular map, type:
```
% ypwhich -m mapname
```

To match a key with an entry in a map, type:
```
% ypmatch key mapname
```

If the item you are looking for is not a key in a map, type:
```
% ypcat mapname | grep item
```

Where *item* is the information you are searching for. To obtain information about other domains, use the '−' d domainname options of these commands.

If the machine requesting information for a domain other than its default does not have a binding for the requested domain, it causes `ypbind` to consult the /var/yp/binding/*domainname*/ypservers file for a list of servers for that domain. If this file doesn't exist it issues an RPC broadcast for a server. In this case, there must be a server for the requested domain on the same subnet as the requesting machine.

Changing a Map's Master Server

To change the master server for a selected map, you first have to build the map on the new NIS master. Since the old master server name occurs as a key−value pair in the existing map (this pair is inserted automatically by `makedbm`), copying the map to the new master or transferring a copy to the new master with `ypxfr` is insufficient. You have to reassociate the key with the new master server name. If the map has an ASCII source file, you should copy this file to the new master.

Here are instructions for remaking a sample NIS map called sites.byname.

1. **Log in to the new master as superuser and type:**
   ```
   newmaster# cd /var/yp
   ```
2. **Makefile must have an entry for the new map before you specify the map to make. If this is not the case, edit the Makefile now.**
3. **To update or remake the map, type:**
   ```
   newmaster# make sites.byname
   ```
4. **If the old master remains an NIS server, remote log in (`rlogin`) to the old master and edit Makefile. Comment out the section of the Makefile that made sites.byname so that it is no longer made there.**
5. **If sites.byname only exists as an `ndbm` file, remake it on the new master by disassembling a copy from any NIS server, then running the disassembled version through `makedbm`:**
   ```
   newmaster# cd /var/yp
   newmaster# ypcat −k sites.byname | makedbm −domain/sites.byname
   ```

   After making the map on the new master, you must send a copy of the new map to the other slave servers. However, do not use `yppush`, because the other slaves will try to get new copies from the old master, rather than the new one. A typical method for circumventing this is to transfer a copy of the map from the new master back to the old master. To do this, become superuser on the old master server and type:
   ```
   oldmaster# /usr/lib/netsvc/yp/ypxfr −h newmaster sites.byname
   ```

   Now it is safe to run `yppush`. The remaining slave servers still believe that the old master is the current master. They attempt to get the current version of the map from the old master. When they do so, they will get the new map, which names the new master as the current master.

   If this method fails, you can try this cumbersome but sure−fire option: log in as root on each NIS server and execute the `ypxfr` command shown above.

Modifying Configuration Files

NIS intelligently parses the setup files. Although this makes NIS administration easier, it does make the behavior of NIS more sensitive to changes in the setup and configuration files.

Use the procedures in this section when modifying any of the following:

- /var/yp/Makefile to add or delete supported maps
- Add or delete /etc/resolv.conf to allow or deny DNS forwarding
- Add or delete $PWDIR/security/passwd.adjunct to allow or deny C2 security. ($PWDIR is defined in /var/yp/Makefile.)

To modify any of the listed files:

1. **Stop the NIS server by typing;**
   ```
   # /etc/init.d/yp stop
   ```
2. **Make the necessary changes to your files.**
3. **Restart the NIS server by typing:**
   ```
   # /etc/init.d/yp start
   ```

You do not have to stop and start NIS when changing NIS maps or the map source files.

Keep in mind the following points:

- Deleting a map or source file from a NIS master server does not automatically result in corresponding deletions from slave servers. You must delete maps and source files from slave servers by hand.
- New maps do not automatically get pushed to existing slave servers. You must run `ypxfr` from the slaves.

Modifying and Using the Makefile

You can modify the Makefile provided by default in /var/yp to suit your needs. (Be sure to keep an unmodified copy of the original Makefile for future reference.) You can add or delete maps, and you can change the names of some of the directories.

To add a new NIS map, you must get copies of the `ndbm` files for the map into the /var/yp/*domainname* directory on each of the NIS servers in the domain. This is normally done for you by the Makefile. After deciding which NIS server is the master of the map, modify the `Makefile` on the master server so that you can conveniently rebuild the map. Different servers can be masters of different maps, but in most cases this leads to administrative confusion, and it is strongly recommended that you set only one server as the master of all maps.

Typically a human−readable text file is filtered through `awk`, `sed`, or `grep` to make it suitable for input to `makedbm`. Refer to the default Makefile for examples. See the `make` man page for general information about the `make` command.

Use the mechanisms already in place in the Makefile when deciding how to create dependencies that `make` will recognize. Be aware that `make` is very sensitive to the presence or absence of tabs at the

beginning of lines within the dependency rules, and a missing tab can invalidate an entry that is otherwise well formed.

Adding Makefile Entries

To add an entry to the Makefile, do the following:

- Add the name of the database to the all rule
- Write the time rule
- Add the rule for the database

For example, in order for the Makefile to work on automounter input files, you would have to add the auto_direct.time and auto_home.time maps to the NIS database.

To add these maps to the NIS database:

1. **Modify the line that starts with the word all by adding the name(s) of the database you want to add:**

   ```
   all: passwd group hosts ethers networks rpc services protocols \
    netgroup bootparams aliases netid netmasks \
    auto_direct auto_home auto_direct.time auto_home.time
   ```

   The order of the entries is not relevant, but the blank space at the beginning of the continuation lines must be a Tab, not spaces.

2. **Add the following lines at the end of the Makefile:**

   ```
   auto_direct: auto_direct.time
   auto_home: auto_home.time
   ```

3. **Add an entry for auto_direct.time in the middle of the file.**

   ```
   auto_direct.time: $(DIR)/auto_direct
    @(while read L; do echo $$L; done < $(DIR)/auto_direct
    $(CHKPIPE)) | \ (sed −e "/^#/d" −e "s/#.*$$//" −e "/^ *$$/d"
    $(CHKPIPE)) | \ $(MAKEDBM) − $(YPDBDIR)/$(DOM)/auto_direct;
    @touch auto_direct.time;
    @echo "updated auto_direct";
    @if [ ! $(NOPUSH) ]; then $(YPPUSH) auto_direct; fi
    @if [ ! $(NOPUSH) ]; then echo "pushed auto_direct"; fi
   ```

   Where:

   - CHKPIPE makes certain that the operations to the left of the pipe (|) are successfully completed before piping the results to next commands. If the operations to the left of the pipe do not successfully complete, the process is terminated with a "NIS make terminated" message.
   - NOPUSH prevents the makefile from calling `yppush` to transfer the new map to the slave servers. If `NOPUSH` is not set, the push is done automatically.

   The while loop at the beginning is designed to eliminate any backslash−extended lines in the input file. The `sed` script eliminates comment and empty lines, and feeds the output to

The same procedure should be followed for all other automounter maps such as auto_home, or any other nondefault maps.

4. **Run make.**
   ```
   # make name
   ```
   Where *name* is the name of the map you want to make. For example, auto_direct.

Deleting Makefile Entries

If you do not want the `Makefile` to produce maps for a specific database, edit the Makefile as follows:

1. **Delete the name of the database from the all rule.**
2. **Delete or comment out the database rule for the database you want to delete.**

   For example, to delete the hosts database, the hosts.time entry should be removed.
3. **Remove the time rule.**

   For example, to delete the hosts database, the hosts: hosts.time entry should be removed.
4. **Remove the map from the master and slave servers.**

Changing Makefile Macros/Variables

You can change the settings of the variables defined at the top of the Makefile simply by changing the value to the right of the equal sign (=). For instance, if you do not want to use the files located in /etc as input for the maps, but you would rather use files located in another directory, such as /var/etc/domainname, you should change the value of DIR from DIR=/etc to DIR=/var/etc/*domainname*. You may also change the value of PWDIR from PWDIR=/etc to PWDIR=/var/etc/*domainname*.

The variables are:

- *DIR*= The directory containing all of the NIS input files except passwd and shadow. The default value is /etc. Since it is not good practice to use the files in the master server's /etc directory as NIS input files, you should change this value.
- *PWDIR*= The directory containing the passwd and shadow NIS input files. Since it is not good practice to use the files in the master server's /etc directory as NIS input files, you should change this value.
- *DOM*= The NIS domain name. The default value of DOM is set using the `domainname` command. Remember that most NIS commands use the current machine's domain which is set in the machine's /etc/defaultdomain file.

Updating Existing Maps

After you have installed NIS, you may discover that some maps require frequent updating while others never need to change. For example, the passwd.byname map may change frequently on a large company's network. On the other hand, the auto_master map changes little, if at all.

When you need to update a map, you can use one of two updating procedures, depending on whether it is a default map or not.

- A default map is a map in the default set created by `ypinit` from the network databases.
- Nondefault maps may be any of the following:
  - Maps included with an application purchased from a vendor
  - Maps created specifically for your site
  - Maps created from a nontext file

The following sections explain how to use various updating tools. In practice, you may decide to only use them if you add nondefault maps or change the set of NIS servers after the system is already up and running.

Modifying Default Maps

Use the following procedure for updating maps supplied with the default set.

1. **Become root on the master server.**

   Always modify NIS maps only on the master server.

2. **Edit the source file for the map you want to change, whether that file resides in /etc or in some other directory of your choice.**

3. **Type the following:**
   `# cd /var/yp# make` *`mapname`*

   The `make` command then updates your map according to the changes you made in its corresponding file. It also propagates the changes among the other servers.

Modifying Nondefault Maps

To update a nondefault map, you must:

1. Create or edit its corresponding text file.
2. Build (or rebuild) the new or updated map. There are two ways to build a map:
   - Use the Makefile. Using the Makefile is the preferred method of building a non−default map. If the map has an entry in the Makefile, simply run `make` *name* where *name* is the name of map you want to build. If the map does not have a Makefile entry, try to create one following the instructions in *Modifying and Using the Makefile @ 2−4*.
   - Use the `/usr/sbin/makedbm` program. (The `makedbm` man page fully describes this

command.)

Using makedbm to Modify a Non−Default Map

There are two different methods for using `makedbm` to modify maps if you do not have an input file:

- Redirect the `makedbm '−' u` output to a temporary file, modify the file, then use the modified file as input to `makedbm`.
- Have the output of `makedbm '−' u` operated on within a pipeline that feeds into `makedbm`. This is appropriate if you can update the disassembled map with either `awk`, `sed`, or a `cat` append.

Creating New Maps

To create new maps, you can use one of two possible procedures: the first method uses an existing text file as input; the second uses standard input.

Creating Maps From Text Files

Assume that a text file /var/yp/mymap.asc was created with an editor or a shell script on the master. You want to create an NIS map from this file and locate it in the *homedomain* subdirectory. To do this, type the following on the master server:

```
# cd /var/yp
# makedbm mymap.asc homedomain/mymap
```

The *mymap* map now exists on the master server in the directory *homedomain*. To distribute the new map to slave servers run `ypxfr`.

Adding Entries to a File−Based Map

Adding entries to mymap is simple. First, you must modify the text file /var/yp/*mymap*.asc. (If you modify the actual `dbm` files without modifying the corresponding text file, the modifications are lost.) Then run `makedbm` as shown above.

Creating Maps From Standard Input

When no original text file exists, create the NIS map from the keyboard by typing input to `makedbm`, as shown below (end with Control−D):

```
ypmaster# cd /var/yp
ypmaster# makedbm − homedomain/mymapkey1 value1 key2 value2 key3 value3
ypmaster#
```

Modifying Maps Made From Standard Input

If you later need to modify the map, you can use `makedbm` to disassemble the map and create a temporary text intermediate file. To disassemble the map and create a temporary file, type the following:

```
% cd /var/yp
% makedbm −u homedomain/mymap > mymap.temp
```

The resulting temporary file *mymap*.temp has one entry per line. You can edit this file as needed, using any text editor.

To update the map, give the name of the modified temporary file to `makedbm` by typing the following:

```
% makedbm mymap.temp homedomain/mymap
% rm mymap.temp
```

Then propagate the map to the slave servers, by becoming root and typing:

```
# yppush mymap
```

The preceding paragraphs explained how to use `makedbm` to create maps; however, almost everything you actually have to do can be done by `ypinit` and Makefile unless you add nondefault maps to the database or change the set of NIS servers after the system is already up and running.

Whether you use the Makefile in /var/yp or some other procedure the goal is the same: a new pair of well−formed `dbm` files must end up in the maps directory on the master server.

Propagating an NIS Map

After a map is changed, the Makefile uses `yppush` to propagate a new map to the slave servers (unless NOPUSH is set in the Makefile). It does this by informing the `ypserv` daemon and sending a map transfer request. The `ypserv` daemon on the slave then starts a `ypxfr` process, which in turn contacts the `ypxfrd` daemon on the master server. Some basic checks are made (for example did the map really change?) and then the map is transferred. `ypxfr` on the slave then sends a response to the `yppush` process indicating whether the transfer succeeded.

Note − The above procedure will *not* work for newly created maps that do not yet exist on the slave servers. New maps must be sent to the slave servers by running `ypxfr` on the slaves.

Occasionally, maps fail to propagate and you must to use `ypxfr` manually to send new map information. You may choose to use `ypxfr` in two different ways: periodically through the root crontab file, or interactively on the command line. These approaches are discussed in the following sections.

Using cron For Map Transfers

Maps have different rates of change. For instance, some may not change for months at a time, such as protocols.byname among the default maps and auto_master among the nondefault maps; but

passwd.byname may change several times a day. Scheduling map transfer using the `crontab` command allows you to set specific propagation times for individual maps.

To periodically run `ypxfr` at a rate appropriate for the map, the root crontab file on each slave server should contain the appropriate `ypxfr` entries. `ypxfr` contacts the master server and transfers the map only if the copy on the master server is more recent than the local copy.

Note − If your master server runs `rpc.yppasswdd` with the default '−' m option, then each time someone changes their yp password, the `passwd` daemon runs `make`, which rebuilds the passwd maps.

Using Shell Scripts with `cron` and `ypxfr`

As an alternative to creating separate crontab entries for each map, you may prefer to have the root crontab command run a shell script that periodically updates all maps. There are sample map−updating shell scripts in the /usr/lib/netsvc/yp directory. The script names are `ypxfr_1perday`, `ypxfr_1perhour`, and `ypxfr_2perday`. You can easily modify or replace these shell scripts to fit your site requirements. *ypxfr_1perday Shell Script @ 2−1* shows the default `ypxfr_1perday` shell script.

ypxfr_1perday Shell Script

```
#! /bin/sh
#
# ypxfr_1perday.sh − Do daily yp map check/updates
PATH=/bin:/usr/bin:/usr/lib/netsvc/yp:$PATH
export PATH
# set −xv
ypxfr group.byname
ypxfr group.bygid
ypxfr protocols.byname
ypxfr protocols.bynumber
ypxfr networks.byname
ypxfr networks.byaddr
ypxfr services.byname
ypxfr ypservers
```

This shell script updates the maps once per day, if the root `crontab` is executed daily. You can also have scripts that update maps once a week, once a month, once every hour, and so forth, but be aware of the performance degradation implied in frequently propagating the maps.

Run the same shell scripts as root on each slave server configured for the NIS domain. Alter the exact time of execution from one server to another to avoid bogging down the master.

If you want to transfer the map from a particular slave server, use the '−' h *machine* option of `ypxfr` within the shell script. Here is the syntax of the commands you put in the script:

`/usr/lib/netsvc/yp/ypxfr −h` *`machine`* `[ −c ]` *`mapname`*

Where *machine* is the name of the server with the maps you want to transfer, and *mapname* is the name of the requested map. If you use the '−' h option without specifying a machine, `ypxfr` tries to get the map

from the master server. If `ypserv` is not running locally at the time `ypxfr` is executed, you must use the '−' c flag so that `ypxfr` does not send a clear current map request to the local `ypserver`.

You can use the '−' s *domain* option to transfer maps from another domain to your local domain. These maps should be the same across domains. For example, two domains might share the same services.byname and services.byaddr maps. Alternatively, you can use `rcp`, or `rdist` for more control, to transfer files across domains.

Directly Invoking `ypxfr`

The second method of invoking `ypxfr` is to run it as a command. Typically, you do this only in exceptional situations —for example, when setting up a temporary NIS server to create a test environment or when trying to quickly get an NIS server that has been out of service consistent with the other servers.

Logging `ypxfr` Activity

The transfer attempts and results of `ypxfr` can be captured in a log file. If a file called /var/yp/ypxfr.log exists, results are appended to it. No attempt to limit the size of the log file is made. To prevent it from growing indefinitely, empty it from time to time by typing:

```
# cd /var/yp
# cp ypxfr.log ypxfr.log.old
# cat /dev/null > /var/yp/ypxfr.log
```

You can have `crontab` execute these commands once a week. To turn off logging, remove the log file.

Adding a New Slave Server

After NIS is running, you may need to create a new NIS slave server that you did not include in the initial list given to `ypinit`.

To add a new NIS server:

1. **Log in to the master server as root.**
2. **Change to the NIS domain directory by typing:**
   `# cd /var/yp/`*`domainname`*
3. **Disassemble the ypservers file, as follows:**
   `# makedbm −u ypservers >/tmp/temp_file`

   The `makedbm` command converts ypservers from `ndbm` format to a temporary ASCII file /tmp/temp_file.
4. **Edit the /tmp/temp_file file using a text editor. Add the name of the new slave server to the list of servers. Then save and close the file.**
5. **Run the `makedbm` command with temp_file as the input file and ypservers as the output file:**

```
# makedbm /tmp/temp_file ypservers
```

`makedbm` then converts ypservers back into `ndbm` format.

6. **Verify that the ypservers map is correct (since there is no ASCII file for ypservers) by typing:**
   ```
   slave3# makedbm −u ypservers
   ```

   The `makedbm` command displays each entry in ypservers on your screen.

Note − If a machine name is not in ypservers, it will not receive updates to the map files because `yppush` consults this map for the list of slave servers.

---

7. **Set up the new slave server's NIS domain directory by copying the NIS map set from the master server.**

   To do this, log in to the new NIS slave as superuser and run the `ypinit` and `ypbind` commands:
   ```
   slave3# cd /var/yp
   slave3# ypinit −c list of servers
   slave3# /usr/lib/netsvc/yp/ypbind
   ```

8. **To initialize this machine as a slave, type the following:**
   ```
   # /usr/sbin/ypinit −s ypmaster
   ```

   Where *ypmaster* is the machine name of the existing NIS master server.

9. **Run `ypstop` to stop the machine running as a NIS client.**
   ```
   #/usr/lib/netsvc/up/ypstop
   ```

10. **Run `ypstart` to start NIS slave service.**
    ```
    #/usr/lib/netsvc/up/ypstart
    ```

    See the *Solaris Naming Setup and Configuration Guide* for a more detailed description of setting up NIS slave servers.

Using NIS with C2 Security

If the $PWDIR/security/passwd.adjunct file is present, C2 security is started automatically. ($PWDIR is defined in /var/yp/Makefile.) The C2 security mode uses the passwd.adjunct file to create the passwd.adjunct NIS map. In this implementation, NIS allows you to use both the passwd.adjunct file and shadow file to manage security. The passwd.adjunct file is only processed when you type:

```
# make passwd.adjunct
```

The make passwd command only processes the passwd map not the passwd.adjunct map when you run `make` manually in the C2 security mode.

Changing a Machine's NIS Domain

To change the NIS domain name of a machine:

1. **Edit the machine's /etc/defaultdomain file, exchanging its present contents with the new domain name for the machine.**

For example, if the current domain name is sales.doc.com, you might change it to research.doc.com.

2. **Run domainname ‘cat /etc/defaultdomain’**
3. **Then set the machine up as a NIS client, slave, or master server.**

   See *Solaris Naming Setup and Configuration Guide* for details.

Using NIS in Conjunction With DNS

Typically, NIS clients are configured with the nsswitch.conf file to use only NIS for machine name and address lookups. If this type of lookup fails, an NIS server may forward these lookups to DNS.

To configure machine name and address lookup to occur through NIS and then through DNS:

1. **The two maps hosts.byname and hosts.byaddr must have the YP_INTERDOMAIN key in them; to set this key, edit the Makefile and modify the lines (at the top of the file) from:**

   ```
   #B=-b
   B=
   ```

   to:

   ```
   B=-b
   #B=
   ```

   This tells `makedbm` to start with the ’−’ b flag when making the maps, and the YP_INTERDOMAIN key will be inserted into the `ndbm` files.

2. **Run make to rebuild that maps.**

   ```
   # /usr/ccs/bin/make hosts
   ```

3. **Make sure that all NIS servers have an /etc/resolv.conf file that points to valid name server(s).**
4. **To enable DNS forwarding, stop each server with the `ypstop` command**

   ```
   # /usr/lib/netsvc/yp/ypstop
   ```

5. **Restart each server with the `ypstart` command:**

   ```
   # /usr/lib/netsvc/yp/ypstart
   ```

   In this implementation of NIS, if a /etc/resolve.conf file exists on the server, `ypstart` automatically starts the `ypserv` daemon with the ’−’ d option to forward requests to DNS.

Note − If you have NIS servers that are not running the Solaris Release 2, then you must make sure that the YP_INTERDOMAIN key is present in the host maps for DNS to be consulted.

Problems in Mixed NIS Domains

Most of the preceding information assumes that both master and slave servers in the NIS domain are running the Solaris Release 2. If that is not the case, problems may arise. *2−1*summarizes how to successfully avoid problems in mixed NIS domains. The notation "4.0.3+" means "release 4.0.3 of the

SunOS operating system or later." The command `makedbm ’−’ b` is a reference to the " ’−’ B" variable in the Makefile.

2−1 NIS/DNS in Heterogeneous NIS Domains

Slave	**Master**		
	4.0.3+		**Solaris NIS**
4.0.3+	Master: makedbm −b Slave: `ypxfr`	Master: makedbm −b Slave: `ypxfr −b`	Master: ypserv −d Slave: `ypxfr −b`
Solaris NIS	Master: makedbm −b Slave: `ypxfr`	Master: makedbm −b Slave: `ypxfr`	Master: ypserv −d Slave: `ypxfr` with resolve.conf or `ypxfr −b`

Turning Off NIS Services

If `ypserv` on the master is disabled, you can no longer update any of the NIS maps. If you choose to turn off NIS on a network currently running it, you can disable NIS after the next reboot by simply renaming the ypbind file to ypbind.orig, as follows:

```
% mv /usr/lib/netsvc/yp/ypbind /usr/lib/netsvc/yp/ypbind.orig
```

To disable NIS after the next reboot on a particular NIS slave or master, type the following on the server in question:

```
% mv /usr/lib/netsvc/yp/ypserv /usr/lib/netsvc/yp/ypserv.orig
```

To stop NIS immediately, type:

```
% /usr/lib/netsvc/yp/ypstop
```

The NIS service is automatically restarted after the next reboot unless the ypbind and ypserv files are renamed as described above.

NIS Problem Solving and Error Messages

- See *NIS Problems and Solutions @ 0−2* and *NIS+ and NIS Compatibility Problems @ 0−4* for problem solving information.
- *APPENDIX B, Error Messages*, for an alphabetic list of the more common namespace error messages and their meanings.