AIX Version 7.2

# *Network Information Service (NIS)*

IBM

**Note**

Before using this information and the product it supports, read the information in "Notices" on page 37.

This edition applies to AIX Version 7.2 and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

**About this document....................v**
Highlighting....................v
Case sensitivity in AIX....................v
ISO 9000....................v

**Network Information Service (NIS).................... 1**
Introduction to Name Services....................1
Domain Name System (DNS) Overview.................... 1
Network Information Service (NIS) Overview.................... 1
Network Information Service.................... 3
NIS Overview.................... 3
Components of NIS.................... 3
NIS Domain....................5
NIS Maps....................5
Maintaining Consistent System Information with NIS.................... 7
NIS Installation and Configuration.................... 7
NIS Maintenance....................19
NIS Automount....................25
NIS Reference.................... 30
Troubleshooting NIS....................31
Identifying NIS Client Problems.................... 31
Identifying NIS Server Problems.................... 34
Migrating from NIS to LDAP services.................... 35
Considerations....................35
Server Setup.................... 35
Migrating data from NIS to LDAP.................... 35
Client Setup.................... 36
Netgroup Setup.................... 36

**Notices....................37**
Privacy policy considerations.................... 38
Trademarks.................... 39

**Index.................... 41**

# About this document

This document provides end users with complete information about how to perform such tasks as installing, configuring, and managing Network Information Service (NIS).

## Highlighting

The following highlighting conventions are used in this document:

| | |
|---|---|
| **Bold** | Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Bold highlighting also identifies graphical objects, such as buttons, labels, and icons that the you select. |
| *Italics* | Identifies parameters for actual names or values that you supply. |
| `Monospace` | Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or text that you must type. |

## Case sensitivity in AIX

Everything in the AIX® operating system is case sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type LS, the system responds that the command `is not found`. Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

## ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

# Network Information Service (NIS)

Whereas DNS focuses on simplification by using workstation names instead of addresses, the Network Information Service (NIS) focuses on simplifying network administration by providing centralized control over a variety of network information.

**Note:** NIS controller server and NIS worker servers are conventionally known as NIS master servers and NIS slave servers. These terms are used interchangeably to use more inclusive language.

## Introduction to Name Services

Learn about name services (also called *network information services*).

### Domain Name System (DNS) Overview

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

The collection of networked workstations that use DNS is referred to as the *DNS namespace*. The DNS namespace can be divided into a hierarchy of *domains*. A DNS domain is a group of workstations. Each domain is supported by two or more *name servers* (a principal server and one or more secondary servers). Each server implements DNS by running a daemon called **named**.

On the client side, DNS is implemented through a *resolver*. The resolver's function is to query a name server, which then returns either the requested information or a referral to another server.

#### Advanced DNS features - BIND version 9.16

For the list of advanced DNS features, see https://bind9.readthedocs.io/en/v9_16_26/advanced.html.

### Network Information Service (NIS) Overview

Whereas DNS focuses on simplification by using workstation names instead of addresses, the Network Information Service (NIS) focuses on simplifying network administration by providing centralized control over a variety of network information.

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

#### NIS Architecture

NIS uses a client-server arrangement similar to DNS.

Replicated NIS servers provide services to NIS clients. The principal servers are called *master* servers and, for reliability, they have backup or *replica* servers (also referred to as *worker* servers). Both server types use the NIS information retrieval software and both store NIS maps.

NIS, like DNS, uses domains to arrange the workstations, users, and networks in its namespace. However, it does not use a domain hierarchy; an NIS namespace is flat. Thus, a hierarchical physical network is arranged by NIS into one domain, as shown in the following figure.

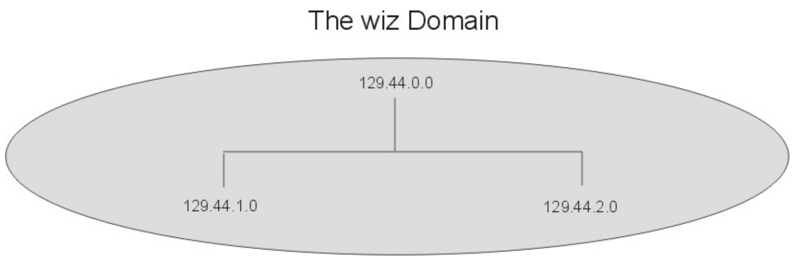

*Figure 1. Example NIS Domain*

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

## NIS Maps

NIS stores information in a set of files called *maps*.

NIS maps were designed to replace traditional UNIX **/etc** files, as well as other configuration files, so they store much more than names and addresses. As a result, the NIS namespace has a large set of maps, as shown in the NIS Maps table below.

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

*Table 1. NIS Maps*

| NIS Map | Description |
|---|---|
| **bootparams** | Lists the names of the diskless clients and the location of the files they need during booting. |
| **ethers.byaddr** | Lists the Ethernet addresses of workstations and their corresponding names. |
| **ethers.byname** | Lists the names of workstations and their corresponding Ethernet addresses. |
| **group.bygid** | Provides membership information about groups, using the group ID as the key. |
| **group.byname** | Provides membership information about groups, using the group name as the key. |
| **hosts.byaddr** | Lists the names and addresses of workstations, using the address as the key. |
| **hosts.byname** | Lists the names and addresses of workstations, using the name as the key. |
| **mail.aliases** | Lists the mail aliases in the namespace and all the workstations that belong to them. |
| **mail.byaddr** | Lists the mail aliases in the namespace, using the address as the key. |
| **netgroup** | Contains netgroup information, using the group name as the key. |
| **netgroup.byhost** | Contains information about the netgroups in the namespace, using workstation names as the key. |
| **netgroup.byuser** | Contains netgroup information, using the user as the key. |

*Table 1. NIS Maps (continued)*

| NIS Map | Description |
|---|---|
| **netid.byname** | Contains the Secure remote procedure call (RPC) netname of workstations and users, along with their user IDs and group IDs. |
| **netmasks.byaddr** | Contains network masks used with Internet Protocol (IP) subnetting, using the address as the key. |
| **networks.byaddr** | Contains the names and addresses of the networks in the namespace, and their Internet addresses. |
| **networks.byname** | Contains the names and addresses of the networks in the namespace, using the names as the key. |
| **passwd.byname** | Contains password information, with the username as the key. |
| **passwd.byuid** | Contains password information, with the user ID as the key. |
| **protocols.byname** | Lists the network protocols used. |
| **protocols.bynumber** | Lists the network protocols used but uses their number as the key. |
| **publickey.byname** | Contains public and secret keys for Secure RPC. |
| **rpc.bynumber** | Lists the known program name and number of RPCs. |
| **services.byname** | Lists the available Internet services. |
| **ypservers** | Lists the NIS servers in the namespace, along with their IP addresses. |

"Network Information Service" on page 3 contains detail about the development and use of NIS.

## Network Information Service

This provides information on the Network Information Service (NIS), which is installed as part of the Network File System (NFS).

See "Troubleshooting NIS" on page 31 for information on diagnosing and resolving NIS-related problems.

### NIS Overview

Network Information Service (NIS) is a distributed database that allows you to maintain consistent configuration files throughout your network. NIS is the current name for the service originally known as *Yellow Pages* (YP). NIS and YP are functionally identical.

NIS is a part of the Network File System (NFS) software package that includes commands and daemons for NFS, NIS, and other services. Although NFS and NIS are installed together as one package, each is independent and each is configured and administered individually. For information on how NFS works with the operating system see the *Networks and communication management*. You should obtain a copy of the book *Managing NFS and NIS*.

**Note:** If the file **/var/yp/securenets** exists, the server only provides NIS services to the hosts within the Internet Protocol (IP) range specified.

### Components of NIS

The NIS environment is composed of *clients* and *servers* logically grouped together in a *domain*.

Each domain has a particular set of characteristics. These characteristics are defined in *maps,* or databases, that specify certain system information such as user names, passwords, and host names. Each of these components is discussed in detail below.

## Servers

An NIS *server* is a host that provides configuration information to other hosts on the network.

Servers retain a set of maps and run the **ypserv** daemon, which processes requests from clients for information contained in maps. There are two types of servers: a *master* server and a *worker* server.

### *Master Servers*

A *master* server is the single host in a particular domain that maintains the authoritative maps.

The master server runs **ypupdated** daemon, which prompts worker servers to update their copies of the maps (all other hosts in the domain must obtain their map information from the master server, either directly or indirectly). The master server also runs the **yppasswdd** daemon, which processes requests to change users' passwords. Recommended characteristics of the master server include:

- Accessible by the system administrator. If something goes wrong, or if updates need to be made, it is easy to reach the master server.
- Stable. The master server usually stays active for long periods of time. It is stable so systems that depend on it can rely on uninterrupted service.
- Accessible from the network. Although networks can be complex with the presence of many gateways or bridges, the master server is accessible from most systems on the network.

For a small number of hosts, each host can access the master server directly. However, for a larger number of hosts in a domain, the master server can become overloaded. To balance the NIS processing load and provide services when the master server is unavailable, additional hosts can be designated as worker servers.

### *Worker Servers*

NIS *worker* servers act as intermediaries between clients and the master server by keeping exact replicas of the master server's maps.

**Note:** NIS worker servers are conventionally known as NIS slave servers. These terms are used interchangeably to use more inclusive language.

All changes to the maps are made on the master server. Then, the changes are propagated from the master server to the worker servers. Once a worker server is added to the domain, it is able to answer the same queries that the master is able to answer. In this way, worker servers can help with extra load on the master server without violating the authority of the master server.

Worker servers also act as a backup in case the master server or the network fails. A client requesting information waits until a server responds. This waiting time varies depending on the reason the server is unreachable. Adding worker servers increases the availability of information even if the master server is unavailable.

Normally, there should be at least one worker server for each domain. The number of worker servers in a domain should be balanced to achieve the desired level of availability and response time without adding the expense of copying data to too many systems.

## Clients

NIS *clients* make up the majority of hosts in a NIS domain.

Clients run the **ypbind** daemon, which enables client processes to obtain information from a server. Clients do not maintain maps themselves, but rather query servers for system and user account information. (Clients do not make a distinction between querying the master server or a worker server.) To access system information contained in an map, a client makes a Remote Procedure Call (RPC) to a server. The server searches its local database and returns the requested information to the client. (See *Communications Programming Concepts* for detailed information about RPCs.)

NIS clients locate the server by broadcasting on the networks that are directly connected to the client machine. Since these broadcast messages are not forwarded by network gateways, if there is no NIS server that can be reached without using a network gateway, the client must specify a server when starting the **ypbind** daemon.

Note that every request for system information requires a server contact, and the speed of your network can affect the response time. Although a local retrieval is usually faster than a network retrieval, the benefits of NIS outweigh the compromise in access time.

## NIS Domain

An NIS domain is a collection of systems that are logically grouped together.

A group of hosts that share the same set of NIS maps belong to the same domain. The hosts are usually grouped together in the domain for a common reason; for example, when working in the same group at a particular location. Each NIS host is assigned to a domain when the system starts. The domain name must be set on all hosts that intend to use NIS.

There is one master server per NIS domain, and the systems in the domain are typically on the same network. However, access to data served by NIS is independent of the relative locations of an NIS client and server. All systems within the NIS domain use the master server to retrieve system information, and the number of systems in a domain must be limited for the sake of efficiency. As the number of systems grows, the response time from the master server increases because of the increased workload. By design, you cannot add another master server to a domain because there would be two authoritative sources for the maps. To reduce master server load, you can add worker servers to the domain, or define more than one domain. Each new domain, of course, has its own master server.

## NIS Maps

NIS *maps* are databases that specify certain system information such as user names, passwords, and host names, in a database format called *DBM* (Database Management).

Each map is constructed from a standard text file by associating an index *key* with a *value*. For example, the information in the master server's **/etc/hosts** file is used to create a map that uses each host name as a key, and the IP address as the value. The key and value pairs (also known as *records*) that are created from the entries in the**/etc/hosts** file comprise the *hosts.byname* map.

**Attention:** An NIS record has a *maximum size* of 1024 bytes. This limitation applies to all NIS map files. For example, a list of users in a group can contain a *maximum* of 1024 characters in single-byte character set file format. NIS cannot operate correctly with map files that exceed this maximum

The most commonly used maps have *nicknames* that some commands can translate into map names. For instance, when you enter:

```
ypcat hosts
```

The output you receive is actually the contents of the *hosts.byname* map, because there is no map called `hosts` in the NIS database. (The **ypcat -x** command produces a list of available nicknames.)

By default, the maps listed in the following table are created if their corresponding files are available on the master server:

| **Map** | **Nickname** | **File** |
|---|---|---|
| **passwd.byname** | passwd | **/etc/passwd** |
| **passwd.byuid** | | |
| **group.byname** | group | **/etc/group** |
| **group.bygid** | | |
| **hosts.byaddr** | hosts | **/etc/hosts** |
| **hosts.byname** | | |

| Map | Nickname | File |
|---|---|---|
| **ethers.byaddr** | ethers | **/etc/ethers** |
| **ethers.byname** | | |
| **networks.byaddr** | networks | **/etc/networks** |
| **networks.byname** | | |
| **rpc.bynumber** | | **/etc/rpc** |
| **services.byname** | services | **/etc/services** |
| **protocols.byname** | protocols | **/etc/protocols** |
| **protocols.bynumber** | | |
| **netgroup** | | **/etc/netgroup** |
| **netgroup.byhost** | | |
| **netgroup.byuser** | | |
| **bootparams** | | **/etc/bootparams** |
| **mail.aliases** | aliases | **/etc/aliases** |
| **mail.byaddr** | | |
| **publickey.byname** | | **/etc/publickey** |
| **netid.byname** | | **/etc/passwd** |
| | | **/etc/group** |
| | | **/etc/hosts** |
| | | **/etc/netid** |
| **netmasks.byaddr** | | **/etc/netmasks** |
| **ypservers** | | |

## ypservers: a Special Map

Notice that no file corresponds to the **ypservers** map. **ypservers** is a special map that contains the names of the NIS servers, both worker and master, in the domain. Clients use the **ypservers** map to find the nearest available server. The master server refers to it to determine the names of the worker servers that need to obtain updated copies of the NIS maps. Information about specifying the input to the **ypservers** map is discussed in "Configuring the NIS Master Server" on page 8 and in "Adding a New NIS Worker Server" on page 20.

## Netgroups: Network-Wide Groups of Machines and Users

The `/etc/netgroup` file is not a standard Transmission Control Protocol/Internet Protocol (TCP/IP) file. Rather, it is strictly an NIS file that resides on the master server. NIS uses the **/etc/netgroup** file to generate the **netgroup.byuser** and **netgroup.byhost** maps. NIS provides these maps for authentication purposes during login, remote login, remote mount, and remote shell processes.

Specifically, the programs that consult these maps are:

**login command**
: Consults the maps for user classifications if it encounters netgroup names in the **/etc/passwd** file.

**rlogin command**
: Consults the maps for machine classifications if it encounters **netgroup** names in the /etc/exports file.

**rlogin command and rsh command**
Consult the **netgroup** map for both machine and user classifications if they encounter **netgroup** names in the **/etc/hosts.equiv** or **/.rhosts** files.

**mountd daemon**
Consults the maps for machine classifications if it encounters **netgroup** names in the /etc/exports file.

### makedbm and Makefile: Creating Maps

NIS maps are created by the **makedbm** command, converting text files into DBM format files. To simplify maintaining your maps, NIS provides a *makefile* for use with the **make** command. The default makefile (**/var/yp/Makefile**) contains all the instructions necessary to create all the default maps. You can add stanzas to **/var/yp/Makefile** to create additional maps. However, the default makefile is sufficient to address the basic needs of most NIS installations.

When the **makedbm** command generates an NIS map, it creates two files: *map.key***.pag** and *map.key***.dir**. For example, the *host.byname* map consists of the **hosts.byname.pag** and **hosts.byname.dir** files. The file with the **.pag** extension contains the key and value pairs, while the file with the **.dir** extension is the index for the **.pag** file. All the maps for a domain are stored on the servers in a subdirectory of the **/var/yp** directory. The subdirectory has the same name as the domain. For example, maps for the `literature` domain are located in the **/var/yp/literature** subdirectory.

## Maintaining Consistent System Information with NIS

NIS maintains consistent system information throughout the domain by designating one system, the master server, as the sole source of information. All the other hosts, whether they are worker servers or clients, obtain their system information from the master server.

Clients obtain their information on an as-needed basis. When a client needs a piece of system information, such as an entry from the **/etc/passwd** file, it sends a request to a server. If the information exists, the server responds with the information. Since the client obtains the information only as needed, the system information at the client remains consistent with the server.

Worker servers, on the other hand, obtain a complete copy of the maps periodically from the master server. To ensure that the system information is consistent at the worker servers, and therefore throughout the whole domain, make all updates to the maps at the master server. Then, propagate the new maps to the worker servers. To propagate a map means to copy it from the master server to all the worker servers. Propagation eliminates the need to update each map individually. In addition, propagation ensures that all copies of the database are exactly the same; therefore, any server can respond to a client's request.

After you update a map on the master server, there are three ways to propagate the new map:

1. Maps propagate automatically every few minutes if the master server is running the **ypupdated** daemon.
2. If you stop and restart NIS on the master server, all the maps propagate to the worker servers.
3. If you enter the **yppush** command at the master server, the changes propagate. The **yppush** command notifies all worker servers that a map must be transferred. The **ypserv** daemon on each worker server runs the **ypxfr** command to get the updated map. A worker server that is out of service when you enter the **yppush** command retains the earlier version of the map when it returns to the network. To prevent such situations, use the **cron** daemon to set each worker server to request updated maps from the master server at regular intervals.

## NIS Installation and Configuration

For information on installing the Network Information Service (NIS) and the Network File System (NFS), see the *Installation and migration*.

### Configuring NIS

For each NIS domain you want to configure on your network, do the following:

1. Decide which hosts on your network you want to include in this domain. Choose a domain name for the domain and make a note of it for use later in the configuration process.
2. Choose a host that has the characteristics described in "Master Servers" on page 4. Then follow the instructions in "Configuring the NIS Master Server" on page 8.
3. Decide which hosts, if any, will act as worker servers. Then, for each worker server, follow the instructions in "Configuring an NIS Worker Server" on page 10.
4. Decide which hosts will be clients in this domain. Then, for each client, follow the instructions in "Configuring an NIS Client" on page 11.

**Note:**

1. If you want non-root users to be able to log into a server, you must configure the server as a NIS client as well.
2. If the file **/var/yp/securenets** exists, the server only provides NIS services to the hosts within the Internet Protocol (IP) range specified.

### *Setting the NIS Domain Name*

To set the NIS domain name of a host (whether client or server), use the Web-based System Manager or use one of the following procedures.

- Using the System Management Interface Tool (SMIT):
  1. Enter the fast path: `smit chypdom`
  2. Enter the domain name in the `Domain name of this host` field.
  3. Specify **both** in the `CHANGE domain name take effect...` field.
  4. Accept your changes and exit SMIT. The NIS domain name is now set.
- Using the command line, enter: `chypdom -B newdomainname`

Each of these methods perform two actions. First, they run the **domainname** command, setting the NIS domain name. Second, they modify the **/etc/rc.nfs** file so that the NIS domain name is set when the system restarts.

### *Configuring the NIS Master Server*

**Attention:** An NIS record has a *maximum size* of 1024 bytes. This limitation applies to all NIS map files. For example, a list of users in a group can contain a *maximum* of 1024 characters in single-byte character set file format. Before doing the following procedure, ensure that no configuration file is beyond this limit. NIS cannot operate correctly with map files that exceed this maximum.

To configure an NIS master server, do the following tasks on the master server host:

1. Follow the instructions in "Preparing a Host for NIS Configuration " on page 12.
2. Set the domain name by following the instructions in "Setting the NIS Domain Name" on page 8.
3. Decide what information you want to manage using NIS. By default, you manage all the information contained in the files listed in "NIS Maps" on page 5. You may want to customize how you manage users, groups, and host names, especially if you have already configured a domain name server. To do so, follow the instructions in "Customizing NIS Map Input" on page 13.

You will now create the directory for this domain, build the NIS maps, and start the NIS daemons. Use the Web-based System Manager or use one of the following procedures.

- Using SMIT, enter: `smit mkmaster`.
  - Specify in the `HOSTS that will be slave servers` field the names of the hosts, if any, that you want to act as worker servers.
  - Specify **yes** in the fields `Can existing MAPS for the domain be overwritten?` and `EXIT on errors, when creating master server?` because you will want to know if an error occurs.

- If you want to configure your NIS domain for secure Remote Procedure Call (RPC) networking, specify **yes** in the `START the yppasswdd daemon?` and `START the ypupdated daemon?` fields. You should also configure secure NFS by following the instructions in *Networks and communication management*.
- Specify **yes** in the `START the ypbind daemon?` field to configure the master server to use the NIS databases.
- Specify **both** in the `START the master server...` field.
- Accept your changes and exit SMIT.

  The system takes a few minutes to perform several tasks. First, it runs the **ypinit** command. If the **ypinit** command exits successfully, the system uncomments the entries in the **/etc/rc.nfs** file for the daemons to which you specified **yes** above. Finally, the system starts these daemons.

  The **ypinit** command is a shell script that performs two tasks. First, it creates the directory **/var/yp/***domainname*, where *domainname* is the domain name you defined above. Second, it runs the **make** command on the **/var/yp/Makefile**, which creates all the NIS maps specified in the **/var/yp/Makefile**.

- Using the command line:
  1. Enter the **ypinit -m** command. This command prompts you for various information, including the names of any worker servers, and takes a few minutes to complete.
  2. Start the **ypserv** and **ypbind** daemons (and the **yppasswdd** and **ypupdated** daemons if you want) by following the instructions in "Starting and Stopping NIS Daemons" on page 14.
  3. Edit the **/etc/rc.nfs** file and uncomment the lines that use the **startsrc** commands to start these daemons (delete the pound signs at the beginning of each line). For example, if the original lines look like the following:

     ```
     #if [ -x /usr/etc/ypserv -a -d /etc/yp/`domainname` ]; then
     #       startsrc -s ypserv
     #fi
     ```

     Remove the pound signs so the file looks like:

     ```
     if [ -x /usr/etc/ypserv -a -d /etc/yp/`domainname` ]; then
            startsrc -s ypserv
     fi
     ```

*Further Considerations When Using the yppasswd Daemon*

If you chose to use a password file other than **/etc/passwd** to build the **passwd** map (see "Customizing NIS Map Input" on page 13), you must specify to the **yppasswdd** daemon the path to that file. By default, the **yppasswdd** daemon changes passwords for entries in the **/etc/passwd** file. To change the default password file to another file, do the following:

1. Edit the **/etc/rc.nfs** file, and locate the following stanza:

   ```
   #Uncomment the following lines to start up the NIS
   #yppasswd daemon.
   DIR=/etc
   if [ -x /usr/etc/rpc.yppasswdd -a -f $DIR/passwd ]; then
           start rpc.yppasswdd /usr/lib/netsvc/yp/rpc.yppasswdd
           /etc/passwd ~m
   fi
   ```

2. Change the `DIR` statement so that it specifies the path to your alternate passwd file. For example, if you use the **/var/yp/passwd** file, the `DIR` statement should look like:

   ```
   DIR=/var/yp
   ```

3. Save the file and exit the editor.
4. Enter the following three commands:

```
stopsrc -s yppasswdd
chssys -s yppasswdd -a '/var/yp/passwd -m passwd'
startsrc -s yppasswdd
```

The **yppasswdd** daemon will now use your alternate password file.

### *Configuring an NIS Worker Server*

After configuring the master server (see "Configuring the NIS Master Server" on page 8), you must decide which hosts are to act as worker servers. Worker servers keep exact replicas of the master server's maps and share the processing burden by answering queries when the master server is busy or unavailable. The following procedure must be done for each worker server.

**Note:** NIS worker servers are conventionally known as NIS slave servers. These terms are used interchangeably to use more inclusive language.

*Prerequisites*

The NIS master server is configured.

*Procedure*

To configure an NIS worker server, do the following tasks on the worker server host:

**Note:**

1. If you are configuring a worker server that is not on the same IP network, you must configure the new server as an NIS client first (see "Configuring an NIS Client" on page 11). Create the file **/var/yp/binding/<*domain_name*>/ypservers** to contain the NIS master to bind to. This file should just contain the IP address of the NIS master. You can also use the **ypset** command to explicitly point the new server to the NIS master. For example, you could use `ypset 129.23.22.1`, where `129.23.22.1` is the IP address of the master server.
2. When using subnets, a worker server must be configured on each subnet that has NIS clients for the given NIS domain. This allows clients to bind at startup and provides a fall back if the master goes down for any reason.

1. Follow the instructions in "Preparing a Host for NIS Configuration " on page 12.
2. Set the domain name by following the instructions in "Setting the NIS Domain Name" on page 8.

You will now create the directory for this domain, start the NIS daemons, and obtain copies of the NIS maps from the master server. Use the Web-based System Manager or use one of the following procedures.

- Using SMIT:
  1. Enter the fast path: `smit mkslave`.
  2. Specify the hostname of the master server for this domain in the `HOSTNAME of the master server` field.
  3. Specify **yes** in the fields `Can existing MAPS for the domain be overwritten?` and `Quit if errors are encountered?` because you will want to know if an error occurs.
  4. Specify **both** in the `START the slave server...` field.
  5. Accept your changes and exit SMIT.

     The system takes a few minutes to perform several tasks. First, it runs the **ypinit** command. If the **ypinit** command exits successfully, the system uncomments the entries in the **/etc/rc.nfs** file for the **ypserv** and **ypbind** daemons. Finally, the system starts these daemons.

     The **ypinit** command is a shell script that performs two tasks. First, it creates the directory **/var/yp/***domainname*, where *domainname* is the domain name you defined above. Second, it runs the **ypxfr** command to obtain the NIS maps from the master server.

**Note:** If this NIS worker server is not on same IP network as the NIS master server (that is, a gateway router is positioned between the worker server and the master server), you must explicitly identify the NIS master server by using the **ypset** command. For example, enter the command:

```
ypset 129.23.22.1
```

where `129.23.22.1` is the IP address of the NIS master server.

- Using the command line:

1. Start the **ypbind** daemon by following the instructions in "Starting and Stopping NIS Daemons" on page 14 to bind to the master server.
2. Enter the **ypinit -s** *mastername* command, where *mastername* is the host name of the master server. This command prompts you for various information and takes a few minutes to complete.
3. Start the **ypserv** and **ypbind** daemons by following the instructions in "Starting and Stopping NIS Daemons" on page 14.

   **Note:** If this NIS worker server is not on same IP network as the NIS master server (that is, a gateway router is positioned between the worker server and the master server), you must explicitly identify the NIS master server by using the **ypset** command. For example, enter the command:

   ```
   ypset 129.23.22.1
   ```

   where `129.23.22.1` is the IP address of the NIS master server.
4. Edit the **/etc/rc.nfs** file and uncomment the lines that use the **startsrc** commands to start these daemons. Delete the pound signs in the following example:

   ```
   #if [ -x /usr/etc/ypserv -a -d /etc/yp/`domainname` ]; then
   #       startsrc -s ypserv
   #fi
   ```

   so it looks like:

   ```
   if [ -x /usr/etc/ypserv -a -d /etc/yp/`domainname` ]; then
          startsrc -s ypserv
   fi
   ```

**Note:** If NIS users need to log into an NIS worker server, the worker server must also be configured as a client, and should have the following line as the last line in its **/etc/passwd** file:

```
+::::::
```

### *Configuring an NIS Client*

NIS clients make up the majority of hosts in an NIS domain. Clients do not maintain maps, but rather query servers for information. (Clients do not distinguish between master and worker servers.) If you are configuring a worker server that is not on the same IP network as the master server, you must configure the new server as an NIS client first.

*Prerequisites*

The NIS master server must be configured. For more information, see "Configuring the NIS Master Server" on page 8.

*Procedure*

To configure an NIS client, do the following tasks on the client host:

1. Follow the instructions in "Preparing a Host for NIS Configuration " on page 12.
2. Set the domain name by following the instructions in "Setting the NIS Domain Name" on page 8.

You then start the client using NIS. Use the Web-based System Manager or use one of the following procedures.

- Using SMIT:
  1. Enter the fast path: `smit mkclient`.
  2. Specify **both** in the **START the NIS client...** field.
  3. Accept your changes and exit SMIT.

     The system performs two tasks. First, it starts the **ypbind** daemon. Second, it uncomments the entry in the **/etc/rc.nfs** file for the **ypbind** daemon.
  4. Follow the instructions in "Setting Up NIS Client Files to Use NIS Services" on page 15.
- Using the command line:
  1. Start the **ypbind** daemon by following the instructions in "Starting and Stopping NIS Daemons" on page 14.
  2. Edit the **/etc/rc.nfs** file and uncomment the lines that use the **startsrc** command to start this daemon. Specifically, delete the pound signs in the following example:

     ```
     #if [ -x /usr/etc/ypbind ]; then
     #       startsrc -s ypbind
     #fi
     ```

     so it looks like:

     ```
     if [ -x /usr/etc/ypbind ]; then
            startsrc -s ypbind
     fi
     ```

## Preparing a Host for NIS Configuration

Before you configure NIS on a master server, worker server, or client, do the following:

1. Verify that the PATH variable in the **/.profile** file includes the **/usr/sbin** directory where the NIS commands reside.
2. Verify that Transmission Control Protocol/Internet Protocol (TCP/IP) is running by entering the command:

   ```
   lssrc -s inetd
   ```

   A message similar to the following displays:

   ```
   Subsystem         Group            PID     Status
    inetd            tcpip            4923    active
   ```

   If the status does not indicate *active*, follow the instructions in Configuring the inetd Daemon for starting the **inetd** daemon.
3. Verify that the **portmap** daemon is running by entering the command:

   ```
   lssrc -s portmap
   ```

   A message similar to the following displays:

   ```
   Subsystem         Group            PID     Status
    portmap          portmap          14003   active
   ```

   If the status does not indicate *active*, enter the command:

   ```
   startsrc -s portmap
   ```

You are now ready to configure NIS on this host. If you are configuring a master server, continue with the following section, "Customizing NIS Map Input" on page 13. If you are configuring a client or worker server, continue with "Starting and Stopping NIS Daemons" on page 14.

### *Customizing NIS Map Input*

The most common customizations made to NIS involve users, groups, and host names. However, you can customize any of the information managed by NIS. Although this discussion focuses on users, groups, and host names, you can use the same techniques to customize input to other maps.

**Note:** Perform all of these instructions on the master server host.

*Users and Groups*

**Attention:** An NIS record has a *maximum size* of 1024 bytes. This limitation applies to all NIS map files. For example, a list of users in a group can contain a *maximum* of 1024 characters in single-byte character set file format. Before doing the following procedure, ensure that no configuration file is beyond this limit. NIS cannot operate correctly with map files that exceed this maximum.

By default, NIS uses the **/etc/passwd** and **/etc/group** files on the master server as the input for the **passwd** and **group** maps. All users and groups on the master server are thus included automatically in the maps. The simplest configuration is to add every user and group in this entire domain to the **/etc/passwd** and **/etc/group** files.

**Note:** It is possible to manage users and groups without using NIS; however, managing users and groups is the primary benefit of NIS.

Either for security, accounting, or performance reasons, you may not want certain users to log into the master server. If so, you can build the **passwd** and **group** maps from other files, such as **/var/yp/passwd** and **/var/yp/group**, that are for NIS users and groups only. With this, **/etc/passwd** and **/etc/group** can contain only the minimum necessary entries. (Using a separate password file also affects the **yppasswdd** daemon. See "Configuring the NIS Master Server" on page 8 for more information.) To configure the master server in this way, do the following:

1. Create the new file to be used instead of the **/etc/group** file (for example, assume that you name the file **/var/yp/group**) by entering the following command:

   ```
   cp /etc/group /var/yp/group
   ```

   You can use a copy of any machine's **/etc/group** file, not just the **/etc/group** file on the master server. Then, using an editor, remove from the **/etc/group** file all the non-local entries, and add the NIS escape sequence (`+:`) as the last line in the file.
2. Create the new file to be used instead of the **/etc/passwd** file (for example, assume that you name the file **/var/yp/passwd**). Again, you can use a copy of any machine's **/etc/passwd** file, not just the **/etc/passwd** file on the master server. Also, you can use the password information from another NIS domain by entering `ypcat passwd > passwd` at the command line of a client in the other domain. Then, copy the **passwd** file into the **/var/yp** directory of the master server in this domain.

   You can either preserve the current passwords or reset the passwords.

   - If you want to preserve existing passwords, use the **mrgpwd** command to merge the **/etc/passwd** file with the **/etc/security/passwd** file, where the encrypted passwords are stored. This step is actually two commands, as shown below:

     ```
     cd /var/yp
     /usr/sbin/mrgpwd > passwd
     ```

     **mrgpwd** takes its input from the **/etc/passwd** and **/etc/security/passwd** files only.
   - If you want to reset all the passwords, enter the following command:

     ```
     cp /etc/passwd /var/yp/passwd
     ```

     Then, using an editor, remove the ! (exclamation point) from the password field in each entry in the **/var/yp/passwd** file. Finally, using an editor, remove from the **/etc/passwd** file all the non-local entries, and add the NIS escape sequence (`+::0:0:::`) as the last line in the file.

**Note:** User IDs (UIDs) created in this way initially contain no passwords.

3. Change the **/var/yp/Makefile** file to reflect the new locations of the input files. You can do so using one of two methods:

   - Locate only the **/etc/passwd** and **/etc/group** files in **/var/yp**. Using an editor, open the **/var/yp/Makefile** file and create a new variable called `PWDIR=/var/yp`. In the **passwd.time** and **group.time** stanzas, replace every occurrence of the `DIR` variable with `PWDIR`.
   - Locate all the **/etc** files in **/var/yp**. Edit the **Makefile** file to modify the default `DIR` variable. Change `DIR=/etc` (the default configuration) to `DIR=/var/yp`. In contrast to the first method, you do not have to edit any of the **Makefile** stanzas.

     **Note:** The SMIT fast paths **smit mkuser** and **smit mkgroup** can be used to create users and groups only in the **/etc/passwd** and **/etc/group** files.

**Note:** As the number of groups managed by NIS increases, it becomes more important to ensure that the `netid.byname` map contains an entry for each user. This can help improve the performance of the NIS servers by reducing the number of lookups required in the group maps. The `netid.byname` map can be queried by running **ypcat netid.byname**. For more information on creating the `netid.byname` map, see the **mknetid** command.

*Host Names*

By default, NIS only uses the **/etc/hosts** file to build the **hosts** map. If you have configured a domain name server in your network, you can configure NIS to include domain name system (DNS) information as well as **/etc/hosts** information in the **hosts** map. (Including DNS information in the **hosts** map eliminates re-entering all this information in the **/etc/hosts** file.) To do so, use an editor to change the **/var/yp/Makefile** file as follows:

1. Locate the **hosts.time** stanza in the **/var/yp/Makefile** file.
2. Change the two lines containing the word **MAKEDBM**:

   ```
   ...
   | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/hosts.byname; \
   ...
   | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/hosts.byaddr; \
   ...
   ```

   so that they look like:

   ```
   ...
   | $(MAKEDBM) -b - $(YPDBDIR)/$(DOM)/hosts.byname; \
   ...
   | $(MAKEDBM) -b - $(YPDBDIR)/$(DOM)/hosts.byaddr; \
   ...
   ```

   In other words, add the **-b** flag, with a space before and after, to both lines.

The **ypserv** and **ypxfrd** daemons use the file **/var/yp/securenets**, if it exists, and only respond to the IP addresses listed in the *netmask netaddr* pairs within that file.

## Starting and Stopping NIS Daemons

Use Web-based System Manager, SMIT, or the following procedure to start or stop NIS daemons.

**Prerequisites**

1. NFS must be installed on your system.
2. The **portmap** daemon must be active. To check this, use the following command:

   ```
   lssrc -s portmap
   ```

The result looks similar to the following:

```
Subsystem         Group          PID          Status
portmap          portmap        4388         active
```

**Procedure**

The five NIS daemons are controlled by the System Resource Controller (SRC). As illustrated in the following table, four of the daemons have the SRC group name **yp**:

| Daemon Name | Subsystem Name | Group Name |
|---|---|---|
| **keyserv** | keyserv | keyserv |
| **ypbind** | ypbind | yp |
| **yppasswdd** | yppasswdd | yp |
| **ypserv** | ypserv | yp |
| **ypupdated** | ypupdated | yp |

To start or stop NIS daemons, use the Web-based System Manager or use one of the following procedures.

- Using SMIT:
  - Enter the SMIT **smit ypstartstop** fast path.

    Select a menu option, depending on whether you want to start or stop the **ypserv**, **ypbind**, **yppasswdd**, or **ypupdated** daemon. Once you make your selection, the daemon you specified will be started or stopped.
  - To start the **keyserv** daemon, use the SMIT **smit mkkeyserv** fast path.
  - To stop the **keyserv** daemon, use the SMIT **smit rmkeyserv** fast path.
- Using the command line to start or stop the NIS daemons, run the **startsrc** or **stopsrc** command. You can start or stop the daemons individually or as a group. For example:
  - To start all the daemons (not including the **keyserv** daemon), enter:

    ```
    startsrc -g yp
    ```
  - To stop all the daemons, enter:

    ```
    stopsrc -g yp
    ```
  - To start a single daemon, enter:

    ```
    startsrc -s daemon_name
    ```
  - To stop a single daemon, enter:

    ```
    stopsrc -s daemon_name
    ```

## Setting Up NIS Client Files to Use NIS Services

In this procedure, you specify which NIS maps that this client will use by adding a special NIS marker to various system files. In general, the system configuration files on an NIS client should have a minimum number of entries because the client should rely primarily on a server for its information. However, you may want to configure a few local entries that you do not want defined throughout the entire domain.

Actually, NIS handles client configuration files in two ways. Some configuration files are completely ignored once the **ypbind** daemon starts, and other files are appended to. If NIS ignores a particular file, the client will only know what its server's map contains. If NIS appends map information to a file, the client can use local information that no other host knows as well as NIS map information.

### *Files that NIS Ignores*

Once the **ypbind** daemon is running, the client relies solely on the following NIS maps instead of their corresponding files:

*Table 2. NIS maps*

| Map | Nickname | File |
|---|---|---|
| *hosts.byaddr* | hosts | **/etc/hosts** |
| *hosts.byname* | | |
| *ethers.byaddr* | ethers | **/etc/ethers** |
| *ethers.byname* | | |
| *networks.byaddr* | networks | **/etc/networks** |
| *networks.byname* | | |
| *rpc.bynumber* | | **/etc/rpc** |
| *services.byname* | services | **/etc/services** |
| *protocols.byname* | protocols | **/etc/protocols** |
| *protocols.bynumber* | | |
| *netgroup* | | **/etc/netgroup** |
| *netgroup.byhost* | | |
| *netgroup.byuser* | | |
| *publickey.byname* | | **/etc/publickey** |
| *netid.byname* | | **/etc/passwd** |
| | | **/etc/group** |
| | | **/etc/hosts** |
| | | **/etc/netid** |
| *netmasks.byaddr* | | **/etc/netmasks** |

You do not need to perform any configuration on the above files in order to use their corresponding NIS maps; the **ypbind** daemon does this automatically. However, the **/etc/hosts** file should have entries for the local loopback name and client's host name. Use either an editor or the **smit hosts** fast path to verify that the **/etc/hosts** file has these entries. For example, the client's **/etc/hosts** file should look similar to this example:

```
127.1         localhost   # local loopback name
200.10.2.101  zepher      # client's host name
```

The **/etc/hosts** file is accessed at boot time before NIS is available. After the system is running and the **ypbind** daemon is started, NIS ignores the **/etc/hosts** file.

### *Files where NIS Appends Map Information*

Each of the following subheadings explains how to configure a client's configuration files to use a particular NIS map. You may choose to use all the available maps, or only a few. Many NIS installations use all the available maps, especially the **passwd** and **group** maps.

*passwd.byname and passwd.byuid map*

These two maps together are referred to by the nickname **passwd**. Using either an editor or the **smit lsuser** fast path, verify that the **/etc/passwd** file contains entries for the root user and the other primary users on the machine (in other words, the entries supplied in the default **/etc/passwd** file). Then, using an editor, add the NIS escape entry, + (plus sign), to enable the NIS password service. For example, the client's **/etc/passwd** file should look similar to the following:

```
root:!.k:0:1:/:/usr/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/usr/bin/csh
bin:*:3:3::/usr/bin:
uucp:*:4:4::/var/spool/uucppublic:
news:*:6:6::/var/spool/news:/usr/bin/csh
+::0:0:::
```

The NIS entry (the last line) instructs library routines to use the NIS password service after examining the local entries. So, when a program examines the **/etc/passwd** file, it first finds the local entries, and then it requests that NIS provide the password information.

In addition to using the entire **passwd** map, you can explicitly include (with a plus entry) and exclude (with a minus entry) NIS password information about specific users and groups.

The following are the types of + (plus) and - (minus) entries that you may define:

- A + (plus) by itself means to include the entire contents of the NIS **passwd** map.
- A + (plus) with a name means to include that name from the NIS map.
- A + (plus) followed by a @ and a netgroup (that is `+@netgroup_name`) means to insert the entries for all the members of the netgroup `netgroup_name` at that point.
- The - (minus) entries mean exclude the user or netgroup specified.

If the + (plus) entry contains data in one of the colon-separated fields (except for the user ID, or UID, and group ID, or GID, fields) of the password entry, that data overrides what is in the NIS map. Also, earlier entries in the file take precedence over later entries with the same user name or user ID. The following are some examples:

To remove the NIS password entry for a user, enter:

```
-user
```

To remove the NIS password for users in a netgroup, enter:

```
-@netgroup
```

The line that subtracts the netgroup or user must appear before any other **/etc/passwd** file entry that includes the netgroup or user. For example, to remove password entries for user `cliffc` and users in the `bad-users` netgroup, the password file entry must contain the user name, UID, and GID:

```
-cliffc:*:218:201::
-@bad-users
+::0:0:::
```

If user `cliffc` is a member of the `good-users` netgroup, the following example does *not* remove user `cliffc` from the **/etc/passwd** file:

```
+@good-users
-cliffc:*:218:201::
+::0:0:::
```

Once the routines that read the password's file find a match for `cliffc`, they stop parsing the file. Therefore, the `-cliffc` entry will never be found, because the `good-users` netgroup includes user `cliffc`.

*group.byname and group.bygid maps*

These two maps together are referred to by the nickname **group**. Using either an editor or the **smit lsgroup** fast path, verify that the **/etc/group** file contains entries for the system and other primary groups on the machine (in other words, the entries supplied in the default **/etc/group** file). Then, using an editor, add the NIS escape entry (+, plus sign) to enable the NIS group service. For example, the client's **/etc/group** file should look similar to the following:

```
system:!:0:root
staff:!:1:root
bin:!:2:root,bin
sys:!:3:root,su,bin,sys
adm:!:4:root,su,bin,adm
uucp:!:5:root,uucp
mail:!:6:root,su
security:!:7:root
cron:!:8:root
printq:!:9:root
audit:!:10:root
+:
```

*mail.aliases and mail.byaddr maps*

These two maps together are referred to by the nickname **aliases**. To enable use of the NIS aliases mapping:

1. Uncomment `O AliasFile` in the **sendmail.cf** file and specify the map name for NIS aliases.
2. Recompile the **sendmail.cf** file with the command **sendmail -bz**.
3. Recompile the alias database with the command **sendmail -bi**.

*netgroup.byhost and netgroup.byuser maps*

As noted in Files that NIS Ignores, NIS uses these two maps automatically. However, you can configure two other system files to reference these maps, specifically, the **/etc/hosts.equiv** file and the **/.rhosts** file. Doing so can help you control remote logins more effectively.

For example, you can edit the **/etc/hosts.equiv** file and add a single line, with only the + (plus) character on it. This allows anyone to log into the machine because all further entries are retrieved from NIS rather than the local file. Or, for more control over logins, add a list of trusted hosts to the **/etc/hosts.equiv** file. For example:

```
+@trusted_group1
+@trusted_group2
-@distrusted_group
```

The names to the right of the `@` (at sign) should be netgroup names defined in the netgroup map.

You can also add a list of trusted hosts to the **/.rhosts** file. For example:

```
+@trusted_group1
+@trusted_group2
-@distrusted_group
```

Because this file controls remote root access to the local machine, unrestricted access is not recommended. You cannot use aliases for host names in the **/.rhosts**, **hosts.equiv**, or **netgroup** files, because they all enable local machines to access remote machines. You can, however, use aliases for host names in the **/etc/hosts** file.

**Note:** If none of the escape sequences are added to the **/etc/hosts.equiv** or **/.rhosts** files, NIS is not used when a program examines these files.

## NIS Maintenance

The Network Information Service (NIS) environment requires adjustments from time to time. In large or complex networks, the NIS environment may change many times a day. This section discusses how to maintain NIS.

### Prerequisite

All the tasks discussed in this section assume that NIS is installed and configured on your network. See "Configuring NIS" on page 7.

### NIS Security

The **/var/yp/securenets** file limits access when using NIS, as described in the following section.

#### */var/yp/securenets*

Both the **ypserv** and the **ypxfrd** use the **/var/yp/securenets** file and, if present, only respond to Internet Protocol (IP) addresses in the range given. This file is read-only when the daemons (both **ypserv** and **ypxfrd**) start. To cause a change in **/var/yp/securenets** to take effect, you must kill and restart the daemon (see "Starting and Stopping NIS Daemons" on page 14). The format of the file is as follows:

```
netmask netaddr
e.g.

255.255.0.0 128.30.0.0
255.255.255.0 128.311.10.0
```

The second line the netmask address is 255.255.255.0 and the network address is 128.311.10.0. This setup will only allow the **ypserv** daemon to respond to those IP addresses within the subnet 128.311.10.0 range.

### Avoiding Broadcasts

The **ypbind** daemon can be configured to try connecting to a list of servers before broadcasting. The server list is a list of internet addresses, one per line, in the **/var/yp/binding/domainname/ypservers** file. For example, if the domain `wiz.com` has servers at `10.5.1.1` and `10.5.1.2` that should be tried before broadcasting, the file **/var/yp/binding/wiz.com/ypservers** would contain the following entries:

```
10.5.1.1
10.5.1.2
```

This file is not managed by NIS, because it is used to do NIS binding; if it is used, it must be maintained on each client.

### Changing an NIS Map

Changing the NIS maps to reflect updated system information can be a common maintenance task. System information, such as a new user account or a changed password, can require constant updating. Whenever you need to modify an NIS map, do so on the master server, and then propagate the changes to the worker servers. Modifying maps on worker servers can break the NIS service algorithm, which can result in unreliable map data. The only exception to this rule is when users change their password with the **yppasswd** command. (See "Changing NIS Passwords" on page 20 for more information.)

To change an NIS map, use the Web-based System Manager on the master server. Alternatively, you can use one of the following procedures on the master server.

- Using the SMIT:
  1. Edit the text file that is used as the input file for the map.
  2. Enter the **smit mkmaps** fast path. Specify the map you changed in the `MAPS that are to be built` field.

3. Exit SMIT.

   The map is now changed, and the master server has requested that all the worker servers update their maps.

- Using the command line:
  1. Edit the text file that is used as the input file for the map.
  2. Update the map by entering the following two commands:

     ```
     cd /var/yp
     make map
     ```

     where map specifies the map to be updated, for example **hosts** or **services**.
  3. Propagate the new map by following the instructions in NIS Propagation Map.

**Note:**

- If you have modified several text files and want to confirm that all NIS maps are updated, issue the **make** command without parameters to automatically evaluate every input file on the NIS master server. If the file has been modified since the latest NIS map for that file was built, the NIS map is automatically rebuilt.
- If the file **/var/yp/securenets** exists, the server only provides NIS services to hosts within the IP range specified.

## Changing NIS Passwords

Users can change their password using any of three methods. If the user is logged on to the master server, and you use the /etc/passwd file on the master server to build the **passwd** map, the user can:

- Use the Web-based System Manager. Start it by typing **wsm** on the command line.
- Enter the SMIT fast path, `smit passwd`
- Enter the command, `passwd`

All of these methods change the user's entry in the **/etc/passwd** file. You must rebuild the **passwd** map manually (see "Changing an NIS Map" on page 19). If the **yppasswdd** daemon is running on the master server, users can change their password from any host in the domain by entering the command:

```
yppasswd
```

This command changes the user's password in the **passwd** map itself, as well as the **/etc/passwd** file, and thus requires no intervention by the system administrator. (For more information on the **yppasswdd** daemon, see "Configuring the NIS Master Server" on page 8).

## Adding a New NIS Worker Server

If your network configuration grows or changes, you may want to add additional worker servers to support the new configuration. Adding a new worker server involves modifying the **ypservers** map. The procedure for modifying the **ypservers** map differs from other maps because no text file is used as input for this map. Instead, the **makedbm** utility is used to create the modified **ypservers** maps.

**Note:** NIS worker servers are conventionally known as NIS slave servers. These terms are used interchangeably to use more inclusive language.

To add a new worker server to your network, do the following on the master server:

1. Change to the **/var/yp** directory by entering:

   ```
   cd /var/yp
   ```

   **Note:** If the file **/var/yp/securenets** exists, the server only provides NIS services to hosts within the IP range specified.
2. Enter the command:

```
(makedbm -u domain/ypservers ; echo new_server new_server) |
makedbm - tmpservers
```

where `domain` specifies the name of this NIS domain, and `new_server` specifies the name of the worker server host that you are adding to the **ypservers** map. This command lists the contents of the current **ypservers** map, and appends the name of the new worker server, and then creates a new map called **tmpservers**.

3. Verify that the new map contains the names of all the worker servers by entering the command:

   ```
   makedbm -u tmpservers
   ```

4. Replace the old **ypservers** map files with the new ones by entering the following two commands:

   ```
   mv tmpservers.pag domain/ypservers.pag
   mv tmpservers.dir domain/ypservers.dir
   ```

   where `domain` specifies the name of this NIS domain.

5. Follow the instructions in "Configuring an NIS Worker Server" on page 10.
6. If you are using the **ypservers** file to avoid broadcasts, you may want to add the new server to the file on the clients. See "Avoiding Broadcasts" on page 19.

## Adding a New NIS User

To add a new NIS user, use the Web-based System Manager on the master server. Alternatively, you can use one of the following procedures on the master server.

- Using SMIT:
  1. Enter the `smit mkuser` fast path. Enter the new NIS user's name in the `User NAME` field. This adds an entry to the **/etc/passwd** file for the new user.
  2. Exit SMIT.
  3. Enter the `smit mkmaps` fast path. Specify **passwd** in the `MAPS that are to be built` field.
  4. Exit SMIT.

     The new NIS user is now added, and the master server has requested that all the worker servers update their maps.
- Using the command line:
  1. Using an editor, add a line for the new NIS user to the **/etc/passwd** file (or whatever file you use as input to the **passwd** map). The new line should look similar to the following:

     ```
     mel::1295:325:Mel Smith:/u/mel:/bin/ksh
     ```

  2. Update the **passwd** map by entering the following two commands:

     ```
     cd /var/yp
     make passwd
     ```

  3. Propagate the new **passwd** map by following the instructions in "Propagating an NIS Map" on page 22.

## Creating Nonstandard NIS Maps

As discussed in "NIS Maps" on page 5, maps are databases that have a special format. The default, or standard, maps are built out of standard system text files. However, NIS maps are flexible in that you can build into a map any information that has a key and a value.

The following example shows how to create a nonstandard map called **udir.nam** that lists the home directories of users in this domain. You will use the **/etc/passwd** file as input file for the **udir.nam** map. (If you use a different file to create the **passwd** map, use that file instead of **/etc/passwd**.) The keys for the map are the user names, and their home directories are the corresponding values. To create this new map, enter the following two commands on the master server:

```
cd /var/yp
awk '{FS=":" ; OFS="\t" ; print $1,$6}' /etc/passwd | \
    makedbm - domain/udir.nam
```

where `domain` specifies this NIS domain, and ` is the single forward quote. The **awk** command extracts from the **/etc/passwd** file the first and sixth fields. It passes this information to the **makedbm** command that builds the **udir.nam** map. You can now propagate and use this map just like any other map. For example, enter the following command to see a list of all the home directories specified in the **/etc/passwd** file.

```
ypcat udir.nam
```

This example used the **awk** command to filter out the unwanted fields from the **/etc/passwd** file. However, you can use any system utility or programming language to create the map input. In fact, you can create the map input manually. For example, to create the **udir.nam** map, you could have entered the command:

```
makedbm - domain/udir.nam
john /u/john
mary /u/mary
sam /u/sam
<Ctrl^D>
```

Whatever method you use to create the map input file, you probably want to update the map periodically. To do so, you can add stanzas to the **/var/yp/Makefile** file so that your nonstandard map can be updated as any other map (see "Changing an NIS Map" on page 19). For detailed information on customizing the **/var/yp/Makefile**, see the make Command Overview topic in *General Programming Concepts: Writing and Debugging Programs*.

**Note:** When you add a new map after the initial set of maps have been pushed to a worker server, you must make the new maps with the **NOPUSH** option set to 1. For example, use the following command to make newmap map.

```
make NOPUSH=1 newmap
```

If you do not use the **NOPUSH** option, the **make** command suspends. (This only applies if the new map does not already exist on the worker server.) Next, use the **ypxfr** command on each worker server for each new map you created (see "Propagating an NIS Map" on page 22).

## Propagating an NIS Map

This procedure explains how to propagate a new or changed map from the master server to one or more worker servers.

### *Prerequisite*

The NIS worker servers must be authorized to copy files remotely (using the **rcp** remote copy command) from the NIS master server.

### *Procedures*

To propagate NIS maps from the master server to worker servers, use the Web-based System Manager or use one of the following procedures.

- From a worker server, you have two choices:
  - To use SMIT, enter the `smit ypxfr` fast path and specify a map name in the `Name of the MAP to be transferred` field. This action retrieves the specified map from the master server.
  - To use the command line, enter the following command:

```
ypxfr mapname
```

- From the master server, you have two choices:
  - To use SMIT, enter the `smit yppush` fast path and specify a map name or names in the `Maps to be transferred to slave` field.
  - To use the command line, enter the following command:

    ```
    yppush mapname
    ```

All of these methods use the **ypserver** database to generate a list of worker servers in your domain. The master server then sends a *transfer database* request to the **ypserv** daemon on each of the worker servers. The **ypserv** daemon on the worker server executes a copy of the **ypxfr -C** command and then passes a summary of the information it needs to identify the database and call back the initiating command.

### *Automating Map Propagation*

You may need to propagate some maps more frequently than others. For example, the **passwd** map can change many times a day and must be propagated more frequently than the **protocols** or **services** maps, which may not change for months at a time. Rather than propagate each map manually, you may want to automate the propagation of maps with the cron daemon. To do so, use the following procedure. (You will need to perform these instructions on each worker server.)

1. Group the maps together according to how often you want to propagate them. The system provides three example shell scripts that organize the maps into three groups:

   ```
   /usr/sbin/ypxfr_1perhour
   /usr/sbin/ypxfr_2perday
   /usr/sbin/ypxfr_1perday
   ```

   For example, the **/usr/sbin/ypxfr_1perhour** shell script contains **ypxfr** commands for several maps that change frequently and should, therefore, be propagated frequently. The other two shell scripts are for maps that change less frequently. Use an editor to modify these shell scripts to meet the needs of your network.
2. Next, configure the **cron** daemon to execute these scripts at the appropriate times. As the root user, enter the command:

   ```
   crontab -l > cron.tmp
   ```

   This records the current **crontab** settings in the file **cron.tmp**.
3. Edit the **cron.tmp** file and add entries for each of the scripts listed above. For example, you might add the entries:

   ```
   00 * * * * /usr/sbin/ypxfr_1perhour      # run every hour
   00 00 * * * /usr/sbin/ypxfr_2perday      # run at midnight
   00 12 * * * /usr/sbin/ypxfr_2perday      # run at noon
   00 1 * * * /usr/sbin/ypxfr_1perday       # run at 1 A.M.
   ```

   To minimize the performance impact on the master server, alter the exact time of execution of the shell scripts on each server. For example, if the first worker server has the above entries, the second worker server might run its shell scripts five minutes later by having the entries:

   ```
   5 * * * * /usr/sbin/ypxfr_1perhour      # run every hour
   5 00 * * * /usr/sbin/ypxfr_2perday      # run at 12:05 A.M.
   5 12 * * * /usr/sbin/ypxfr_2perday      # run at 12:05 P.M.
   5 1 * * * /usr/sbin/ypxfr_1perday       # run at 1:05 A.M.
   ```
4. Save the file and exit the editor.
5. Enter the command:

   ```
   crontab cron.tmp
   ```

This defines the **crontab** settings to what the **cron.tmp** file contains. (For more information on defining **crontab** settings, see the cron command.)

### *Logging Map Propagation*

Transfers and transfer attempts are logged in the **/var/yp/ypxfr.log** file on worker servers. If the file exists, logging results are appended to it.

To start logging, enter the command:

```
touch /var/yp/ypxfr.log
```

To stop logging, either enter the command:

```
mv /var/yp/ypxfr.log /var/yp/ypxfr.log.old
```

or, enter the command:

```
rm /var/yp/ypxfr.log
```

## Moving the Master Server to a Different Host

Once you have NIS configured on your network, your security, performance, or other needs may change. You may want to move the master server configuration to a different host, perhaps one that is more secure or offers greater performance. To do so, use the following procedure:

1. Copy all the map input files (not the map files themselves) from the old master server host to the new master server host. In the case of the **ypservers** map, there is no input file to copy. To create a temporary **ypservers** map input file, do the following on the old master server host:

   a. Change to the **/var/yp** directory by entering:

      ```
      cd /var/yp
      ```

   b. Enter the command:

      ```
      makedbm -u domain/ypservers > tmpservers
      ```

      where `domain` specifies the name of this NIS domain. This sends the contents of the current **ypservers** map to a file called **tmpservers**.

   c. Edit the **tmpservers** file, and change the name of the master server from the old master server host to the new master server host. For example, if the **tmpservers** file contains the line:

      ```
      YP_MASTER_NAME old_master
      ```

      change this line to:

      ```
      YP_MASTER_NAME new_master
      ```

   d. Save the file and exit the editor.

   e. Copy the **tmpservers** file to the **/var/yp** directory on the new master server host.

2. Configure the new master server host by following the instructions in "Configuring the NIS Master Server" on page 8.

3. On the new master server host, enter:

   ```
   cd /var/yp
   ```

   Then, enter the commands:

```
makedbm - tmpservers < tmpservers
mv tmpservers.pag domain/ypservers.pag
mv tmpservers.dir domain/ypservers.dir
```

4. On each of the worker servers, enter the `smit ypxfr` fast path. Specify **ypservers** in the `Name of the MAP to be transferred` field, and specify the hostname of the new master server host in the `HOSTNAME of the master server` field.
5. Exit SMIT to retrieve the updated **ypservers** map from the new master server host.
6. Propagate the rest of the newly rebuilt maps by following the instructions in "Propagating an NIS Map" on page 22.
7. If you are using the **ypservers** file to avoid broadcasts, you may need to change the internet address in the file on the clients. See "Avoiding Broadcasts" on page 19.

## NIS Automount

The use of **automount** maps involves map formats, replicated file systems, comments in maps, directory patterns, multiple mounts, included maps, maps for the **automount** command, and the **auto_master** (or, when necessary for compatibility, **auto.master**) map file.

### Map Entry Format

A simple map entry (mapping) example follows:

```
key [-mount-options] location ...
```

where

**key**
: Is the full path name of the directory to mount when used in a direct map. In an indirect map, `key` represents a simple name.

**`mount-options`**
: Is a list of options separated by commas.

**`location`**
: Specifies a file system from which the directory can be mounted. In the example case, `location` takes the form:

```
hostname:pathname
```

where *hostname* is the server name where the file system will mount and *pathname* is the directory path to mount.

### Replicated File Systems

Multiple *location* fields can be specified for replicated NFS file systems, in which case **automount** and the kernel each tries to use that information to increase availability. If the read-only flag is set in the map entry, **automount** mounts a list of locations that the kernel may use, sorted by several criteria. If a server does not respond, the kernel switches to an alternative server. Sort ordering is used by **automount** to determine how the next server is chosen. If the read-only flag is not set, **automount** mounts the best single location, chosen by the same sort ordering, and new servers are chosen only when an unmount has been possible and a remount is done. Servers on the same local subnet are given first preference, and servers on the local net are given second preference. Among servers equally far away, response times determine the order (if no weighting factors are used).

If the list includes server locations using both Versions 2 and 3 of the NFS protocol, **automount** chooses only a subset of the server locations on the list to ensure all entries are the same protocol level. It gives preference to Version 3 servers and uses Version 2 servers if Version 3 servers are unavailable.

If each location in the list shares the same path name, a single location can be used with a list of host names, separated by commas.

```
hostname,hostname...:pathname
```

### *Weighting Factor*

Requests for a server can be weighted by appending the weighting factor (as an integer within parentheses) to the server name. A weighting factor of zero is the highest priority. Servers without a weighting factor are assumed to have a weighting factor of zero. Progressively higher values decrease a server's chance of being selected. In the following example,

hosts `masterlib` and `mystery` have the highest priority; host `doyle`, the lowest.

```
man -ro masterlib,mystery,christie(1),doyle(4):/usr/man
```

**Note:** In the selection process, server proximity takes higher priority than weighting. In the previous example, if server `doyle` were on the same network as the user and the other servers were on different network segments, then `doyle` would be selected and the weighting value would be ignored.

When each server has a different export point, you can still apply weighting. For example:

```
man -ro masterlib:/usr/man mystery,christie(1):/usr/share/man doyle(3):/export/man
```

Mapping can be continued across input lines by escaping the newline character with a backslash (\). Comments begin with a pound sign (#) and end at the subsequent newline character.

## Map Key Substitution

The ampersand (&) character is expanded to the value of the key field for the entry in which it occurs. In the following example, the ampersand expands to `suspense`.

```
suspense
   masterlib:/thriller/&
```

### *Wild Card Key*

When used in the key field, the asterisk (*) is recognized as a *catch-all* entry. The asterisk matches any key not previously matched. For instance, if the following entry appears in the indirect map for **/mystery**:

```
*
  &:/masterlib/mystery/&
```

the entry would allow automatic mounts in **/mystery** of any remote file system whose location was specified as

```
hostname:/masterlib/mystery/hostname
```

## Multiple Mounts

A multiple mount entry takes the following form:

```
key [-mount-options] [ [/mountpoint] [-mount-options] location ...  ] ...
```

The initial `/[mountpoint]` is optional for the first mount and mandatory for all subsequent mounts. The optional `mountpoint` is taken as a pathname relative to the directory named by `key`. If `mountpoint` is omitted in the first occurrence, a mountpoint of / (root) is implied. The following is an example entry in the indirect map for **/src**:

```
beta -ro\

    /          svr1,svr2:/export/src/beta \
    /1.0       svr1,svr2:/export/src/beta/1.0 \
    /1.0/man   svr1,svr2:/export/src/beta/1.0/man
```

All offsets must exist on the server under `beta.automount`, which automatically mounts `/src/beta`, `/src/beta/1.0`, and `/src/beta/1.0/man`, as needed, from either `svr1` or `svr2` (whichever is nearer and responds more quickly).

The initial / (forward slash) within `/[mountpoint]` is required. The optional mount point is taken as a path name relative to the destination of the symbolic link for key. If the `mountpoint` option is omitted in the first occurrence, a mount point of / (or the root directory) is assumed.

## Other File System Types

The automounter assumes NFS mounts as a default file system type. Other file system types can be described using the **fstype** mount option. Other mount options specific to this file system type can be combined with the **fstype** option. The location field must contain information specific to the file system type. If the location field begins with a slash (/), a colon (:) must be prepended. To mount a CD file system, use the following example:

```
cdrom -fstype=cdrfs,ro      :/dev/cd0
```

To perform an **autofs** mount, use the following example:

```
src   -fstype=autofs      auto_src
```

## Indirect Maps

An indirect map allows you to specify mappings for the subdirectories you wish to mount under the directory indicated on the command line. In an indirect map, each **key** consists of a simple name that refers to one or more file systems that are to be mounted as needed.

## Direct Maps

Entries in a direct map are associated directly with **autofs** mount points. Each **key** is the full path name of an **autofs** mount point. The direct map, as a whole, is not associated with any single directory.

## Special Maps

There are two special maps available: **-hosts** and **null**. The **-hosts** map is used with the **/net** directory and assumes that the map key is the host name of an NFS server. The **automountd** daemon dynamically constructs a map entry from the server's list of exported file systems. For example, a reference to **/net/sales/usr** would initiate an automatic mount of all exported file systems from **sales** that are mountable by the client. References to a directory under **/net/sales** would refer to the corresponding directory relative to the **sales** root.

The **-null** map, when indicated on the command line, cancels a previous map for the directory indicated. This is used in the **/etc/auto_master** map for canceling entries that would otherwise be inherited from the **+auto_master** include entry. The **-null** entries must be inserted before the included map entry.

## Executable Maps

**Note:** Local maps that have the executable bit set in their file permissions are executed by the automounter and provided with a key to be looked up as an argument.

An executable map returns the content of an automounter map entry into standard output (stdout) or returns no output if the entry cannot be determined.

## Configuration and the Master (auto_master or auto.master) Map

When initiated without arguments, **automount** consults the master map for a list of **autofs** mount points and their maps. It mounts any **autofs** mounts that are not already mounted and unmounts **autofs** mounts that have been removed from the master map or direct map. The **automount** command may be initiated to establish mounts by specifying the map information on the command line. This information can be lost if the **automount** command is invoked a second time. For the sake of administration and debugging, the **automount** command writes a reference map file that reflects the map information specified on the command line. The file is generated on *each* invocation of the command and is called **/etc/autofs_cmdline**.

**Note:** The master map can be called **auto_master** or **auto.master**. If **auto_master** isn't found, NIS looks for **auto.master**.

## Included Maps

The contents of another map can be included within another map by entering:

```
+mapname
```

The *mapname* can be a file name or the name of an NIS map. Or, *mapname* can be one of the special maps described in the following section. If the key being searched for is not located in an included map, the search continues with the next entry.

## Managing NIS Automount Maps

The following procedure addresses the most common management tasks for NIS automount maps.

### Prerequisites

NIS must be configured on your network. See "NIS Installation and Configuration" on page 7 for further information.

### Procedure

1. Edit the **/etc/auto.master** file. The >**automount** daemon, by default, reads the NIS **/etc/auto.master** map to find which directories to watch for mounts. The **auto.master** map has the following format:

   ```
   DirectoryPath     AutomountMapName
   ```

   The *AutomountMapName* field specifies a file containing the **automount** map for the directory specified by the *DirectoryPath* field. For example, the contents of the **/etc/auto.master** file on the NIS server might be as follows:

   ```
   /home/home     /etc/auto.home
   /usr/lpp       /etc/auto.direct
   ```

   The above **auto.master** file entries direct the **automount** daemon to use the `/etc/auto.home` **automount** map for the `/home/home` directory and the `/etc/auto.direct` **automount** map for the `/usr/lpp` directory.
2. Create the *AutomountMapName* files. The *AutomountMapName* files have the following format:

   ```
   Subdirectory   MountOptions   ServerName:ServerDirectory
   ```

   The `Subdirectory` field specifies a subdirectory of the *DirectoryPath* field directory of the **auto.master** file. For example, the contents of the `/etc/auto.home` file on the NIS client might be as follows:

   ```
   john           -rw,hard,intr    host1:/home/john
   bill           -rw,hard,intr    host3:/home/bill
   sally          -rw,hard,intr    host5:/home/sally
   fred           -rw,hard,intr    host9:/home/fred
   jane           -rw,hard,intr    host1:/home/jane
   ```

   The contents of the `/etc/auto.direct` file on the NIS client might be as follows:

   ```
   X11            -ro,hard,intr    lppserver:/usr/lpp/X11
   bsmEn_US       -ro,hard,intr    lppserver:/usr/lpp/bsmEn_US
   ```

```
gnuemacs        -ro,hard,intr    lppserver:/usr/lpp/gnuemacs
info            -ro,hard,intr    lppserver:/usr/lpp/info
```

3. Update the **/var/yp/Makefile** file, as follows:

   a. Add **auto.master** to the `all:` listing.

   b. Add an entry for `$(DIR)/auto.master:` at the appropriate point in the file.

   c. Add the following stanza to the **Makefile** file:

   ```
   auto.master.time: $(DIR)/auto.master
           -@if [ -f $(DIR)/auto.master ] ; then \
                   $(MAKEDBM) $(DIR)/auto.master $(YPDBDIR)/$(DOM)/auto.master; \
                   touch auto.master.time ; \
                   echo "updated auto.master" ; \
                   if [ ! $(NOPUSH) ] ; then \
                            $(YPPUSH) auto.master ; \
                            echo "pushed auto.master" ; \
                   else \
                            : ; \
                   fi \
           else \
                   echo "couldn't find $(DIR)/auto.master" ; \
           fi
   ```

   d. Add an entry for `auto.master: auto.master.time` at the appropriate point in the **Makefile** file.

   In general, the same format that is used for the **netmasks** entry in the **Makefile** file can be used for the **auto.master** entry.

4. Build the **auto.master** map with the following command:

   ```
   make auto.master
   ```

   If errors are generated, check for improper configuration of NIS, errors in the **Makefile** file, or errors in the syntax of the **/etc/auto.master** file.

5. Start the **automount** daemon with the following command:

   ```
   /usr/sbin/automount
   ```

   This starts the **automount** daemon, which reads the **auto.master** NIS map.

   In the preceding examples, when these procedures are completed, a user on the client can issue the `cd /home/home/bill` command and have the `/home/bill` directory mounted from the `host3` system onto the `/home/home/bill` directory.

## Maintaining All of the automount Maps with NIS

In the first example, the `/etc/auto.home` and `/etc/auto.direct` were local files on the client that contained all of the **automount** map needed. The contents of the **automount** maps can also be maintained by NIS. The files would still exist on the client, but the contents would be different. For example, the `/etc/auto.home` file would contain the following:

```
+auto.home
```

And the **/etc/auto.direct** file would contain the following:

```
+auto.direct
```

This directs the **automount** daemon to consult the NIS maps `auto.home` and `auto.direct` when it reads the local files. The NIS server would contain two new NIS maps. The maps would be `auto.home` and `auto.direct`. They would be added to the **/var/yp/Makefile** in the same way that the **auto.master** NIS map was added. This makes them available for use by the NIS clients running the **automount** daemon.

This facility can also be used to define local portions of the **automount** maps and then refer to the NIS maps for the rest of the **automount** map. For example, the `/etc/auto.home` file could contain the following:

```
sandy         -rw,hard,intr            host10:/home/sandy
james         -rw,hard,intr            host2:/home/james
bill          -rw,hard,intr            host20:/home/bill
+auto.home
```

This **automount** map has three local entries and also contains the NIS map `auto.home`. This way, local definitions can be maintained while taking advantage of the NIS map for the `/home/home` directory. The entry `bill` in the local map would appear in the `auto.home` NIS map. The local map entry overrides the NIS map entry.

## NIS Reference

View the commands and daemons for Network Information Service (NIS).

### NIS daemons

| Daemon | Description |
|---|---|
| **keyserv** | Stores public and private keys. |
| **ypbind** | Enables client processes to bind, or connect, to NIS. |
| **yppasswdd** | Receives and executes requests from the **yppasswd** command. |
| **ypserv** | Looks up information in local NIS maps. |
| **ypupdated** | Updates information in NIS maps. |

### NIS Commands

| Command | Description |
|---|---|
| **chkey** | Changes the user's encryption key. |
| **chmaster** | Re-executes the **ypinit** daemon and restarts the NIS daemons. |
| **chslave** | Retrieves maps from a master server and restarts the **ypserv** daemon. |
| **chypdom** | Changes the current domain name of the system. |
| **domainname** | Displays or sets the name of the current domain. |
| **keyenvoy** | Provides an intermediary between user processes and the key server. |
| **keylogin** | Decrypts and stores the user's secret key. |
| **lsmaster** | Displays the characteristics of the configuration of an NIS master server. |
| **lsnfsexp** | Displays the characteristics of directories that are exported with NFS. |
| **lsnfsmnt** | Displays the characteristics of mounted NFS file systems. |
| **makedbm** | Makes an NIS map. |
| **mkclient** | Starts the **ypbind** daemons and uncomments the appropriate entries in the **/etc/rc.nfs** file. |
| **mkkeyserv** | Starts the **keyserv** daemon and uncomments the appropriate entries in the **/etc/rc.nfs** file. |
| **mkmaster** | Starts the NIS daemons on the master server. |
| **mkslave** | Retrieves maps from the NIS master server and starts the **ypserv** daemon. |
| **newkey** | Creates a new key in the **publickey** file. |

| Command | Description |
|---|---|
| **revnetgroup** | Reverses the listing of users and hosts in network group files in NIS maps. |
| **rmkeyserv** | Stops the **keyserv** daemon and comments the entry for the **keyserv** daemon in the **/etc/rc.nfs** file. |
| **rmyp** | Removes the configuration for NIS. |
| **ypcat** | Prints out the NIS map specified by the *MapName* parameter. |
| **ypinit** | Sets up NIS maps on an NIS server. |
| **ypmatch** | Displays the value of a given key within an NIS map. |
| **yppasswd** | Changes your network password in NIS. |
| **yppoll** | Displays the order number (ID number) of the NIS map currently in use on the server. |
| **yppush** | Prompts the NIS slave servers to copy updated NIS maps. |
| **ypset** | Directs a client machine to a specific server. |
| **ypwhich** | Identifies either the NIS server or the server that is the master for a given map. |
| **ypxfr** | Transfers an NIS map from an NIS server to a local host. |

## Troubleshooting NIS

Learn how to fix problems you might encounter while administering NIS. Problems are grouped according to type. For each problem there is a list of common symptoms, a description of the problem, and one or more suggested solutions.

## Identifying NIS Client Problems

**Note:** When attempting to solve one map problem, keep in mind that the same problem may be affecting other maps as well. See "Files where NIS Appends Map Information" on page 16 for a more detailed explanation.

### Using rsh

When a machine has two interfaces and they both are given the same name, **gethostbyname** lookups for **rsh** command will fail if NIS is being used because NIS does not return both addresses, but only the first one found. This is an implementation limitation imposed by the New Database Manager (NDBM) and performance considerations. The error message is:

```
0826-825: there is a host address that does not match
```

### When Commands Hang

The most common problem occurring at an NIS client node is for a command to hang. A command can appear to hang, even though the system seems to be operating correctly. In such a case, a message similar to the following can be generated at the console:

```
NIS: server not responding for domain domainname. Still trying
```

This error message indicates that the **ypbind** daemon on the local machine is unable to communicate with the **ypserv** daemon in the given domain because systems that run the **ypserv** daemon have failed. It may also occur if the network or the NIS server machine is overloaded to the extant that the **ypserv** daemon cannot return a response to your **ypbind** daemon within the time-out period.

Under these circumstances, all the other NIS clients on your network show the same or similar problems. The condition is usually temporary. The messages are cleared when the NIS server machine reboots and the **ypserv** daemon restarts, or else when the load on the NIS server and the network decreases.

If the **ypbind** daemon is communicating with the **ypserv** daemon and the NIS server is not overloaded, one of the following problems may exist:

- The *domainname* on the NIS client machine is not set or is set incorrectly. Clients must use a domain name that the NIS servers recognize. The domain name is case-sensitive and initially set with lowercase letters. If this is case, set the domain correctly.
- Your local network may not have an NIS server machine. One way for the NIS client to bind to a server is to broadcast on the net for servers to bind to. The disadvantage to this is that it only works on the subnet, and it can result in a storm traffic when a server stops and the clients attempt to rebind. Another method is to use **ypset** command to specify a NIS server to bind to. This has the advantage of allowing a client to bind to a server on a remote network, but it must be repeated after each reboot. The preferred method is to create the **/var/yp/binding/<*domain_name*>/ypservers** file. This file contains a list of server IP addresses to attempt to bind to, one server IP address per line. The client will attempt to bind to one of the specified servers before attempting to locate one using a broadcast. If the currently bound server that is down is also listed in the file, by default the client will attempt to contact it. However, if the `YPBIND_SKIP` environment variable is set to `1` (usually set in the `/etc/environment` file) before the `ypbind` daemon is started, the server that is currently down will not be contacted again.
- The NIS server may not be up and running. Check other machines on your local network. If several clients have problems simultaneously, the server may be the cause.

  On a client system that is working normally, run the **ypwhich** command. If the **ypwhich** command never returns an answer, stop the command. Then type the following at the NIS server machine:

  ```
  ps -ef | grep yp
  ```

  Look for the **ypserv** and **ypbind** processes. If the server's **ypbind** daemon is not running, start the daemon, using the instructions in "Starting and Stopping NIS Daemons" on page 14.

  If a **ypserv** process is running, run the **ypwhich** command on the NIS server machine. If this command returns no answer, stop and restart the **ypserv** daemon by following the instructions in .

## When NIS Service Is Unavailable

When other machines on the network appear to have no problems, but NIS service becomes unavailable on your system, a variety of symptoms can occur:

- Some commands may operate correctly while others terminate and produce an error message about the unavailability of NIS.
- Some commands run slowly in a backup-strategy mode particular to the program involved.
- Some commands or daemons crash with error messages or no message at all.

  For example, messages such as the following might be generated:

  ```
  ypcat myfile
  ypcat: can't bind to NIS server for domain <wigwam>
  Reason: can't communicate with ypbind.
  ```

  OR

  ```
  /usr/etc/yp/yppoll myfile
  RPC: timed out
  ```

When symptoms like these occur, do the following:

1. Run the **ls -l** command in a directory containing files owned by many users, including users not in the local machine's **/etc/passwd** file.
2. In the listing, determine whether file owners who are not in the local machine's **/etc/passwd** file are shown as numbers rather than names. If so, NIS is not working, usually because the **ypbind** daemon is not running.
3. Use the **ps -ef** command and look for the **ypbind** daemon in the list of processes. If it is not there, start it by following the instructions in "Starting and Stopping NIS Daemons" on page 14.

By default, the NIS client will wait indefinitely for the NIS server. While waiting, logging-in to the client system is not possible. It is possible, however, to limit the length of this wait. If the **YPBIND_MAXWAIT** environment variable is set before the **ypbind** daemon is started, this value (in seconds) will limit the amount of time the NIS client will wait for the NIS server (the **YPBIND_MAXWAIT** environment variable is usually set in the **/etc/environment** file). If this limit is exceeded, the client behaves as if NIS were unavailable and continues using local files. This will allow local logins, such as root.

## When the ypbind Daemon Becomes Inoperable

If the **ypbind** daemon repeatedly crashes immediately after it is started, look for a problem in some other part of the system.

- Check for the presence of the **portmap** daemon by typing:

  ```
  ps -ef | grep portmap
  ```

  If the daemon is not running, reboot the system.
- If the **portmap** daemon is running but does not operate reliably, check the network software.

  Try to communicate with the **portmap** daemon on your machine from a different machine that is operating normally. From such a machine, type:

  ```
  rpcinfo -p client
  ```

  where `client` is the host name of the machine.
- If the **portmap** daemon is up and running correctly, the output appears in a format similar to the following:

  ```
  program   vers proto  port
  100007     2    tcp   1024  ypbind
  100007     2    udp   1028  ypbind
  100007     1    tcp   1024  ypbind
  100007     1    udp   1028  ypbind
  100021     1    tcp   1026  nlockmgr
  100024     1    udp   1052  status
  100020     1    udp   1058  llockmgr
  100020     1    tcp   1028  llockmgr
  100021     2    tcp   1029  nlockmgr
  100012     1    udp   1083  sprayd
  100011     1    udp   1085  rquotad
  100005     1    udp   1087  mountd
  100008     1    udp   1089  walld
  100002     1    udp   1091  rusersd
  100002     2    udp   1091  rusersd
  100001     1    udp   1094  rstatd
  100001     2    udp   1094  rstatd
  100001     3    udp   1094  rstatd
  ```

  – If the daemons are not listed, the **ypbind** daemon is unable to register its services. Reboot the machine.

  – If the daemons are listed, but they change each time you try to restart the **ypbind** daemon, reboot the system (even though the **portmap** daemon is up).

## When the ypwhich Command Is Inconsistent

When you use the **ypwhich** command several times at the same client node, the response varies because the status of the NIS server changes. The status changes are normal.

The binding of NIS client to NIS server changes over time on a busy network, when the NIS servers are busy. Whenever possible, the system stabilizes so that all clients get acceptable response time from the NIS servers. The source of an NIS service is not important, because an NIS server machine often gets its own NIS services from another NIS server on the network.

## Identifying NIS Server Problems

NIS server problems can most commonly occur at the following times:

### When Different Versions of an NIS Map Exist

Because NIS works by propagating maps among servers, you can sometimes find different versions of a map at the network servers. This is normal only as a temporary situation. Normal update is prevented when an NIS server or a router between NIS servers is down during a map transfer attempt. When all the NIS servers and all the routers between them are up and running, the **ypxfr** command should run successfully. If a particular worker server has problems updating a map, use the following procedure to detect and solve the problem:

1. Log in to the problem server and run the **ypxfr** command interactively. If this command fails, use the information in the error message to fix the problem.
2. If the **ypxfr** command succeeds, but you still suspect a problem, create a log file to enable logging of messages by typing the following:

   ```
   cd /var/yp
   touch ypxfr.log
   ```

   This saves all output from the **ypxfr** command to the **ypxfr.log** file. The output looks much like what the **ypxfr** command creates when it is run interactively, but each line in the log file is time stamped. The time stamp tells when the **ypxfr** command began its work. It is normal to see unusual orderings in the time stamps. If copies of the **ypxfr** command ran simultaneously but their work took differing amounts of time, the summary status line may be written to the log files in an order that differs from the order in which they were invoked.
3. Examine the log for any pattern of intermittent failure. After you fix the problem, turn off logging by removing the log file; otherwise, it continues to grow without limit.
4. If you are still experiencing problems, inspect the system **/etc/crontab** entries in the log, and the **ypxfr** shell scripts it invokes.
5. Make sure that the NIS worker server is in the **ypservers** map. If not, the **yppush** command cannot notify the worker server when a new copy of a map exists.

### When the ypserv Daemon Becomes Inoperable

When the **ypserv** process repeatedly crashes immediately after it is started, the debugging process is similar to that described for **ypbind** crashes. First, check for the **portmap** daemon:

```
ps -ef | grep portmap
```

If you do not find the **portmap** daemon, reboot the server. If there is a **portmap** daemon, type:

```
rpcinfo -p hostname
```

where `hostname` is the host name of the NIS server.

On your particular machine, the port numbers will be different. The four entries that represent the **ypserv** daemon are:

```
100004     2   udp   1027  ypserv
100004     2   tcp   1024  ypserv
100004     1   udp   1027  ypserv
100004     1   tcp   1024  ypserv
```

If these entries do not exist, the **ypserv** daemon is unable to register its services. Reboot the machine. If the **ypserv** entries exist, but they change each time you try to restart the **ypserv** daemon, reboot the machine again.

## Migrating from NIS to LDAP services

This topic describes the process of migrating from NIS to LDAP services (NIS_LDAP), which uses the schema defined by RFC 2307.

### Considerations

- The RFC 2307 schema uses case-insensitive strings to hold data. This means that aliases that differ only in case may be lost in the transition. For example, in the default protocols file, all aliases will be lost. However, both TCP and tcp will match the entry in the LDAP directory.
- UIDs and GIDs greater than 2^31-1 will be converted to their negative equivalents in twos complement arithmetic.
- The RFC 2307 schema only covers the following files and their equivalent maps:
  - passwd
  - group
  - networks
  - netgroups
  - rpc
  - hosts
  - services
  - protocols

  Other data will not be migrated

### Server Setup

To prepare the server, you will need to do the following:

1. Install the **ldap.server** and **ldap.client** packages.
2. Use the **mksecldap** command to configure the server. An example follows:

   ```
   mksecldap -s -a cn=admin -p adminpwd -S rfc2307 -u NONE
   ```

   The `-u NONE` option prevents the **mksecldap** command from migrating users and groups. If users and groups are to be migrated from NIS, this is necessary. See the **mksecldap** command description in *Commands Reference* for more details.

### Migrating data from NIS to LDAP

Data is migrated to the LDAP directory using the **nistoldif** command. The **nistoldif** command can operate in two modes: it can output LDIF data, or it can write the data directly to the server. The **nistoldif** command will not add a user or a group with a UID or GID that conflicts with one already on the server.

**Note:** You may have to increase the size of the partition containing the database that LDAP is using. By default, this will be the **/home** directory. If not enough space is allocated, and you are migrating data to the server, the **nistoldif** command will fail. In this case, increase the size of the partition and rerun the **nistoldif** command.

If you are migrating data from the default NIS domain, the **nistoldif** command will use this data by default. If you wish to use a NIS domain other than the default, you should use the `-y` flag to specify a domain. Following is an example:

```
nistoldif -h server1.ibm.com -a cn=admin -p adminpwd -d cn=aixdata
```

This migrates NIS maps from the default domain to the LDAP server `server1.ibm.com` under the `cn=aixdata` DN. If no NIS maps are present, it will fall back to the data in the **/etc** directory. The `-f` flag changes the fallback directory.

For more information, see the **nistoldif** command description in *Commands Reference*.

## Client Setup

The server must be set up before the client. Client setup depends on the migrated data being on the server.

Once the data has been migrated to the server, each client must be set up using the **mksecldap** command.

```
mksecldap -c -a cn=admin -p adminpwd -h server1.ibm.com
```

This sets up the local system to use the LDAP server on `server1.ibm.com`.

See the **mksecldap** command description in *Commands Reference* for more details.

## Netgroup Setup

Netgroup support in **nis_ldap** involves additional configuration. To enable netgroup support, the module definition for LDAP in the **/usr/lib/security/methods.cfg** file will need to include an **options** attribute with a `netgroup` value. For example, the following configuration will enable netgroup support for LDAP:

```
LDAP:
     program = /usr/lib/security/LDAP
     program_64 =/usr/lib/security/LDAP64
     options = netgroup
```

Enabling netgroup support will also activate the following behaviors:

- Users defined in the **/etc/security/user** file as members of the LDAP registry (in other words, having `registry=LDAP` and `SYSTEM="LDAP"`) will not be able to authenticate as LDAP users. These users will now become **nis_ldap** users and will require native NIS netgroup membership. To fully enable **nis_ldap** netgroup users, corresponding entries in the **/etc/security/user** file must have the `registry` and `SYSTEM` values removed or set to `compat`.
- Only **nis_ldap** users will show `compat` as their registry. Other users will show their absolute registry value.
- The meaning of registry `compat` will be expanded to include modules supporting netgroup. For example, if LDAP module is netgroup enabled, `compat` will include the following registries: files, NIS, and LDAP.

# Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing*
*Legal and Intellectual Property Law*
*IBM Japan Ltd.*
*19-21, Nihonbashi-Hakozakicho, Chuo-ku*
*Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

## Privacy policy considerations

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

UNIX is a registered trademark of The Open Group in the United States and other countries.

# Index

## Special Characters

-hosts map 27
* character, NIS automount 26
/etc/passwd file 32
/var/yp/securenets file 19
& character, NIS automount 26

## A

auto_master map file 25
autofs and master map 27
automount command 25
automounting in NIS
    direct maps 27
    executable maps 27
    file system types 27
    included maps 28
    key substitution 26
    master map 27
    multiple mounts 26
    special maps 27
    weighting factors 26
    wildcard 26

## B

bootparams map 6

## C

clients
    configuring files for 15
    configuring in NIS 11
    NIS 4
commands
    hanging 31

## D

daemons
    also see name of daemon 31
    portmap 33
    ypbind 31
    ypserv 31
DBM
    NIS maps 5
DNS
    overview 1
domain
    NIS 5
Domain Name System
    see DNS 1

## E

ethers.byaddr map 6
ethers.byname map 6
expansion key, NIS automount 26

## F

file system types, NIS 27
formatting 25

## G

group.bygid map 5, 18
group.byname map 5, 18

## H

hosts.byaddr map 5
hosts.byname map 5

## I

installing
    NIS 7

## K

key
    NIS maps 5
key substitution, NIS automount 26

## M

mail.aliases map 6, 18
mail.byaddr map 6, 18
map entries, automount 25
map entry format 25
maps, NIS
    direct 27
    executable 27
    included 28
    master 27
    special 27
master server
    configuring in NIS 8
multiple mounts, NIS 26

## N

named daemon 1
netgroup map 6
netgroup.byhost map 6
netgroup.byuser map 6
netgroups 6

netgroups.byhost map 18
netgroups.byuser map 18
netid.byname map 6
netmasks.byaddr map 6
network information services
  see DNS, NIS 1
networks.byaddr map 6
networks.byname map 6
nicknames
  NIS maps 5
NIS
  automount
    autofs 27
    direct maps 27
    executable maps 27
    file system types 27
    included maps 28
    key substitution 26
    map entry format 25
    master map 27
    multiple mounts 26
    special maps 27
    weighting 26
    wildcard 26
  chapter 3
  clients
    configuring 11
    configuring files for 15
  commands
    list of 30
  components of 3
  daemons
    list of 30
    starting and stopping 14
  domain 5
  installing 7
  maintaining 19
  maps
    changing 19
    creating nonstandard 21
    customizing input 13
    DBM 5
    key 5
    makefile 5
    nicknames 5
    propagating 22
    record 5
  maps, NIS
    how to manage automount 28
  master server
    configuring 8
  netgroups 6
  overview 1
  passwords
    changing 20
  servers 4
  setting domain name 8
  slave server
    adding 20
    configuring 10
  troubleshooting 31
  users
    adding 21
NIS (Network Information Service)
NIS (Network Information Service) *(continued)*
  servers
    master 24
  troubleshooting
    different versions of maps 34
NIS automount 26
NIS Maintenance 19
NIS Reference 30
null map 27

## P

passwd.byname map 5, 17
passwd.byuid map 5, 17
pbind daemon
  inoperable 33
portmap daemon 33, 34
protocols.byname map 6
protocols.bynumber map 6
publickey.byname map 6

## R

record
  NIS maps 5
replicated file systems 25
replicating file systems with NIS automount 25
rpc.bynumber map 6
rsh command
  troubleshooting 31

## S

servers
  NIS
    problems 34
services.byname map 6
setting
  NIS domain name 8
slave server
  configuring in NIS 10

## T

troubleshooting
  NIS
    client problems 31
    hung commands 31
    identifying server problems 34
    unavailable service 32
    ypbind inoperable 33
    ypserv inoperable 34
    ypwhich inconsistent 33

## W

weighting factors, NIS automount 26
wildcards 26

## Y

ypbind daemon 4, 31

YPBIND_MAXWAIT environment variable 32
YPBIND_SKIP 32
ypserv daemon 19, 31, 34
ypservers file 31
ypservers map 6, 34
ypset command 32
ypwhich command
    inconsistent 33
ypxfr command 34
ypxfrd daemon 19