Network Information Service (NIS) [4]

Warning: The network information service (NIS) feature is not part of the Cray ML-Safe configuration of UNICOS. This chapter does not contain any further warnings or information pertaining to the use of the Cray ML-Safe configuration of UNICOS.

4.1 About NIS

The UNICOS network information service (NIS) facility (formerly known as yellow pages) is a network service that allows information such as passwords and group IDs for an entire network to be held in one database. Implemented along with the Remote Procedure Call (RPC) and eXternal Data Representation (XDR) library routines, UNICOS NIS has the following features:

- Look-up service. UNICOS NIS maintains a set of databases that can be queried through the use of pointers, or *keys*. Programs can request the value associated with a particular key, or all of the keys, in a database.
- Network service. Programs do not need to know the location of data or how it is stored. Instead, they use a network protocol to communicate with a database server that contains the information.
- Distributed service. Databases are fully replicated on several machines, known as *NIS servers*. The servers propagate updated databases among themselves, ensuring consistency.

NIS+ was designed to accommodate large and complex networks by allowing administrators to organize users into a hierarchical structure that reflects their interactions and not physical network setups. This structure, which is a collection of network information about users, machines, and privileges, is called the NIS+ *namespace*. You define the namespace with this information in a series of tables to arrange how your organization is best structured to fulfill and share computing needs.

The namespace is organized by domains. Each domain has a principal or *master* server and at least one backup or *replica* server. Information about each user, workstation, and domain in the namespace is organized in NIS+ tables. Twelve different NIS+ tables are referenced in an NIS+ database on a UNICOS system. NIS+ tables are maintained on both master server and replica servers, and

changes to network information are easily made from any server and quickly propagated from the master server to other replicas.

The structure of the NIS+ namespace and information about its machine and human members is protected by a security system that determines the privileges of any user making requests. User access requests to network information tables can be finely controlled by NIS+ security.

The UNICOS NIS environment includes at least one Cray Research computer system and one or more front-end machines that are used to access the Cray Research system. Because UNICOS NIS differs from the NIS facility used on other systems based on the UNIX system, administration of an NIS domain that includes a Cray Research system must be different from administration of an NIS domain that does not.

If you are unfamiliar with NIS, it is recommended that you begin by reading the NIS documentation for the front-end system. When you have familiarized yourself with the general NIS mechanism, read the following sections to familiarize yourself with UNICOS NIS:

- NIS databases
- NIS maps
- NIS domains
- Servers and clients
- Masters and slaves
- Naming
- Data storage
- Supported databases
- Using NIS

  Note: Information on network information service plus (NIS+) is presented at the end of this chapter. NIS+ software is part of SunSoft's Open Network Computing plus (ONC+) product now shipped with the UNICOS operating system. UNICOS customers must purchase a separate license in order to use NIS+. See your license file, `/etc/craylm/license.dat`, to determine if the required license is installed.

For more information about NIS, see the Sun manual *YP Protocol Specifications*.

4.2 NIS Databases

UNICOS NIS contains network-wide databases. Usually, these databases contain files, such as `/etc/passwd` and `/etc/group`, that previously resided in directory `/etc`. However, users can add their own databases. Without the NIS service, each machine on the network would have its own identical copy of certain administrative files (for example, the `/etc/group` file on each machine would have to be updated with the same entry each time you added a user to the network).

UNICOS NIS can serve many databases. Servers containing copies of the databases are spread throughout the network. For example, when a machine in the network must look up something in `/etc/passwd`, it makes an RPC call to one of the servers to get the information. One server is the *master*; the other servers are the *slaves*. Only the database on the master server can be modified. The slaves can be periodically updated so that their information is synchronous with that of the master (see Section 4.6, page 313, for more information on this relationship).

4.3 NIS Maps

NIS databases contain *NIS maps*. Each map contains a set of keys and associated values. For example, the host map contains (as keys) all host names on a network and (as values) the corresponding Internet addresses. Each NIS map has a map name; programs use the map name to access data in the map.

A program must also know the format of the data in the map it wants to access. On Cray Research systems, the information in the NIS map is usually identical to the information in the `/etc/passwd` and `/etc/group` ASCII files. The maps are implemented in `dbm`(3C) format in the subdirectories of `/etc/yp` on NIS server machines.

4.4 NIS Domains

An *NIS domain* is a named set of NIS maps. An NIS server contains all of the NIS domain maps in a subdirectory of `/etc/yp`; this subdirectory is named after the domain. You can determine the name of your NIS domain by executing the `domainname`(1) command.

You must use a domain name to retrieve data from an NIS database. For example, if your NIS domain is `menagerie`, and you want to find the Internet address of host `dbserver`, you must use the following command to ask NIS

for the value associated with the `dbserver` key in the `hosts.byname` map within the NIS domain `menagerie`:

```
ypmatch -d menagerie dbserver hosts.byname
```

Maps for the `menagerie` domain would be in subdirectory `/etc/yp/menagerie`. Each machine in the network belongs to a default domain, set when `/etc/ypstart` is entered at boot time. The `domainname` command is entered and sets the domain name to the domain that is configured in `/etc/config/ypdomain.txt`. For information on changing a domain name, see Section 4.10.3, page 318.

4.5 Servers and Clients

Servers provide resources; *clients* use them. However, neither a server nor a client is necessarily restricted to one role. For example, consider the following services:

Service	Description
UNICOS NFS	If a server machine has exported a file system, UNICOS NFS allows client machines to mount the file system remotely and to access files in place. However, a server that exports file systems can also mount remote file systems exported by other machines, thus becoming a client; therefore, a specific machine can be both server and client, client only, or server only.
NIS	The NIS server is a process that runs on a machine that might not be an NFS server. A process can request information from the NIS database, making it unnecessary to have such information on all machines. All processes that use NIS services are NIS clients. Clients can be served by NIS servers on the same machine or by NIS servers that run on another machine. If a remote machine that runs an NIS server process crashes, client processes can receive NIS services

from another machine. Consequently, NIS services are usually available.

4.5.1 Servers

To become a server, a machine must have the NIS databases and run the NIS daemon `ypserv`(8). The `ypinit`(8) command automatically invokes this daemon and contains a flag that indicates whether you are creating a master or a slave server. When you update the master copy of a database, you can use the `yppush`(8) command to send the changes to the slave server. Conversely, you can use the `ypxfr`(8) command from the slave server to receive any changes from the master.

The makefile in `/etc/yp` first uses the `makedbm`(8) command to create a database, and then calls the `yppush`(8) command to send the change throughout the network.

4.5.2 Clients

A client machine that is not a server does not access local copies of `/etc` files (except for the `/etc/passwd` and `/etc/group` files); instead, it makes an RPC call to an NIS server each time it needs information from an NIS database. The `ypbind` (see `ypserv`(8)) daemon retains the name of a server. When a client boots, `ypbind` broadcasts, requesting the name of an NIS server. Similarly, if the old server crashes, `ypbind` broadcasts, requesting the name of a new NIS server. The `ypwhich`(1) command gives the name of the server to which `ypbind` currently points.

You can use the `ypcat`(1) and `ypmatch`(1) commands to read and search files because client machines do not contain entire copies of files in the NIS database. For example, to search for a user's password entry, enter one of the following commands:

`ypcat passwd | grep` *username*
`ypmatch` *username* `passwd`

4.6 Masters and Slaves

An NIS server is either a master or a slave. For any map, one NIS server is designated the master; all changes to the map should be made on that machine. The changes are then propagated from NIS master to NIS slaves. When a map is built by the `makedbm`(8) command, it is internally time-stamped. If you build

an NIS map on a slave server, you temporarily break the NIS update algorithm and must then synchronize all versions manually.

Different maps can have different servers as the master. A given server can also be master with regard to one map, and slave with regard to another. To avoid confusion, however, it is recommended that one server be designated as master for all maps created by the `ypinit`(8) command in one domain. The examples in this chapter assume that one server is the master for all maps in a given domain.

4.7 Naming

You can use the `domainname`(1) command and the `getdomainname` library routine (see `getdomain`(3C)) to give two networks different domain names. The NIS databases for each domain name is stored in the `/etc/YP/`*domainname* directories. Thus, the `passwd.byname` map for the `menagerie` domain is stored as `/etc/yp/menagerie/passwd.byname/pag` and `/etc/yp/menagerie/passwd.byname/dir.`

For example, assume that a company has two different networks, each with its own separate list of hosts and passwords. Within each network, the user names, numerical user IDs, and host names are unique. However, some duplication occurs between the two networks. Although one domain name should be used whenever possible, in this case, the `hostname`(1) command and the `gethost`(3C) library routine would not be able to identify a host or user name uniquely. Therefore, the `domainname` command and the `getdomainname` library routine should be used to identify the domain in which a particular host or user name can be found.

4.8 Data Storage

The NIS data is stored in `dbm`(3C) format. For example, the NIS map `passwd.byname` for the `menagerie` domain is stored on an NIS server as `/etc/yp/menagerie/passwd.byname.pag` for the file that contains only data, and as `/etc/yp/menagerie/passwd.byname.dir.` for the directory that contains a bit map. The `makedbm`(8) command takes an ASCII file such as `/etc/passwd` and converts it into a `dbm` file suitable for NIS to use. However, you usually use the makefile in `/etc/yp` to create new NIS `dbm` files. This makefile, in turn, calls `makedbm`.

4.9 Supported Databases

Cray Research supports the `publickey`, `passwd`, `group`, and `netgroup` NIS files. The `publickey` and `netgroup` files function the same on Cray Research systems as on Sun systems. The `passwd` and `group` files function differently; they are described in the following sections.

4.9.1 the `/etc/passwd` File

The `/etc/passwd` file is stored as two separate maps in the NIS database. The first map, `passwd.byname`, is indexed by login name. The second map, `passwd.byuid`, is indexed by user ID.

You can also reference the maps by nicknames. For example, when you use the `ypcat`(1) command to display or print information about a map, and you use the map's nickname, the routine translates that nickname into the actual name of the map. Therefore, `ypcat passwd` is translated into `ypcat passwd.byname`, because no file is named `passwd` in the NIS database. The `ypcat -x` command furnishes a list of map names and nicknames.

The user database (UDB) generates the `/etc/passwd` file. The UNICOS system requires the use of the UDB for login control, accounting, and user limits. Therefore, a user must have a UDB entry in order to log in to a Cray Research system that runs the UNICOS system. The UDB file is searched first when a user calls the `getpwent`(3) library routine.

If password control is to be administered from the NIS database rather than from the UDB, all users should be set up in the UDB with an empty password field, and the UDB flag `permbits:YP:` must be set in the UDB entry for each user whose password is to be maintained by NIS. This method is analogous to the method used on other systems based on the UNIX system, in which each user listed in the NIS password map has the following entry in the `/etc/passwd` file:

+*user*::*uid*:*gid*:::

These entries direct NIS to fill in the missing fields from the NIS password database. (The colons are delimiters for the login name, password, uid, gid, comment, home directory, and shell fields, respectively.)

For more information on the UDB, see General *General UNICOS System Administration*, publication SG–2301.

The generation of the `/etc/passwd` file is done automatically by certain UDB commands. The password file generated by the UDB is merged with the master

copy of the password file that resides on the master server. The login name, user ID, and default group ID fields in the UDB-generated password file must be identical to those in the master copy when the files are merged. Because these fields are required to be present in the UDB, they will never be empty in the UDB-generated password file. However, if any of the password fields in the UDB-generated password file are empty, you must enter the associated passwords into the master copy when merging the two files.

4.9.2 The /etc/group File

On Cray Research systems, the `/etc/group` file is read and sometimes modified by the UDB. However, the group file does not depend on the UDB as much as the password file does. The UDB can modify the group file only when a numerical group ID is assigned to a UDB entry and that numerical ID does not appear in the group file. In this case, the UDB generates a new group name for that numerical ID and adds the numerical ID and the new name to the end of the group file.

The group file can be administered as on any UNIX system, even though it can be modified by the UDB commands. However, because the UDB modifies the `/etc/group` file directly, the `/etc/group` file on the Cray Research system should be a complete file (a file without a `+` entry).

4.9.3 Changing NIS Data

Each time the `/etc/passwd` file or `/etc/group` file is modified on the master NIS server, a new NIS database should be generated. You can use the `passwd`(1) command to change the password before generating the new NIS database or you can use the `yppasswd`(1) command to change the password after generating the new NIS database.

To change other data in the NIS, you must log in to the master server and edit databases there; the `ypwhich`(1) command tells you which NIS server to use.

To change their passwords in the NIS database, users of Cray Research systems should use the `yppasswd` command (this command works only if you have started the `yppasswdd`(8) daemon on the NIS master server machine).

4.10 Using NIS

The primary goal of NIS is to maintain one administrative environment for the machines in a local network. The NIS facility maintains a consistent set of login

names and IDs across multiple machines. When you configure NIS into the UNICOS operating system, the Cray Research system becomes a participating member of one administrative environment. UNICOS commands and library calls include the necessary code to support access to NIS databases. Although you can use NIS to access other information, Cray Research currently supports the distribution of only the following databases:

- `group`
- `passwd`
- `publickey`
- `netgroup`

The following sections describe the relationship of UNICOS NFS to NIS, NIS installation procedures, NIS domain configuration procedures, the procedure for adding a user to the UNICOS NIS domain, precautions concerning procedures not recommended by Cray Research, and secure RPC.

4.10.1 NIS and UNICOS NFS

The UNICOS network file system (NFS) is designed to be used within one administrative environment (such as an NIS domain). Use of NIS ensures that login names and, most importantly for NFS, numerical user IDs and group IDs are unique for all users in the administrative environment.

The need for ID mapping in UNICOS NFS is reduced when the Cray Research system is a member of an NIS domain. Although the ID mapping process incurs minimal overhead, the ID mapping tables reside in kernel memory. However, the Cray Research system is typically placed in the middle of several (perhaps many) different administrative environments. Therefore, ID mapping can still be used between the Cray Research system and any host or network that is not part of the NIS domain of which the Cray Research system is a member. In this case, the UDB is generally a superset of the NIS password and group maps. See Chapter 3, page 243, for more information on UNICOS NFS and ID mapping.

4.10.2 Configuring NIS

If you are upgrading from UNICOS 9.0 and are using the conversion utility, the NIS feature is on or off, depending on whether the feature was turned on or off in your UNICOS 9.0 configuration. Otherwise, the NIS feature is on by default.

If you are using the UNICOS Installation and Configuration Menu System (ICMS), see the `Configure System -> Major software configuration` menu for the selection that turns on the NIS feature.

If you are not using the UNICOS ICMS for your configuration, you can turn on the NIS feature by modifying the `/etc/config/config.mh` file. Change the following line

```
#define CONFIG_YP 0
```

to read as follows:

```
#define CONFIG_YP 1
```

After you make this change, follow the remaining system build procedures outlined in the *UNICOS System Configuration Using ICMS*, publication SG–2412.

4.10.3 UNICOS NIS Domain Configuration Procedure

After NIS has been installed, you are ready to configure the Cray Research system as a slave server. Making the Cray Research system a slave server instead of a master server or a client provides the most efficient performance.

It is not recommended that you configure the Cray Research system as a master server, because responding to requests from other clients that are binding to the Cray Research server uses Cray Research cycles; this is not the most efficient use of the capabilities of the Cray Research system. Also, as a master server, the Cray Research system can overload the network with NIS traffic.

It is not recommended that you configure the Cray Research system as a client because the speed of the system overloads the NIS server, and because the performance of user programs is slowed when the network is accessed for look-up functions.

Use the following steps to configure the Cray Research system as a slave server:

1. Before initializing the system for multiuser mode, set the NIS domain name to null, as follows:

   ```
   domainname ""
   ```

2. Initialize multiuser mode, or initialize and configure the network.

3. Specify the domain name on the Cray Research system to be the same as the name of the domain for which it is to be a slave server, as follows:

   `domainname` *domain*

4. Run the `ypinit`(8) script (this needs to be done only once), as follows:

   `/etc/yp/ypinit -s` *YP_master_server_hostname*

5. Reset the domain name to null, as follows:

   ```
   domainname ""
   ```

   This prevents error messages from occurring during any unintentional reference to NIS until the remaining configuration is complete.

The Cray Research system is now known to the NIS master server as a slave server; that is, the host name of the Cray Research system has been added to the NIS database `ypservers`. The Cray Research system now has copies of the NIS databases that it supports.

If you are using the startup procedures provided with the UNICOS system (that is, the `/etc/ypstart` file), and if you turned on the NIS feature during installation, you can specify the NIS domain name by placing the name in the `/etc/config/ypdomain.txt` file. One way of specifying the NIS domain name in this file is as follows:

`echo` *your_NIS_domain_name* `> /etc/config/ypdomain.txt`

The `/etc/ypstart` script accesses this file to set the NIS domain name and then starts the required NIS daemons.

If you are using the UNICOS ICMS for your configuration, use the `Configure System -> Network configuration -> NIS configuration` menu to set the contents of this file.

If you are not using the startup procedures provided with the UNICOS system (that is, if you are using a modified `/etc/netstart` file or another script of your own creation), add the following commands to your start-up file:

```
domainname your_NIS_domain_name
ypserv
ypbind -h 'hostname'
```

If you want to use secure RPC, you must start up the `keyserv` process by invoking the `/etc/keyserv` daemon in the `/etc/netstart` script.

4.10.4 Adding a User to the UNICOS NIS Domain

If the NIS domain has been configured as recommended in Section 4.10.3, page 318, you can add a user to the NIS domain, as follows:

1. Choose a user ID for the new user. Examine the UDB and the NIS database to ensure that the user ID is unique.
2. Add the user to the UDB. Ensure that the `NIS_PERMBITS` flag is set in the `PERMBITS` field of the UDB entry (this flag indicates the use of UNICOS NIS). If password control is to be administered from the NIS database rather than from the UDB, you must leave the password field empty.
3. Copy the new user's UDB-generated `/etc/passwd` file entry into the `/etc/passwd` file that resides on the master server.
4. As `root`, run `passwd`(1) on the NIS master server machine to give the new user a password, because the UDB-generated password file entry does not include one.
5. Remake the NIS database.
6. Create the new user's home directory. You must determine whether the home directory on the Cray Research system is the same as that in the rest of the NIS domain.

   Note: When a user is added to or deleted from the UNICOS NIS domain, both the UDB and the NIS master server's password database reflect the change.

4.10.5 Precautions Concerning Sets of Users

It is possible, but not recommended, to have an intersecting set of users in the NIS database and the UDB; that is, systems other than Cray Research systems might have users who are allowed to use the Cray Research system and users who are not allowed to use the Cray Research system. Consequently, some users on a given system would be listed in the UDB, and some would not. Although users listed in the NIS databases can log in anywhere else within the NIS domain, only users listed in the UDB can use the Cray Research system. Such an environment defeats the purpose of putting the Cray Research system into an NIS domain, unless this domain is one of many administrative environments that include the Cray Research system.

When intersecting sets of users are present, you must be careful when merging the UDB-generated password file with the password file that is used to generate the NIS database. Take the following precautions:

- Ensure that users who are allowed to use the Cray Research system and are members of the NIS domain have identical login names, user IDs, and group IDs in the UDB and the NIS databases.

- Ensure that users who are allowed to use the Cray Research system but are not members of the NIS domain do not have their UDB login names, user IDs, or group IDs in the NIS database.
- Ensure that users who are not allowed to use the Cray Research system but are members of the NIS domain do not have their NIS database login names, user IDs, or group IDs in the UDB.

4.10.6 Precautions Concerning the Cray Research System As a Master Server

If the Cray Research system is configured as a master server, it is recommended that no other machines bind to it.

You cannot use the `/etc/passwd` and `/etc/group` files to generate the NIS password and group maps. Copies of these files must be placed elsewhere; the information missing from the UDB must be added manually. This makes password control more difficult.

You cannot use a standard command to give a user a password before making the NIS password map. The `passwd`(1) command modifies the UDB (not the `/etc/passwd` file). The `yppasswd`(1) command modifies the NIS password map. However, you should assign the user a password before building the NIS database. Therefore, you must use the `-f` option of the `passwd`(1) command to change the password field in a `passwd`(5) format file, rather than in the UDB.

4.10.7 Precautions Concerning NIS and UNICOS Security

If UNICOS security features are enabled on your Cray Research machine, you must take special precautions to ensure that NIS will operate properly. Specifically, all local host network interfaces must be added to the network access list (NAL) in the network security file `/etc/config/spnet.conf`. For more details, see the `spnet`(8) command. As a result, NIS processes (such as `portmap`, `ypserv`, and `ypbind`) can communicate with each other through all local interfaces.

4.10.8 Secure RPC

The secure Remote Procedure Call (RPC) subsystem is the means by which the `AUTH_DES` style of RPC authentication is implemented. See the *Remote Procedure Call (RPC) Reference Manual*, publication SR–2089, for details of RPC authentication.

Each user of secure RPC must have an entry in a special NIS database of public and private keys. There must be one such entry for each host that the user accesses. Similarly, each host that supports secure RPC must have an entry for each server it accesses.

The NIS database file is called `publickey.byname` and it consists of 16 characters. This implies that, if secure RPC is to be run, this file must exist on a file system that supports names of this length.

The following sections describe methods for generating the database and developing applications.

4.10.8.1 Generating the Database

Issue the `newkey`(8) command to add entries to the `/etc/publickey` file. This command creates public key/private key pairs for users and hosts on the network. A *public key* is accessible to all users; a *private key* is encrypted according to the Data Encryption Standard (DES) with the existing password, and it is accessible only to the user or host to which it is assigned.

To add a user to the database, enter the following command:

`/etc/newkey -u` *username*

The `newkey` program prompts for the login password of the user specified by *username* and then creates a unique public key/private key pair for that user.

To add a host to the database, enter the following command:

`/etc/newkey -h` *hostname*

The `newkey` program prompts for the root password of the host specified by *hostname* and then creates a unique public key/private key pair for that host.

4.10.8.1.1 Database Format

Secure RPC authentication uses a cryptographic scheme that allows each client/server pair to obtain a unique key with which authentication data can be encrypted. The entries in the `publickey.byname` database are of the following format:

opsys.id`@`*domain publickey*`:`*privatekey*

The fields of the entry are as follows:

opsys	Operating system. This field is always `unix` in the current implementation.
id	UNICOS or UNIX identifier for the user or host. For users, it is the same as the *uid* field in the `passwd` file. For hosts, it is the full name of the host, which can be obtained by executing the `ypwhich`(1) command.
domain	Name of the NIS domain within which the database resides.
publickey	String of 48 hexadecimal digits that represent a 192-bit public key for this ID. This public key is stored in a nonencrypted form and can be read by any user on the system.
privatekey	String of 48 hexadecimal digits that represent a 192-bit private key, corresponding to the previous public key field. Unlike the public key field, the private key field is stored in a DES-encrypted format. Only the user or host specified by *id* can obtain access to the private key in its nonencrypted format.

After the new users and hosts are added to the `/etc/publickey` file, you must rebuild the `publickey.byname` database. If the master server is a Cray Research machine (which Cray Research does not recommend), enter the following command in the `/etc/yp` directory:

```
make -f yp.mk publickey
```

Note: The secure RPC subsystem depends on the existence of the NIS database file of public and private keys. This implies that NIS must be configured on the Cray Research system if it is to support secure RPC.

4.10.8.1.2 Database Access

The `keyserv`(8) program accesses the `publickey.byname` database, encrypts and decrypts private keys, and performs the relatively complex mathematics used to implement the public key system used by secure RPC. The `keyserv` process is a daemon that is usually started up from the `/etc/netstart` (or equivalent) script.

A `keyserv` process must run on both the client and the server machines. This process can be run only by `root`, and it binds to a reserved port on a user

datagram protocol (UDP) socket. `keyserv` registers itself with `portmap`(8), and it is always program number 100029.

To determine whether a `keyserv` process is active on any given host, users can enter the following command:

`rpcinfo -u` *hostname* `100029`

The `rpcinfo` command either returns the following error message:

```
rpcinfo: RPC: Program not registered
program 100029 is not available
```

or returns the following completion message:

```
program 100029 version 1 ready and waiting
```

Communication between a client or server and the `keyserv` process is accomplished through a process called `keyenvoy`(8). The `keyenvoy` process is `setuid root` and cannot be run interactively. It is created and destroyed dynamically by the RPC library routines in `libc`. The `keyenvoy` process creates a secure communications channel between a client or server process and `keyserv`. The `keyenvoy` process communicates with `keyserv` through secured local RPC channels. `keyenvoy` communicates with the client or server process through `stdin` and `stdout`. This process should be completely transparent to all users.

The `keylogin`(1) program informs `keyserv` that a user is interested in using the secure RPC subsystem. `keyserv` then caches the public key/private key entry for this user so that it does not have to look up this information at run time.

Usually, there is no reason for the user to run the `keylogin` program, because the `login`(1) program informs `keyserv` of new users. However, if for some reason the `login` program fails to communicate with `keyserv`, or if the `keyserv` program crashes and must be restarted, or if a user has a null password, the `keylogin` program can be invoked to inform `keyserv` directly of the user's interest in using secure RPC.

When `keylogin` is invoked, the user is prompted for the password. If the password is incorrect, this error is reported to the user. Unless an error message indicates otherwise, the password has been accepted by the RPC subsystem, and `keyserv` has cached the relevant key information successfully.

4.10.8.2 Developing Secure RPC Applications

To prevent false identification, some restrictions must be observed in developing secure RPC applications.

If a client wants to use a server process that is running as `root`, the client must specify the network name of the host as the first parameter of the `authdes_create()` call. This specification ensures that the server that is running on the remote host is actually a `root` process. The network name of the host can be obtained by making a call to the `host2netname` routine, as follows:

```
char    netname[MAXNETNAMELEN+1];
char    *hostname;
char    *domainname;
hostname = "mycray";
domainname = "my_yp_domain";
host2netname(netname, hostname, domainname);
```

When this call returns, the network name of the host is written in array `netname`. If the `domainname` pointer is null, the `host2netname` routine uses the default domain name.

If a client wants to use a server process that is not running as `root`, the user must know the network name of the user who is running it. This name can be obtained through a call to the `user2netname` routine, as follows:

```
char    netname[MAXNETNAMELEN+1];
int     uid;
char    *domainname;
uid = 134;      /* assume remote user has uid 134 */
domainname = "my_yp_domain";
user2netname(netname, uid, domainname);
```

When this call returns, the network name of the user is written into array `netname`. Again, if the domain name pointer is null, the `user2netname` routine uses the default domain name. This network name can then be passed as the first argument to the `authdes_create` call.

4.11 About NIS+

The following sections briefly describe the administration basics of the network information service plus (NIS+) facility. This information is aimed at system and network administrators who have a working knowledge of NIS version 2

and client-server networks. If you are unfamiliar with NIS, you should first read the previous sections in this chapter.

For an introduction to NIS+ that describes conceptual aspects and pre-setup planning strategies, see *ONC+ Technology for the UNICOS Operating System*, publication SG–2169. This guide contains a discussion of enterprise networks, NIS+ domains, NIS+ objects, and the hierarchical structure of the NIS+ namespace. The following sections represent a subset of information on the functionality of NIS+. See also Rick Ramsey's book, *All About Administering NIS+*, for more detailed information on NIS+ operations, and migration from NIS to NIS+.

The following sections briefly describe what NIS+ is and how to set it up on your UNICOS system:

- NIS+ licensing
- NIS+ overview
- Comparing NIS and NIS+
- Components of NIS+
- Planning your NIS+ namespace
- Setting up your first NIS+ domain
- How to set up a root domain
- Initializing an NIS+ client
- Setting up an NIS+ server
- How to set up a nonroot domain
- Administering your NIS+ namespace
- Migrating from NIS to NIS+

4.12 NIS+ Licensing

Network Information Service plus (NIS+), is a trademarked software product developed for the Solaris operating system by SunSoft Inc., a Sun Microsystem company. NIS+ is one of several software products that make up SunSoft's Open Network Computing plus (ONC+) product line.

Under agreement with SunSoft, Cray Research has ported NIS+ for the UNICOS operating system. Although this NIS+ software is packaged with the current release of UNICOS, it is a separately licensed product. You must purchase a Cray Research license to use NIS+ and other SunSoft ONC+ products on your UNICOS systems.

To see if you already have a Cray Research license for NIS+, enter any of the NIS+ commands, such as the following `nisls`(8) command:

```
nisls
```

If you do not have an ONC+ license, the following error message appears on your screen:

```
ONC+ license required
```

If you would like to purchase a Cray Research license to use NIS+ or other ONC+ software, contact your purchasing department or your Cray Research sales representative. NIS+ is a new network information service that was created to replace NIS.

4.13 Comparing NIS and NIS+

As computer networks become larger and more complex, administrative complexity grows as well. The model of central administration for computing networks is obsolete because keeping track of information for all the users and machines in a enterprise network is labor intensive and slow to update.

NIS+ has many advantages over NIS. Speed and security are the most prominent differences. A comparison of NIS and NIS+ is summarized in the following table.

Table 5. Comparing the features of NIS and NIS+

Feature	NIS	NIS+
Organization	Flat domains	Hierarchical structure
Data storage	Bicolumn maps	Multicolumn tables
Security	No authentication available	DES authentication

Feature	NIS	NIS+
Information	One network service	Multiple network services (NIS, NIS+, DNS, or local files)
Server updates	Delayed batch propagation	Immediate incremental updates

4.14 Components of NIS+

NIS+ consists of the following main components:

- NIS+ namespace
- Directory objects
- NIS+ domains
- NIS+ servers
- NIS+ clients
- NIS+ tables
- Name service switch

4.14.1 NIS+ Namespace

The NIS+ namespace is a collection of network information stored by NIS+. The namespace can be configured in a variety of ways to fit the needs of an organization. Although the structure of an NIS+ namespace can vary from site to site, all sites use the same structural components including directories, tables, and groups. These components are called NIS+ objects. There is always only one namespace per computing environment.

4.14.2 Directory Objects

Directory objects define the sections of the namespace. When these directory objects are arranged in a tree-like structure, they divide the namespace into separate parts. A namespace can have several levels of directories. The topmost directory is called the *root* directory. A namespace with only one directory is defined as *flat*.

With two or more directories arranged in a hierarchical organization, an NIS+ namespace looks similar to the way UNIX directories are organized in

hierarchical file systems. UNIX directories, however, are designed to hold files, and NIS+ directories are designed to hold NIS+ objects.

NIS+ objects include other directories, tables and groups containing information about the network's machines, processes and users. Any NIS+ directory that stores NIS+ groups is named `group_dir`. Any directory that stores NIS+ system tables is named `org_dir`. NIS+ system tables contain several categories of network information and they are briefly described in this section.

4.14.3 NIS+ Domains

Domains are not tangible locations or physical objects of the NIS+ namespace. They are names for sections of the namespace and therefore they reflect the organization of users and machines in a computing environment. Domains are designated to support separate portions of the namespace. An NIS+ domain consists of a directory object, its `org_dir` directory, its `groups_dir` directory, and a set of NIS+ tables. Each domain contains client machines and servers to support the users working in that section of the namespace. The servers keep track of who (clients) is on the network and what access rights they have.

4.14.4 NIS+ Servers

The servers store the domain's directories, groups, and tables. They answer requests for access from users, administrators, and applications. Each domain is supported by one set of servers, but that same set of servers can also support more than one domain. A server that supports a domain is not associated with the domain but with the domain's directory.

When the connection between the server and the directory is established, the directory stores the name and Internet Protocol (IP) address of its server. This information is used by clients to send requests for service.

Two type of servers support an NIS+ domain: *master* and *replica* servers. The master server of the root domain is called the *root master server*. A namespace has only one root master server. Both master and replica servers store NIS+ tables and answer client requests. The master server stores the master domain tables. The replicas store only duplicates. The administrator loads information into the tables in the master server, and the master server propagates the information to replica servers. Any workstation can be set up as either a master server or a replica server. Sometimes a master server is a workstation that has more disk capacity to handle the many client requests.

4.14.5 NIS+ Clients

An NIS+ client is a workstation that has been set up to receive NIS+ service. An NIS+ client can access any part of the namespace subject to security constraints. If the client has been authenticated and if it has been granted permission rights, the client can access information or objects in any domain in the namespace. When a client requests access to the namespace, it is actually requesting access to a particular domain in that namespace. Therefore, a client sends its request to the server that supports the domain it is trying to access.

4.14.6 NIS+ Tables

There are at least 11 standard NIS+ tables that contain network information about your namespace. NIS+ tables on UNICOS systems are listed in Table 6, page 330. These tables have a column-entry structure and, therefore, have several advantages over other sources of network information, such as flat bicolumn maps or local flat `/etc` files. All NIS+ tables have the same structure of rows and columns. A client with proper access rights can access the information of the whole table or by an individual key, column or entry. NIS+ software examines the privileges of a user by *authentication*, and either allows or refuses access requests to network information by a process called *authorization.*

Custom NIS+ tables can be set up by administrators. They can be searched individually by entry or colum, or several can be symbolically linked or connected by a concatenation path.

Table 6. NIS+ tables on UNICOS systems

NIS+ table	Fields	Table information for each domain
`auto_home`	Mount point Options and locations	Names of users and the locations of their home directories
`auto_master`	Mount point Map name	Mount points and names of automounter maps
`cred`	Secret key Public key	Credentials of client workstations and client users
`group`	Name Password GID Numbers	Names, passwords, IDs, and members of UNICOS groups

NIS+ table	Fields	Table information for each domain
`ethers`	Ethernet address Official host name	Names and Ethernet addresses of workstations
`netgroup`	Group name List of members	Names of network groups and lists of their members.
`networks`	Network name Network number Aliases	Internet names, and numbers of networks and their aliases
`protocols`	Protocol name Protocol number Aliases Comments	Names, numbers, and aliases of protocols used by the Internet
`rpc`	RPC program name RPC program number Aliases Comments	Names, numbers, and aliases of RPC programs
`services`	Service name Port/Protocol Aliases Comments	Names, port numbers, and aliases of Internet services
`timezone`	Time zone name Workstation/Domain name Comments	The default time zone and names of workstations or one domain name

4.14.7 Name Service Switch

NIS+ interacts in a different way with `/etc` files than NIS. NIS+ interacts with all network information services, including `/etc` files, through the *name service switch*. A configurations file is located on every NIS+ client and lists the sources for network information for that client. Sources of network information include:

- `/etc` files
- NIS bicolumn maps
- Domain name service (DNS) zone files
- NIS+ tables

NIS+ lookups follow the look up sequence defined by the configuration file on NIS+ clients. A sample configuration file is set up like the following table.

Table 7. Sample NIS+ client configuration

Type of information	Service	Look up sequence
`passwd:`	`files`	`nisplus`
`group:`	`files`	`nisplus`
`hosts:`	`nisplus`	`[NOTFOUND=return] files`
`services:`	`nisplus`	`[NOTFOUND=return] files`
`networks:`	`nisplus`	`[NOTFOUND=return] files`
`protocols:`	`nisplus`	`[NOTFOUND=return] files`
`rpc:`	`nisplus`	`[NOTFOUND=return] files`
`ethers:`	`nisplus`	`[NOTFOUND=return] files`
`publickey:`	`nisplus`	
`netgroup:`	`nisplus`	
`automount:`	`files`	`nisplus`
`aliases:`	`files`	`nisplus`

See the `nsswitch`(4) man page for more detailed information on the name service switch and corresponding data files.

4.14.8 NIS+ Commands

The following NIS+ administration commands control the setup and maintenance of the NIS+ namespace. The printed man pages for these commands are found in section 8 of the *UNICOS Administrator Commands Reference Manual*, publication SR–2022. Man pages describe in detail the options and parameters of command use.

Table 8. NIS+ administration commands

Command	Description
`nisaddcred`	Creates security credentials for NIS+ principals
`nisaddent`	Creates NIS+ tables from corresponding `/etc` files and NIS maps
`nis_cachemgr`	Maintains a cache of location information about NIS+ servers
`niscat`	Displays the contents of NIS+ tables and objects
`nischgrp`	Changes the group owner of NIS+ objects or entries
`nischmod`	Changes the access rights of NIS+ objects or entries
`nischown`	Changes owner of NIS+ object
`nischttl`	Changes the time-to-live value of an NIS+ object or entry
`nisctl`	Controls the operation of NIS+ servers; enters cache flushing, debugging and report printing
`nisd`	ONC RPC daemon that implements NIS+ service
`nisdefaults`	Displays NIS+ default values returned by NIS+ local name functions
`niserror`	Displays and translates NIS+ error number messages into text
`nisgrpadm`	Creates or destroys NIS+ groups and administers principals in those groups.
`nisinit`	Initializes a machine to be an NIS+ client and server
`nisln`	Symbolically links NIS+ objects
`nislog`	Displays the contents of NIS+ server transaction log
`nisls`	Lists the contents of an NIS+ directory
`nismatch`	Searches NIS+ tables by matching text strings
`nisgrep`	Searches NIS+ tables by text string or *keypat* expressions
`nismkdir`	Creates a nonroot NIS+ directory or adds a replica to an existing directory
`nispasswd`	Changes NIS+ password information
`nispath`	Prints out search path of a given NIS+ name
`nisping`	Sends a ping to all replica servers of an NIS+ directory
`nisrm`	Removes NIS+ objects from the namespace
`nisrmdir`	Removes existing NIS+ directories

Command	Description
`nissetup`	Initializes NIS+ domain
`nisshowcache`	Prints out the contents of the shared cache file
`nisstat`	Reports NIS+ server statistics
`nistbladm`	Creates, deletes, adds, modifies and removes NIS+ tables
`nistest`	Returns the state of the NIS+ namespace
`nisupdkeys`	Updates the public keys in an NIS+ directory object

4.14.9 NIS+ API

The NIS+ application programming interface contains 54 functions that can be called by an application to maintain and manipulate NIS+ objects. For reference, these functions can be organized into nine families. Each function name listed below is also the name of the NIS+ man page that describes its family of routines.

Table 9. NIS+ API functions

Functions	Family
`nis_subr`	Application subroutine functions
`nis_db`	Database access
`nis_error`	Error message display functions
`nis_groups`	Group manipulation functions
`nis_local_names`	Local name functions
`nis_names`	Object manipulation functions
`nis_tables`	Table access functions
`nis_admin`	Transaction log functions

Man pages describing NIS+ functions and library routines are found in an appendix in *ONC+ Technology for the UNICOS Operating System*, publication SG–2169. An API NIS+ programming example using these routines also presented in *ONC+ Technology for the UNICOS Operating System*, publication SG–2169.

4.15 Planning Your NIS+ Namespace

Before setting up NIS+ on your UNICOS system, it is recommended that you do the following tasks:

1. Diagram the structure of your organization.
2. Divide this structure into administration groups.
3. Select servers that will support each group.
4. Determine the access rights of users and groups in your organization.

   Note: If an NIS namespace is already defined at your site, you can use its flat domain structure for your new NIS+ namespace. NIS+ contains scripts that allow administrators to construct an NIS+ namespace from NIS bicolumn maps or local files.

When designing your namespace and setting up your domains, consider the following factors in your plan:

- Locations of NIS+ tables

  Consider that the lower in the domain hierarchy that you store tables, the easier it is to administrate them.

- Custom NIS+ tables

  Decide which NIS maps can be converted to NIS+ tables and whether any of your current applications depend on existing NIS maps.

- Connections between NIS+ tables

  Decide whether to connect your tables by paths or links. This decision weighs convenience and performance.

  Establishing a path from tables low in one hierarchical domain to a single table in a remote domain may save look-up time because a search examines the remote table directly, without scanning the tables above it in the home domain. Another advantage to this strategy is that administrative updates to the remote domain table are visible to users across domains. Linking NIS+ tables eliminates some performance concerns because NIS+ lookups do not search a local table in the users path first, but enter a search directly on the linked table.

  Note: Although decreasing the number of tables searched increases the performance of NIS+, look-up searches across domains makes users in one domain dependent on the network availability of another domain.

The following guidelines may help you decide how to set up, customize, locate, or link NIS+ tables.

- Every domain must have access to every standard table.
- Frequently accessed data should be located as close to users in a domain as possible.
- The lower in the hierarchy data is located, the easier it is to administer.
- Data that is frequently accessed by many domains should be located higher in the hierarchy.
- NIS clients cannot access NIS+ tables that are symbolically linked or referenced by paths.

4.16 Setting up Your First NIS+ Domain

After you have diagrammed the organization of users and workstations at your site, you must consider the traffic load of your proposed NIS+ namespace. Ideally, each domain in your namespace will have one master server and a number of replicas. To make each domain efficient, it is recommended that you limit the number of replicas to two. Since each domain may have a different traffic load, you need to calculate the disk space requirements of each master server in your namespace. This calculation is described in the following section.

4.16.1 Calculating Disk Requirements for Your Master Servers

Consider the following factors when determining disk requirements for master servers:

- Disk space for `/etc/nis` (and `/etc/yp`)
- Amount of memory

NIS+ tables, objects, and groups and client information are stored in `/etc/nis`. Typically, `/etc/nis` uses about 5 Kbytes of disk space per client. An NIS+ namespace with 1000 clients requires approximately 5 Mbytes of disk space. It is recommended however, that you add an additional 10 to 15 Mbytes of disk space to handle transaction logs. It is also recommended that you checkpoint transaction logs regularly to reduce their size.

The amount of memory that is required for NIS+ servers is also determined by the size of your NIS+ domains. The minimum requirement is 32 Mbytes. We

recommend that you have up to 64 Mbytes for servers that support large domains.

4.17 How to Set up a Root Domain

To set up an NIS+ root domain on your UNICOS system, perform the following steps:

1. Log into the root master server machine as `root`.
2. Set the domain name of the root master server by using the `domainname` command. For example, if you want to name your first domain `nis.com`, enter the following command line:

   ```
   domainname nis.com
   ```

3. Make sure that the `/etc/nsswitch.conf` file is configured for NIS+ service. See the `nsswitch`(4) man page for more information on the syntax of the `nsswitch.conf` file.
4. Remove any NIS+ files and kill any existing processes by entering the following command lines:

   ```
   rm -rf /etc/nis/*
   ps -aef | grep nis_cachemgr
   kill -9 $PID
   ```

5. Set the administration group for the root domain by exporting the environment variable `NIS_GROUP` for either standard/Korn shell or C shell environments by entering one of the following command lines:

   ```
   export NIS_GROUP=admin.nis.com
   ```

   (for standard or Korn shell)

   ```
   setenv NIS_GROUP admin.nis.com.
   ```

   (for C shell)

6. Initialize the root master server by entering the `nisinit` command as follows:

   ```
   nisinit -r
   ```

7. Terminate any running NIS+ daemons by searching for `nisd` processes and killing their process IDs by using the following command line:

```
ps -aef | grep nisd
kill -9 $PID
```

8. Start the NIS+ daemon by entering the `nisd` command with the `-r` and `-S` options as follows:

   ```
   nisd -r -S 0
   ```

   If you are setting up to run in NIS-compatibility mode, enter the `nisd` command with the `-r`, `-Y`, and `-S` options as follows:

   ```
   nisd -r -Y -S 0
   ```

9. Create NIS+ subdirectories and tables by entering the `nissetup` command as follows:

   ```
   nissetup
   ```

   If you are setting up to run in NIS-compatibility mode, enter the `nissetup` command with the `-Y` option as follows:

   ```
   nissetup -Y
   ```

10. Create the DES credentials by entering the `nisaddcred` command with the `DES` argument as follows:

    ```
    nisaddcred DES
    ```

11. Update the public key for the parent and subdirectories by entering the `nisupdkeys` command as follows:

    ```
    nisupdkeys nis.com.
    nisupdkeys org_dir.nis.com.
    nisupdkeys groups_dir.nis.com.
    ```

    See the `nisupdkeys`(8) man page for more information.

12. Search for and eliminate any NIS+ daemon that is currently running by entering the following command lines:

    ```
    ps -aef | grep nisd
    kill -9 $PID
    ```

13. Restart the `nisd` daemon with security level 2 (the default) by entering the following command:

    ```
    nisd -r
    ```

If you are setting up to run in NIS-compatibility mode, enter the `nisd` command with the `-r` and `-Y` options as follows:

```
nisd -r -Y
```

14. Add root's DES credentials to the root domain by entering the `nisaddcred`(8) command with the `-p` option as follows:

```
nisaddcred -p unix.0@nis.com -P root.nis.com. DES
```

4.18 Initializing an NIS+ Client

Initialize your NIS+ clients by using the following steps:

1. Log in to the domain's master server.
2. Create DES credentials for the new client machine by entering the `nisaddcred` command with the `-p` option as follows:

```
nisaddcred -p unix.mach_name@nis.com -P mach_name.nis.com. DES
```

3. Log in to the client machine as `root`.
4. Assign the domain by entering the `domainname` command as follows:

```
domainname nis.com
```

5. Check that the `/etc/nsswitch.conf` file is configured for NIS+ service.
6. Remove any existing NIS+ files in `/etc/nis` and search for and kill any active processes by entering the following command lines:

```
rm -rf /etc/nis/*
ps -aef | grep nis_cachemgr
kill -9 $PID
```

7. Initialize the NIS+ client by entering the `nisinit` command with the broadcast, host name, or coldstart file option. Choose and enter only one of the following command lines:

```
nisinit -c -B
```

(for broadcast)

```
nisinit -c -H master_server
```

(for host name)

```
nisinit -c -C /tmp/NIS_COLD_START
```

(for coldstart file)

8. Search for and eliminate any existing active processes and restart the `keyserv` daemon by entering the following command lines:

```
ps -aef | grep keyserv
rm -f /etc/.rootkey
keyserv
```

9. Enter the `keylogin` command with the `-r` option as follows:

```
keylogin -r
```

10. Reboot the client.

4.19 Setting up an NIS+ Server

Set up an NIS+ server on your UNICOS system by performing the following steps.

Note: Before a machine is set up as a replica, a server must be set up as the root domain, and the potential replica machine must be initialized as an NIS+ client.

1. Log in as `root` to the machine that you want to be a replica server.
2. Start the NIS+ daemon by entering the `nisd`(8) command as follows:

```
nisd
```

If you are setting up to run in NIS-compatibility mode, enter the `nisd` command with the `-Y` option as follows:

```
nisd -Y
```

3. Start the NIS+ cache manager by entering the `nis_cachemgr`(8) command, as follows:

```
nis_cachemgr
```

4.20 How to Set up a Nonroot Domain

Set up a nonroot domain on your UNICOS system by performing the following steps:

1. Log in to the domain's master server.
2. Set the administration group for the domain by exporting the environment variable `NIS_GROUP` for either standard/Korn shell or C shell environments by entering one of the following command lines:

   ```
   export NIS_GROUP=admin.sales.nis.com.
   ```

   (for standard or Korn shell)

   ```
   setenv NIS_GROUP admin.sales.nis.com.
   ```

   (for C shell)
3. Create the new domain's directory and specify its servers by entering the `nismkdir`(8) command with the `-m` and `-s` options as follows:

   ```
   nismkdir -m master_server -s replica_server sales.nis.com.
   ```

   See the `nismkdir`(8) man page for information about command options and arguments.
4. Create the subdirectories and tables by entering the `nissetup` command as follows:

   ```
   nissetup sales.nis.com.
   ```

   If you are setting up to run in NIS-compatibility mode, enter the `nissetup` command with the `-Y` option as follows:

   ```
   nissetup -Y sales.nis.com.
   ```
5. Create the domain's administration group by entering the `nisgrpadm` command with the `-c` option as follows:

   ```
   nisgrpadm -c admin.sales.nis.com.
   ```
6. Assign group members access rights to the new directory by entering the `nischmod`(8) command with the *rights* argument as follows:

   ```
   nischmod g+rmcd sales.nis.com.
   ```

   See the `nischmod`(8) man page for details about the content and syntax of the *rights* argument.
7. Add master and replica servers to the domain's administration group by entering the `nisgrpadm` command with the `-a` option as follows:

```
nisgrpadm -a admin.sales.nis.com. master_server.nis.com.
        replica_server.nis.com.
```

4.21 Administering Your NIS+ Namespace

Table 10 shows NIS+ commands.

Table 10. Common NIS+ commands

Task	Command line
For NIS+ groups:	
Create a group	`nisgrpadm -a` *groupname.domainname*
List members of a group	`nisgrpadm -l` *groupname*
Delete a group	`nisgrpadm -a` *groupname*
Add members to a group	`nisgrpadm - a` *groupname members* `...`
For NIS+ clients:	
Initialize an NIS+ client	`nisinit -c -H` *hostname*
Initialize an NIS+ client	`nisinit -c -B`
Initialize an NIS+ client	`nisinit -c -C` *coldstart filename*
Display all information about a workstation	`nisdefaults`
For NIS+ servers:	
Initialize the root master server	`nisinit -r`
Start a cache manager	`nis_cachemgr`
Checkpoint a database	`nisping -C` *domainname* `.com`
Display the contents of the transaction log	`nislog`

Task	Command line
For NIS+ tables:	
Display the contents of a table	`niscat -h -A` *tablename*
Create a table	`nistbladm -c` *table-type column-spec... tablename*
Delete a table	`nistbladm -d` *tablename*
Add an entry to a table	`nistbladm -a` *entry*
Modify a table entry	`nistbladm -m` *new-entry old-entry*
Remove a table entry	`nistbladm -r` *indexedname*
Load information into tables from text files	`nisaddent -t` *filename table type*
Load information into tables from NIS maps	`nisaddent -t` *NISdomain table type*
Display the object properties of a table	`niscat -o` *tablename* `.org_dir`
Search for regular expressions in NIS+ tables	`nisgrep` *expression tablename*
Search for strings in NIS+ tables	`nismatch` *string tablename*
Change the time to live value of table entries	`nischttl` [*column* `=` *value*], *tablename*
For NIS+ objects:	
Expand an existing NIS+ directory into a domain	`nissetup` *directory*
Create a link between NIS+ objects	`nisln` *source target*
Display the contents of a client's directory cache	`nisshowcache`
Display the time-to-live value of an object	`nisdefaults -t`
Change the time-to-live value of NIS+ objects	`nischttl` *time-to-live object name*
For NIS+ security:	
Set the security level of service	`nisd -S` *level*
Add credentials for a new user	`nisaddcred -p` [`-P`]
Remove credentials	`nisaddcred -r` *principal name*

Task	Command line
Update or change your credentials	`nisaddcred` *credential-type*
Add access rights for an object	`nischmod` *category +right object name*
Add access rights for an entry	`nischmod` *category +right* [*column-name* = *value*],*tablename*
Display all local domain password information	`nispasswd -a`
Display password information for a specific user	`nispasswd -d` *username*
Clear public keys stored by a directory	`nisupdkeys -C` *directory*
Update the IP addresses of a server	`nisupdkeys -a -H` *server-name*

4.22 Migrating from NIS to NIS+

NIS+ can handle requests from both NIS version 2 and NIS+ clients. NIS+ has an NIS built-in compatibility mode that provides two service interfaces to answer NIS and NIS+ requests transparently. Although there is no additional setup required to run an NIS+ server in NIS-compatibility mode, the instructions for setting up in this mode are different than setting up a standard NIS+ server. An NIS server running in NIS-compatibility mode does not support the `ypupdate` and `ypxfr` protocols and, therefore, cannot be used as a master or slave NIS server.

Note: NIS servers do not provide any of the user or request credentials that NIS+ servers expect from their clients. Therefore, all NIS+ tables must have access rights set to allow unauthenticated requests in order for NIS client requests for information to be accepted and fulfilled by NIS+ servers.

4.22.1 NIS-compatibility Mode

NIS+ uses fewer tables than NIS. Determine the information sources that you need to load by examining the mapping of local files, NIS maps, and NIS+ tables in the following table.

Table 11. Correspondence between information sources on a UNICOS system

Local files	NIS maps	NIS+ tables
auto.home		auto_home
auto.master		auto_master
/etc/ethers		ethers
/etc/group	group.bygid	group
/etc/group	group.byname	group
/etc/hosts		hosts
/etc/netgroup	netgroup	netgroup
/etc/netgroup	netgroup.byhost	netgroup
/etc/netgroup	netgroup.byuser	netgroup
		cred
/etc/networks		networks
/etc/passwd	passwd.byname	passwd
/etc/passwd	passwd.byuid	passwd
/etc/protocols		protocols
/etc/rpc		rpc
/etc/services		services

4.22.2 NIS to NIS+ Command Compatibility

The following table lists the NIS and NIS+ commands supported on a UNICOS system.

Table 12. Comparing NIS and NIS+ commands on a UNICOS system

NIS server	NIS-compatible server	NIS+ server
makeddm	---	nistbladm, nisaddent
ypbind	ypbind	---

NIS server	NIS-compatible server	NIS+ server
ypcat	ypcat	niscat
ypwhich -m	ypwhich -m	niscat -o
ypinit -m	ypinit -c	---
ypinit -s	---	---
ypmake	---	nissetup, nisaddent
ypmatch	---	nismatch, nisgrep
yppasswd	---	nispasswd
yppush	---	nisping
yppoll	---	---
ypserv	nisd -Y	nisd
ypset	---	---
ypxfr	---	---