Warranties and Liabilities

Trademarks

Transarc Corporation
The Gulf Tower
707 Grant Street
Pittsburgh, PA 15219

Appendix C
Managing the NFS/AFS Translator

The NFS®/AFS® Translator is a separate-purchase product that allows users working on NFS client machines to access, create and remove files stored in AFS. This chapter assumes the reader is familiar with both NFS and AFS.

C.1 Summary of Commands C-1
C.2 Overview C-2
C.3 Setting Up an NFS/AFS Translator Machine C-7
C.4 Setting Up an NFS Client C-11
C.5 Configuring a User Account to Enable Issuing AFS Commands C-14
C.6 Authenticating from Non-Supported NFS Client Machines C-16
C.7 Disabling and Enabling a Translator Machine C-19

C.1. Summary of Commands

Mount directory on translator machine . . . . . **mount**
Examine value of *@sys* . . . . . **fs sysname**
Authenticate from NFS client of a non-supported system type . . . . . **knfs**
Enable/disable translator machine . . . . . **fs exportafs**

C.2. Overview

The NFS/AFS Translator allows NFS and AFS users to collaborate and share AFS files. With the Translator, users on NFS client machines can create, remove and access AFS files to the same extent as users on AFS client machines.

An ***NFS/AFS translator machine*** (or simply ***translator machine***) is a machine configured as both an NFS server and an AFS client:

- Its NFS server functionality makes it possible for the translator machine to export its */afs* directory (the entry point to AFS's multi-cellular name space) to NFS client machines. When the NFS clients mount */afs* via the translator machine, they have immediate and transparent access to AFS. Note that the NFS client machines do not need to run AFS software.
- Its AFS client functionality makes the */afs* directory available. The Cache Manager on the translator machine requests and caches files from AFS file server machines. The Cache Manager also maintains tokens for NFS users, if you have made the configuration changes necessary for NFS users to authenticate in AFS.

C.2.1. Levels of Access to AFS

You can provide three levels of AFS access to your users working on NFS client machines.

- Users working on NFS client machines do not authenticate with AFS, and so are considered to be the unprivileged user *anonymous*. They can exercise only the file system rights granted to *system:anyuser*. They can issue only those AFS commands for which no privilege is required, and then only if they are working on a supported system type (an NFS client machine of a system type for which AFS binaries are available). The binaries must be accessible by *anonymous*. Such users presumably do not have AFS accounts.
- Users working on unsupported system types (NFS client machines for which AFS binaries are not available) can authenticate with AFS over a connection with their translator machine, and so can safely be granted the same file system rights as any AFS user. The users presumably have AFS accounts, but they cannot issue AFS commands on the NFS client because binaries are not available. To authenticate with AFS, the users issue the **klog** and **knfs** commands on the translator machine to make its Cache Manager use the tokens correctly while they work on the NFS client. Section C.6 describes how to use **knfs**.
- Users working on NFS client machines for which AFS binaries are available can authenticate with AFS by issuing **klog** on the NFS client machine, and can issue the same AFS commands as users on AFS client machines. Because they have AFS accounts and are authenticated, such users can safely be granted the same file system rights as any AFS user.

Sections C.2.2 through C.2.6 provide detailed conceptual information about the configuration changes you must make to enable the different types of access.

Sections C.3 through C.5 provide instructions for making the configuration changes.

C.2.2. NFS/AFS Translator Machine Requirements

The following sections describe the hardware and software requirements for an NFS/AFS translator machine.

C.2.2.1. Translator Machine Hardware

An NFS/AFS translator machine must be configured as both an NFS server machine and an AFS client machine. Currently, all AFS-supported system types can serve as translator machines, except the Ultrix system types (*pmax_ul43*, *pmax_ul43a* and *vax_ul43*). Ultrix does not permit conversion between two remote file systems.

C.2.2.2. Translator Machine Software

You can control the amount of AFS access available to users on NFS clients by configuring the translator machine differently.

- With the simplest possible configuration, users can access AFS at the first two levels described in section C.2.1. They cannot issue AFS commands.
- A slightly more involved set of configuration changes enables users to issue AFS commands on NFS clients for which AFS binaries are available. To make this possible, the NFS/AFS translator machine must run a special daemon that executes AFS-related system calls on behalf of remote NFS client machines.

Section C.3 describes how to configure a translator machine.

C.2.3. NFS Client Machine Requirements

The following sections describe the hardware and software requirements for an NFS client machine.

C.2.3.1. NFS Client Hardware

Any NFS client machine can use the NFS/AFS Translator to access files in AFS. To enable users on NFS clients to issue AFS commands, however, the NFS client must be a system type for which AFS binaries are available. The NFS client does not need to be configured as an AFS client machine.

C.2.3.2. NFS Client Software

You can control the amount of AFS access available to users on an NFS client machine by configuring the machine differently.

- With the simplest possible configuration, users can access AFS at the first two levels described in section C.2.1. They cannot issue AFS commands.

- A slightly more involved set of configuration changes enables users on NFS clients to issue AFS commands. As mentioned above, the machine must be a system type that Transarc supports as an AFS client. Also, the `AFSSERVER` and `AFSCONF` environment variables must be defined. The definitions can be stored either on the NFS client machine or in each user's home directory; see section C.2.5.

Section C.4 describes how to configure an NFS client machine to use the NFS/AFS Translator.

C.2.4. User Account Requirements

For users to

- authenticate with AFS, they must have entries in the Protection and Authentication Databases.
- create and store files in their own home directory, they must have a volume mounted in the AFS file tree.
- issue AFS commands while working on an NFS client machine for which AFS binaries are available, the `AFSSERVER` and `AFSCONF` environment variables must be defined. The definitions can be placed in files in the user's home directory, if appropriate. See section C.2.5.

Chapters 16 and 17 of the *AFS System Administrator's Guide* describe how to create the components of an AFS user account.

C.2.5. The AFSSERVER and AFSCONF Environment Variables

Setting the `AFSSERVER` and `AFSCONF` environment variables allows users to issue AFS commands when working on NFS client machines for which AFS binaries are available.

Users working on NFS client machines for which AFS binaries are not available can authenticate using the **knfs** command, but cannot issue AFS commands. It is not necessary to define `AFSSERVER` and `AFSCONF` for such users. Section C.6 describes how to use **knfs**.

C.2.5.1. The AFSSERVER variable

This variable designates the AFS client machine that executes AFS-specific system calls on behalf of the NFS client machine. The AFS client is thus referred to as the ***remote executor***. It also acts as the Cache Manager for users working its NFS clients, storing their AFS tokens so that AFS will recognize the user as authenticated. Thus, the two machines (the remote executor and the translator) must be the same.

The choice of remote executor affects commands that influence or report on Cache Manager performance or configuration, such as **fs getcacheparms**, **fs getcellstatus**, **fs newcell** and **fs setcell**. When issued on an NFS client, these commands affect the Cache Manager on the remote executor machine. Several of these commands require the issuer to be logged into the remote executor's UNIX file system as "root".

The ability of NFS client users to log in as "root" is controlled by NFS, not by the NFS/AFS Translator, so setting the remote executor properly is not enough to allow an NFS client user to issue such commands.

The choice of remote executor is irrelevant for AFS commands of two types: those whose results are the same no matter which machine they originate from (such as **fs listacl**), and those that do not require privilege (such as the unprivileged commands in the **bos**, **kas**, **pts** and **vos** suites).

C.2.5.2. The AFSCONF variable

The `AFSCONF` environment variable specifies which directory the NFS client machine should consult to find the *ThisCell* and *CellServDB* files when executing AFS commands. As on an AFS client, these files determine the default cell for command execution.

If your cell has an AFS directory that serves as the central "update source" for the files in the */usr/vice/etc* directory on client machines, it is recommended that you set `AFSCONF` on all of the client machines to refer to that directory. Normally, this directory is called */afs/**cellname**/common/etc*.

C.2.5.3. How the NFS Client Defines the Variables

To get values for `AFSSERVER` and `AFSCONF`, the NFS client consults the following three places in sequence:

1. the current command shell's environment variable definitions
2. the *.AFSSERVER* or *.AFSCONF* file in the user's home directory
3. the */.AFSSERVER* or */.AFSCONF* file in the NFS client machine's root (*/*) directory. If the client does not have a disk, the root directory resides on an NFS server machine.

(Actually, the NFS client looks for *CellServDB* and *ThisCell* in its own */usr/vice/etc* directory, just as an AFS client does, before looking for a definition for `AFSCONF`. If that directory exists, the NFS client does not attempt to define `AFSCONF`. Most NFS clients are not also AFS clients, so */usr/vice/etc* does not exist, and `AFSCONF` must be defined by consulting the three sources shown.)

Where you should define the variables depends on whether most users usually work on the same NFS client machine or not.

- If users typically work on the same NFS client machine, create *.AFSSERVER* and *.AFSCONF* files in the root (/) directory on that machine.
- If users typically work on different NFS client machines at different times, create *.AFSSERVER* and *.AFSCONF* files in their home directories.

C.2.6. Delayed Write Semantics

To avoid degrading the performance of a Cache Manager that is serving NFS clients, AFS uses a "delayed write" mechanism for AFS files saved from an NFS client machine. One of the Cache Manager's daemons knows how to identify AFS files accessed from NFS clients. Rather than executing requested write operations on them immediately, it executes them every 30 seconds.

The delayed write mechanism implies that it can take up to 30 seconds for AFS data saved on an NFS client to be visible to users on other AFS client machines, or users on NFS clients that are using a different translator machine.

C.2.7. About Building the Translator from Source

As of January 1993, source code for the NFS/AFS Translator is no longer available, except for Solaris systems. If you wish to build the Translator for any systems other than Solaris, Transarc will provide an object version of *afs_nfssrv.c*. Contact your AFS Product Support Representative.

C.3. Setting Up an NFS/AFS Translator Machine

An NFS/AFS translator machine must:

- Be configured as an AFS client. All system types supported as AFS clients can be translator machines, except those that run Ultrix (*pmax_ul43*, *pmax_ul43a* and *vax_ul43*).
- Be configured as an NFS server. The appropriate number of NFS server daemons (**nfsd** and others) depends on the anticipated machine load.
- Export its */afs* directory.

The following sections explain further requirements for translator machines that allow their clients to issue AFS commands, and for certain system types.

C.3.1. General Requirements for Loading NFS and AFS Kernel Extensions

In general, you must load the NFS kernel extensions before loading the AFS kernel extensions. On system types where the NFS kernel extensions are already part of the kernel, it is not necessary to load the NFS kernel extensions.

Note: The order in which the respective kernel extensions are loaded is irrelevant on machines running Solaris 2.2 (system types *sun4_52*, *sun4c_52*, and *sun4m_52*).

After loading the kernel extensions, you can run the **bosserver** command if the AFS/NFS translator machine is also an AFS file server machine. Finally, you should issue the **afsd** and **nfsd** commands.

C.3.2. Requirements for NFS Client Users to Issue AFS Commands

To enable users of its NFS clients to issue AFS commands, an NFS/AFS translator machine must run **afsd** (the program that initializes the Cache Manager) with the **-rmtsys** flag. This flag invokes an additional daemon that accepts and executes AFS-specific system calls from the machine’s remote NFS clients. Remember to add the **-rmtsys** flag to the **afsd** command line that appears in the machine’s initialization file.

Using the **-rmtsys** flag enables NFS users using this translator machine to examine the configuration of its Cache Manager using AFS commands like **fs getcacheparms**. Users can only change the Cache Manager configuration by logging onto the translator machine’s UNIX file system as "root." Administrators should use NFS mechanisms to limit access to the root account.

Since daemon started by **-rmtsys** does not require many additional resources, it is recommended that you run it on all translator machines for convenience.

C.3.3. Requirements for AIX 3.2 Systems

To function properly as translator machines, AIX 3.2 systems (system type *rs_aix32*) require the following. More details on the commands and files discussed here appear in the section entitled "Using the Kernel Extension Facility on AIX 3.2 Systems" in chapters 2 and 4 of the *AFS Installation Guide*.

NFS must already be loaded into the kernel before you load AFS modifications with the **cfgafs** and **cfgexport** commands. For AIX version 3.2.2 and earlier, load the NFS extensions by loading *nfs.ext*; AIX version 3.2.3 and later is distributed with NFS extensions already in the kernel.

You must substitute *afs.ext.trans* for *afs.ext* when you issue the **cfgafs** command during initial installation. If the machine is already running AFS, you must reboot and make the substitution now. This is the substitution referred to in the *AFS Installation Guide* section of chapters 2 and 4 entitled "Using the Kernel Extension Facility on AIX 3.2 Systems."

Calls to *nfs.ext* (if used), **afsd**, and **nfsd** must appear in that order in the machine's initialization files. As explained further in the *AFS Installation Guide* section on "Using the Kernel Extension Facility on AIX 3.2 Systems," IBM delivers several function-specific initialization files for the AIX 3.2 system, rather than the single file used on some other systems. The following list shows the correct ordering for commands in these files:

- Commands to load *nfs.ext*, for AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel.
- Commands to load the AFS kernel extensions. Use the contents of */usr/vice/etc/dkload/rc.dkload.client* to invoke the kernel extension facility on AFS client machines. Be sure to substitute ***afs kernel extensions*** (*afs.ext.trans*) for *afs.ext* in the **cfgafs** command here as you did at initial installation.
- **bosserver**, if the machine will function as an AFS file server machine.
- **afsd**.
- **nfsd**.

C.3.4. Requirements for AOS 4.3 and AIX 2.2.1 Systems

System types *rt_aix221* and *rt_aos4* must run the Transarc-supplied version of the *rpc.mountd* daemon. This modified version allows NFS mounts to directories as well as partitions. The ability to mount directories is standard for all other operating system types that are running recent versions of NFS.

System type *rt_aix221* must use the Transarc-supplied version of *libnfs*. For this system type, the vendor-supplied version of *libnfs* causes errors. The Transarc version is found in the fifth **tar** set on the Binary Distribution tape as */usr/afsws/root.client/sys/libnfs*.

⇒ To configure an NFS/AFS translator machine

Note: The following instructions set up the translator so that users can issue AFS commands. Omit step 5 if you do not want to enable this functionality.

Step 1: Configure the NFS/AFS translator machine as an NFS server, if it is not already. Follow the instructions provided by your NFS supplier. The number of NFS server daemons (such as **nfsd**) should be appropriate for the number of potential NFS clients.

Step 2: Configure the NFS/AFS translator machine as an AFS client, if it is not already. For the most predictable level of performance, the translator machine's local copies of *CellServDB* and *ThisCell* (in */usr/vice/etc*) should match the files in the cell's official central update directory, if there is one (recommended location is */afs/**cellname**/common/etc*).

Step 3: Become "root" on the NFS/AFS translator machine, if you are not already. Issue the **su** command, if necessary.

% **su root**
`Password:` <*root password*>

Step 4: On Solaris systems (system types *sun4_52*, *sun4c_52* and *sun4m_52*), modify the */etc/dfs/dfstab* file to include a **share** command that allows remote mounts of */afs*; the **share** binary is standardly distributed as */usr/sbin/share*. The following example commands allow remote mounts of the root partition (/) and of */afs*. Using the *unshare* command disables remote mounts. Consult the manual pages for *share* and *unshare* for more information and complete syntax.

```
share -F nfs -o rw -d "root" /
share -F nfs -o rw -d "afs gateway" /afs
```

On all system types except Solaris, modify the */etc/exports* file to include a line that allows export of the */afs* directory to NFS clients. If you want to restrict access to */afs*, include only the names of authorized NFS clients. For a description of the proper format of the */etc/exports* file, see the NFS manual page for *exports(5)*.

The following example would allow any NFS client to mount */afs*, */usr*, and */usr2* on this machine:

```
/afs
/usr
/usr2
```

Step 5: Modify the **afsd** command in the machine's boot-time initialization file to include the **-rmtsys** flag. As described in section C.2.2.2, the **-rmtsys** flag starts up a daemon on the machine to execute remote AFS-specific system calls issued by NFS clients.

For all system types other than IRIX, add the `-rmtsys` flag to the `afsd` command line in */etc/rc* or equivalent (*/etc/init.d* for Solaris systems). For example:

```
/usr/vice/etc/afsd -rmtsys
```

For IRIX systems, activate the file */etc/config/afsxnfs*.

% **cd /etc/config**

% **/etc/chkconfig -f afsxnfs on**

Depending on the number of NFS clients you expect this machine to serve, you may also wish to add other switches to **afsd** that start up performance-enhancing AFS "background daemons" and/or set configuration parameters. See the page on **afsd** in the *AFS Command Reference Manual*.

Step 6: Reboot the machine. On many system types, the appropriate command is **/etc/reboot**; consult your UNIX administrator's guide.

/etc/reboot

C.4. Setting Up an NFS Client

To mount AFS, an NFS client machine must:

- Have an */afs* directory on which to mount the */afs* file system exported by the translator machine.

  The directory on which an NFS client mounts */afs* can actually be called something other than */afs*. This is handy if the NFS user wishes to mount more than one translator machine's */afs* directory. If one translator machine goes down, the user can then quickly switch to another translator machine. However, AFS commands that include or require a pathname to a file or directory will not work properly unless AFS is mounted at */afs* on the translator machine. For example, if AFS were mounted at */afs2* on a translator machine, a pathname to a directory in AFS provided with the **-dir** switch on the **fs setacl** command would not be properly resolved and the command would fail.

- Create an NFS mount of the translator machine's */afs* directory onto the local */afs* directory. The mount command should be added to the */etc/fstab* file (or equivalent) on the NFS client.

- Run an appropriate number of NFS client *biod* daemons, which improve performance by handling pre-reading and delayed writing. Most NFS suppliers recommend running four such daemons, and most NFS start-up scripts initialize them automatically. Consult your NFS documentation.

To mount AFS and enable users to issue AFS commands, an NFS client machine must also:

- Be a system type supported by AFS; AFS command binaries must be available on the NFS client (if you are not sure a system type is supported, consult the latest *AFS Release Notes* or *AFS Installation Guide*, or contact your AFS Product Support Representative). The machine does not need to be configured as an AFS client.

- Define an `AFSSERVER` environment variable that defines the ***remote executor***, as described in section C.2.5. If users will issue commands that involve tokens (such as **klog**, **tokens**, and **unlog**), the remote executor must be the same as the translator machine whose */afs* directory (or equivalent) is mounted.

- Define an `AFSCONF` environment variable that defines which *CellServDB* and *ThisCell* files to consult (described in section C.2.5).

⇒ To configure an NFS client machine to access AFS

Note: The following instructions set up the NFS client machine so that users can issue AFS commands. Omit steps 5 and 6 if you do not want to enable this functionality.

Step 1: Configure the machine as an NFS client machine, if it is not already. Follow the instructions provided by your NFS supplier. The number of NFS client (*biod*) daemons should be appropriate for the expected load on this machine. The usual recommended number is four.

Step 2: Become "root" on the NFS client machine, if you are not already. Issue the **su** command, if necessary.

```
% su root
Password: <root password>
```

Step 3: Create an */afs* directory on the client if one does not already exist. Other directories can be used to mount */afs* also, as described previously.

\# **mkdir /afs**

Step 4: Modify the machine's "file systems registry" file to mount a translator machine's */afs* directory. On many system types, the registry file is */etc/fstab*, but see the section entitled "Setting Up AFS Partitions" in chapter 2 of the *AFS Installation Guide* for more information. The mount command differs from system to system. Be sure to consult the **mount(5)** man page to learn the proper syntax. The following example shows the suggested options.

On many systems the appropriate line is

```
mount -o hard,intr,timeo=300 <translator>:/afs /afs
```

On AIX 2.2.1, the corresponding line is

```
mount -o hard,intr,timeo=300 -v nfs -n <translator> /afs /afs
```

where

`hard` indicates that the NFS client should retry NFS requests until the translator machine (NFS server) responds. Because requests must travel through two file systems, they may take longer than usual. With a "soft" mount, a delayed response from the translator machine could cause the request to abort. Many NFS versions use hard mounts by default; if your version does not, we recommend that you add this option.

`intr` allows the user to use a keyboard interrupt signal (such as **Ctrl-C**) to "escape" when the translator machine has gone down and made work impossible. Include this option only if the `hard` option is used, since the connection will not automatically break off when a translator machine goes down.

`timeo` sets the maximum time (in tenths of seconds) the translator can take to respond to the NFS client's request before the client considers the request timed out. With a hard mount, setting this option to a high number like 300 reduces the number of messages like

```
NFS server translator is not responding, still trying
```

the user sees when the translator does not respond immediately. With a soft mount, it reduces the number of actual errors returned on timed out requests.

translator names the NFS/AFS translator machine whose */afs* directory should be mounted on the client machine's */afs* directory. Specifying the translator machine's full Internet-style host name is recommended.

If you wish to mount the translator machine's */afs* directory onto a directory other than */afs*, substitute the alternate directory name for the second instance of */afs*.

Step 5: If appropriate, create the */.AFSSERVER* file to define the value of the `AFSSERVER` environment variable. The file should contain a single line specifying the complete Internet-style name of the translator machine that will serve as the remote executor. To enable users to issue commands that handle tokens, define the remote executor to be the machine named as *translator* in step 4.

Section C.2.5 explains when it is appropriate to create this file, as opposed to defining `AFSSERVER` for each user.

Step 6: If appropriate, create the */.AFSCONF* file to define the value of the `AFSCONF` environment variable. The file should contain a single line specifying the directory in which the NFS client machine should read *CellServDB* and *ThisCell*. If the cell has a central update source for these files, normally */afs/**cellname**/common/etc*, this should name it.

Section C.2.5 explains when it is appropriate to create this file, as opposed to defining `AFSCONF` for each user.

C.5. Configuring a User Account to Enable Issuing AFS Commands

To issue AFS commands from an NFS client, a user must:

- Be working on a system type supported by AFS, and be able to access the AFS binaries (presumably available in the cell's AFS tree).
- Have an entry in the AFS Protection and Authentication Databases, in order to issue the **klog** command (or any command that requires being authenticated). Chapters 16 and 17 explain how to create user accounts.
- Include the pathname of the appropriate AFS binaries in their `PATH` environment variable. If the user will use NFS client machines of different system types, the pathname should probably use "*@sys*" in place of a specific system type.
- Have files in the home directory to set the `AFSSERVER` and `AFSCONF` environment variables, if appropriate. This is appropriate if the user typically works on different NFS client machines. The existence of these files prevents use of the global definitions in the NFS client machine's root (/) directory.

To have a complete AFS account, a user must:

- Have a volume mounted as his or her home directory in the cell's AFS file tree. Note that this is necessary in order to set values for the `PATH`, `AFSSERVER`, and `AFSCONF` environment variables in initialization files.

⇒ To configure a user account

Step 1: Create entries for the user in the Protection and Authentication Databases, or create a complete AFS account. See the instructions for account creation in section 17.5 or for **uss** in chapter 16.

Step 2: Modify the user's `PATH` variable to include the pathname of AFS binaries, such as */afs/**cellname**/**systype**/usr/afsws/bin*. If the user will be working on NFS clients with different system types, considering replacing the specific ***systype*** with *@sys*. The `PATH` variable is commonly defined in a login or shell initialization file (such as *.login* or *.cshrc*).

Step 3: If appropriate, create *.AFSSERVER* and *.AFSCONF* files in the user's home directory, defining values for the `AFSSERVER` and `AFSCONF` environment variables respectively. This is most appropriate if the user typically works on different NFS client machines at different times. The values defined in these files prevent use of the values defined in the NFS client machine's root (/) directory. If used, the *.AFSSERVER* file should contain a single line specifying the complete Internet-style name of the translator machine that will serve as the remote executor. The *.AFSCONF* file should contain a single line specifying the directory in which the NFS client machine should read *CellServDB* and *ThisCell*.

Step 4: If the pathname you defined in step 2 includes *@sys*, notify users that they should check to make sure their system name is defined correctly before issuing AFS commands. The definition can be checked with the **fs sysname** command as follows:

% **fs sysname**

The output from this command uses the standard Transarc names for system types. For example, on an IBM RS/6000 running AIX 3.2 the output would read

```
Current sysname is 'rs_aix32'
```

C.6. Authenticating from Non-Supported NFS Client Machines

The **knfs** command allows users to authenticate with AFS when they are working on NFS clients of system types for which AFS binaries are not available (i.e., unsupported system types). This allows such users to access the AFS file tree to the same extent as any other AFS user. They cannot, however, issue AFS commands, which is still possible only on NFS client machines of system types for which AFS binaries are available.

The user first logs into the translator machine through which AFS will be accessed, and obtains tokens for all the cells he or she wishes to contact during the upcoming NFS login session. He or she then issues the **knfs** command on the translator machine. The command creates a new credential structure on the translator machine for the AFS tokens already obtained. Depending on which of **knfs**’s arguments the user provides, the PAG that identifies the credential structure is associated with

- The issuer (and the issuer only) when he or she is working on the NFS client machine. To obtain this kind of credential structure, the issuer of **knfs** specifies both the NFS client machine’s name and his or her UID number on the NFS client. The Cache Manager uses the tokens in this type of credential structure only for someone who logs into the NFS client under that UID. It can therefore be described as a "personal" credential structure.
- The NFS client machine. To obtain this kind of credential structure, the issuer of **knfs** specifies only the NFS client machine’s name. The Cache Manager uses the tokens in this type of credential structure for anyone working on the NFS client who does not have a personal credential structure (anyone who issues **knfs** without providing a UID number, or does not issue **knfs** at all). This type of credential structure is therefore described as "generic".

C.6.1. Security Concerns about Generic Credential Structures

Use of generic credential structures is discouraged, because it creates security problems. They are available mostly to accommodate single-user NFS client machines and/or machines that do not identify users with UIDs. On all other types of NFS client machine, all users should obtain a personal credential structure by providing their NFS-client UID to **knfs**.

The problem with a generic credential structure is that it means that many different users on the NFS client machine might be sharing the same tokens. As mentioned above, all users who do not obtain their own credential structure use the generic credential structure—this applies to users who do not specify a UID with **knfs**, and to users who do not issue **knfs** at all.

As an example, suppose that *terry* is working on the NFS client *nfs1.transarc.com* and issues **knfs** without a UID on the translator machine *trans1.transarc.com*. The Cache Manager on *trans1* places *terry*’s tokens into the generic credential structure and uses them for every other user on *nfs1* who issued **knfs** without a UID, or didn’t issue **knfs** at all. This potentially means that all of those users could read and alter *terry*’s files. Suppose further that while *terry* is working on *nfs1*, *pat* is also working there and issues **knfs** without a UID on *trans1*. Now, *pat*’s tokens replace *terry*’s in the generic credential structure, so *terry* (and everyone else on *nfs1* without his or her own credential structure) no longer accesses files as *terry*, but rather as *pat*.

A similar problem arises when someone using the credential structure discards the tokens in it (by adding the **-unlog** flag on **knfs**; see the instructions below). Everyone else using the generic credential structure becomes unauthenticated at the same time.

From the standpoint of security, the token sharing implied by use of the generic credential structure is acceptable only on single-user machines, or if it is acceptable that a group of people have the same access to AFS files. Users who are using a generic credential structure should be warned that they may be sharing tokens with others.

C.6.2. A Note on Discarding Tokens with knfs

Using the **-unlog** flag on **knfs** does not allow a user who has a personal credential structure to start using the generic credential structure. This is because when the Cache Manager receives a file request from a user on an NFS client, it first checks if there is a personal credential structure associated with that user and that machine. If the credential structure exists, then the Cache Manager uses it. Only if there is no personal credential structure does the Cache Manager use the generic credential structure.

When a user uses the **-unlog** flag on **knfs** to discard tokens from a personal credential structure, the Cache Manager on the translator machine destroys the tokens but not the credential structure, which it retains until the machine is rebooted. When the user later accesses AFS through the translator machine, the Cache Manger uses his or her existing personal credential structure. If the credential structure does not have tokens in it, the user accesses AFS unauthenticated rather than getting to use the generic credential structure.

⇒ To authenticate on a NFS client machine of a non-supported type

Step 1: Establish a login session (e.g., on the console or through **telnet**) on the AFS client machine serving as the translator machine for the NFS client.

Step 2: Obtain tokens for every cell you wish to access from the NFS client during the NFS session. The AFS **login** program accomplishes this for the translator machine's cell; use **klog** to obtain tokens for other cells.

Step 3: Issue the **knfs** command to tell the Cache Manager on the translator machine which NFS client machine and user to associate with the newly obtained tokens.

% **knfs** <*host name*> **[**<*user ID*>**]**

where

host name names the NFS client machine you are working on. As with all AFS commands, it is safest to provide the machine's full name, but abbreviations may be acceptable depending on the state of the cell's name server at the time the command is issued.

user ID specifies your UID number on the NFS client (the equivalent of the UNIX UID). This is the UID that the NFS client passes to the translator machine to identify you. One way to learn your UID is to consult your entry in the local */etc/passwd* file.

Step 4: Close the connection to the translator machine (if desired) and work on the NFS client machine as normal.

⇒ To discard tokens using knfs

Step 1: If you closed the connection to the translator machine after issuing **knfs**, re-establish it (perhaps on the console or through **telnet**).

Step 2: Issue the **knfs** command with the **-unlog** flag, to tell the Cache Manager on the translator machine to discard your tokens.

% **knfs** <*host name*> **[**<*user ID*>**] -unlog**

where

host name names the NFS client machine you are working on. As with all AFS commands, it is safest to provide the machine's full name, but abbreviations may be acceptable depending on the state of the cell's name server at the time the command is issued.

user ID specifies your UID number on the NFS client (the equivalent of the UNIX UID). This is the UID that the NFS client passes to the translator machine to identify you. One way to learn your UID is to consult your entry in the local */etc/passwd* file.

-unlog instructs the Cache Manager to discard your tokens.

Step 3: If desired, close the connection to the translator machine.

C.7. Disabling and Enabling a Translator Machine

You can disable/enable NFS clients from mounting a translator machine's */afs* directory by issuing the **fs exportafs** command. You must be logged into the translator machine's UNIX file system as the super-user "root." Note that a normally installed AFS client machine allows export of its */afs* directory (the state achieved by issuing **fs exportafs** with the **on** argument). Under normal circumstances, you would need to use the **on** argument only if the **off** argument was previously used.

⇒ To disable a Translator machine temporarily

Step 1: Become "root" on the NFS/AFS translator machine, if you are not already. Issue the **su** command, if necessary.

`%` **su root**
`Password:` <*root password*>

Step 2: Disable the translator machine from exporting AFS.

\# **fs exportafs nfs off**

NFS clients can no longer mount this machine's */afs* directory.

⇒ To re-enable a disabled Translator machine

Step 1: Become "root" in the translator machine's UNIX file system, if you are not already, by issuing the **su** command.

`%` **su root**
`Password:` <*root password*>

Step 2: Enable the translator machine to export AFS once again.

\# **fs exportafs nfs on**

NFS clients can once again mount this machine's */afs* directory.