1. Installation Overview

This chapter describes pre-installation requirements for AFS®, lists the system types supported by the current version, and explains how to unload the contents of the Binary Distribution.

Before beginning the installation of your cell's first machine, you should read at least this chapter and chapter 2, as well as the material listed in section 1.2.2. Similarly, you should read through the appropriate chapters in this Guide before installing additional file server or client machines. If you are unfamiliar with basic AFS concepts, read chapter 1 of the *AFS System Administrator's Guide*, "An Overview of AFS for the System Administrator," now, before proceeding.

If you are already running a version of AFS, contact Transarc® Corporation before proceeding with the installation.

1.1. Overview

The information in this guide describes procedures for two types of installation tasks: initial tasks (such as installing the first AFS machine, or incorporating AFS into a kernel) and as-needed tasks (such as installing additional AFS file server machines or client machines).

1.1.1. Initial Tasks

Incorporating AFS Into the Kernel

AFS must be incorporated into the kernel of all AFS file server and client machines, either using a dynamic kernel loader or during a kernel build. Chapters 2, 3 and 4 include instructions for incorporating AFS into the kernel on the different types of machines you will install. Chapter 5 provides more detailed instructions to be used for reference.

Installing the First AFS Machine

The first AFS machine will be installed as both an AFS file server and client. First, you will install AFS into the kernel and then will install the AFS file server software.

As the first file server in your cell, this machine will take several "roles," including

- ***system control machine***, distributing certain configuration files to other file servers in the cell.
- ***binary distribution machine*** for its system type, distributing AFS binaries to other file servers of its system type.
- ***database server machine***, running the server processes that maintain the AFS administrative databases.

After you install file server functionality, you will install AFS client functionality and complete other tasks specific to the first machine, including setting up the top levels of your cell's AFS tree.

1.1.2. As-needed Tasks

Installing Additional AFS File Server Machines

To install additional file servers, follow the procedures in Chapter 3. This chapter also provides instructions for adding and removing database server functionality from file server machines.

Installing Additional AFS Client Machines

To install additional client machines, follow the procedures in Chapter 4.

Other Tasks

To incorporate AFS during a kernel build, see chapter 5. This chapter also provides a detailed reference on dynamically loading AFS into the kernel.

To install **package** files to be used to automate configuration of the local disk on client machines, see chapter 6.

To build AFS from source, see chapter 7.

1.2. Requirements

The installation instructions assume that you have met the following requirements.

1.2.1. Login Identity

All procedures assume that you are logged on as "root" at the console of the machine you are installing or building a kernel on.

1.2.2. Recommended Reading

It is suggested that you read the entire chapter or section that describes an installation task before performing that task (for example, read all of chapter 2 before installing the first machine in your cell).

In addition, you will be more efficient if you are familiar with some general AFS concepts, because you will understand better why you are performing a given step. The following is a prioritized list of the sections you should read before installing an AFS system. At minimum, read Chapter 1 of the *AFS System Administrator's Guide*. Then continue your reading in the indicated order, as extensively as you can.

Selected Topics in the *AFS System Administrator's Guide*

- Overview (All of Chapter 1)
- Command Syntax (vi - xii)
- File Server Machine Processes and Files (Chapter 3)
- Controlling and Checking Process Status (Section 4.2)
- CellServDB on File Server Machines (Section 9.2)
- Encryption Keys (Section 10.2)
- Volumes (Sections 5.2 - 5.4)
- Mount Points (Section 5.7)
- Cache Manager (Sections 13.2 - 13.4)
- Access Control Lists (Section 19.2)

Selected Topics in the *AFS User's Guide*

- Concepts (All of Chapter 1)

More Selected Topics in the *AFS System Administrator's Guide*

- Backup System (Section 6.2)

- Backup Volumes (Section 5.6)
- ReadOnly Volumes (Section 5.5)
- Protection Groups (Section 18.2)
- Components of a User Account (Section 17.2)
- Uss (Section 16.2)
- Privileged Users/Accounts (Section 20.2)

1.2.3. General Requirements

- You must have a separate AFS Binary Distribution tape for each different system type you will install. Unless otherwise noted, the Binary Distribution includes software for both client and file server machines of a given system type.
- The machines in the AFS cell must be able to access each other through the network.
- A tape reading facility (drive) must be operational on at least one file server or client machine.
- The installer must have root access to the machines that will be in the AFS cell. **All installation procedures must be performed by "root"**.
- The installer should be familiar with the existing configuration of potential AFS machines. The System Administrator (or person responsible for cell configuration and administration) should assist in the installation.
- The hardware and software of all potential file server and client machines must be functioning normally.
- All potential AFS file server and client machines must be running a standard vendor-supplied version of UNIX®, or a version of AFS earlier than AFS 3.0. The root partition with the UNIX operating system must already be installed.
- Since the installation procedure requires you to reboot the machine frequently, you should ask other users to avoid using the machine during installation.

1.2.4. File Server Machine Requirements

- The first machine you install should, if possible, have the lowest IP address of any file server machine you currently plan to install, because you will make it your cell's first database server machine. As explained in more detail in the introduction to section 3.3, if you later install database server functionality on a machine with a lower Internet address, you must update the */usr/vice/etc/CellServDB* file on all of your cell's client machines before the installation.
- Each file server machine must have a */usr* partition with about 18 megabytes of disk space available for AFS file server machine binaries (stored in */usr/afs/bin*). The complete set of AFS binaries requires more space, as indicated in the chart in step 4 in section 2.27, but they are normally stored in an AFS volume rather than on a machine's local disk.

- Each file server machine that uses the **dkload** dynamic kernel loader must have one megabyte of free disk space in */tmp* for temporary files created as **dkload** runs.
- Each file server machine must have at least one partition dedicated to storing AFS volumes. AFS volumes contain user files plus client binaries for each system type. The total number and size of AFS file server partitions (distributed on file servers throughout the cell) determines how much space is available for AFS files.

1.2.5. Client Machine Requirements

- Each client machine that uses disk caching must have a */usr* partition with at least 5 megabytes of free disk space for the AFS cache, though caches of at least 10 megabytes perform better. Client machines using memory caching should be able to devote 5 megabytes of memory to the cache. The sections on setting up the cache in chapters 2 and 4 further discuss cache size.
- Each client machine that uses the **dkload** dynamic kernel loader must have one megabyte of free disk space in */tmp* for temporary files created as **dkload** runs.

1.3. Supported System Types

Transarc currently supports AFS on the following system types. This list is subject to change.

System name	CPU and operating system type
hp300_ux90	Hewlett-Packard 9000 Series 300 or 400 running HP-UX 9.0
hp700_ux90	Hewlett-Packard 9000 Series 700 running HP-UX 9.0
hp800_ux90	Hewlett-Packard 9000 Series 800 running HP-UX 9.0
next_mach30	NeXT (68030 or 68040 system) running NeXT OS Release 3.0
pmax_ul43	DEC DECstation 2100, 3100 or 5000 (single processor only) running Ultrix 4.3
pmax_ul43a	DEC DECstation 2100, 3100 or 5000 (single processor only) running Ultrix 4.3a
rs_aix32	IBM RS/6000 running AIX 3.2
rt_aix221	IBM-RT/PC running AIX 2.2.1
rt_aos4	IBM-RT/PC running AOS Release 4
sgi_ir50	Silicon Graphics IRIS Indigo running IRIX 5.0.1 (client only)
sgi_ir51	Silicon Graphics IRIS Indigo running IRIX 5.1
sun3_411	Sun 3 (68020 system) running SunOS 4.1.1, 4.1.2 or 4.1.3
sun3x_411	Sun 3 (68030 system) running SunOS 4.1.1, 4.1.2 or 4.1.3
sun4_411	Sun 4 (except SPARCstation) running SunOS 4.1.1, 4.1.2 or 4.1.3
sun4_52	Sun 4 (except SPARCstation) running Solaris 2.2
sun4c_411	Sun SPARCstation running SunOS 4.1.1, 4.1.2 or 4.1.3
sun4c_52	Sun SPARCstation running Solaris 2.2
sun4m_412	Sun 600 Series MP SPARCstation running SunOS 4.1.2 or 4.1.3
sun4m_52	Sun 600 Series MP SPARCstation running Solaris 2.2
vax_ul43	DEC VAX system running Ultrix 4.3

1.3.1. About Upgrading to a New Operating System

As the chart in section 1.3 indicates, sites upgrading to AFS 3.3 from previous AFS versions must also upgrade the operating system on several system types (for example, DEC machines to Ultrix 4.3, HP machines to HP-UX 9.0, and NeXT machines to NeXT OS 3.0). Sites may also choose to update Sun 4s and SPARCstations to Solaris 2.2.

Whenever you upgrade an AFS machine to a new operating system version, you must take several actions to maintain proper AFS functionality. These actions include (but may not be limited to):

- unmounting the */vicep**x*** partitions on all file server machines, to prevent the vendor-supplied **fsck** program from running on them when you reboot the machine during installation of the new operating system. Before upgrading the operating system, you may need to comment out commands in the machine's initialization file that remount the */vicep**x*** partitions, to make sure that the partitions are not remounted until you can replace standard **fsck** with the AFS version. Section 2.6 explains how to replace **fsck** on each system type.
- protecting the AFS-modified versions of several commands from being overwritten by vendor-supplied versions, including **login**, **vfsck** (the AFS version of **fsck**) and the "remote" commands described in the "Modified UNIX Commands" chapter of the *AFS Command Reference Manual* (**rsh**, **inetd**, etc.). After you have successfully installed the operating system, remember to move the AFS-modified commands back to the locations where they will be accessed during normal functioning.

1.4. The AFS Binary Distribution

The AFS Binary Distribution is available on several types of magnetic tape. The tape divides the files you will need into six **tar** sets. Section 1.4.2 provides preliminary instructions for unloading files onto the local disk before beginning the installation proper. **Follow the instructions appropriate for your machine configuration before continuing on to other chapters.**

If you are installing AFS onto machines of different system types, you need a different tape for each system type.

If you have an AFS Source Distribution tape—implying you have a source license—you may build AFS from source for those system types for which you have also purchased binaries. For instructions, see chapter 7. Sites with source licenses may also receive a Binary Distribution tape for each purchased system type.

1.4.1. The Contents of the Binary Distribution Tape

The AFS Binary Distribution tape contains the following **tar** sets in the order listed:

1. the first **tar** set contains the files necessary for running the dynamic kernel loader for this system type. The files from this set must reside on the local disk of any AFS machine where you wish to run a dynamic kernel loader. The name of the set differs for various system types:

    - *sgiload* on SGI systems
    - *modload* on Solaris and SunOS machines (for SunOS, there is also a *dkload* directory)
    - *dkload* on any other system type that uses a dynamic kernel loader

    If a dynamic loader is not available for the system type, this set is empty.
2. */usr/afs/sys* contains files necessary to build a kernel that incorporates AFS modifications. If kernel building is not possible for the system type, this set is empty.
3. */usr/afs* contains the file server process binaries. These files must reside on the local disk of AFS file server machines.
4. */usr/vice/etc* contains the Cache Manager initialization program (**afsd**), Cache Manager configuration files, and files necessary for running a dynamic kernel loader. These files must reside on the local disk of AFS client machines.
5. */usr/afsws* includes second copies of all files listed above in the other **tar** sets (*dkload*, */usr/afs/sys*, */usr/afs*, and */usr/vice/etc*), in addition to user command binaries, system administrator command binaries, AFS include and library files for programming, and online man pages for AFS command suites. This set is divided into the following subdirectories:

    - *bin* - AFS user command binaries for use on client machines

- *etc* - AFS system administrator command binaries for use on client machines
- *include* - AFS include files for programming on top of AFS
- *lib* - AFS library files for programming on top of AFS
- *root.client* - files necessary for building kernels that incorporate AFS modifications, for running a kernel dynamic loader, and for running the Cache Manager on client machines
- *root.server* - files necessary for building kernels that incorporate AFS modifications, and for running a file server machine

6. *wsadmin* contains configuration files used by the **package** program.

1.4.2. Unloading Files from the Binary Distribution Tape

There are two ways to extract files from the AFS Binary Distribution tape available for all supported systems:

- onto the local machine (where you are performing the installation), if it has a tape drive attached. See section 1.4.2.1 for preliminary instructions.
- onto a remote machine with a tape drive attached. You then copy the needed files to the local machine's disk using **ftp**, NFS®, or another network transfer method. See section 1.4.2.2 for preliminary instructions.

1.4.2.1. Loading Files Using a Local Tape Drive

If the machine that you are installing has a tape drive attached, you can unload files directly from the Binary Distribution Tape onto the local disk.

You do not need to take any action at this point. When later chapters in this guide indicate a need to load files onto the local disk, follow the instructions for using a local tape drive; they will tell you which **tar** set to unload.

1.4.2.2. Loading Files Using the Tape Drive on a Remote Machine

If the machine onto which you are installing AFS does not have a tape drive, you must load the files from the Binary Distribution tape's fifth **tar** set into the */usr/afsws* directory on a remote machine that has a tape drive. Later instructions in this guide will have you copy files from the remote machine's */usr/afsws* directory onto the local disk of the machine you are installing, using **ftp**, NFS or another file transfer method.

Perform the following steps now, before continuing on to the remainder of this guide. When later chapters indicate a need to load files to the local disk, follow the instructions for unloading from a remote machine.

Step 1: **Working on the remote machine,** create the */usr/afsws* directory.

```
# mkdir /usr

# mkdir /usr/afsws
```

Step 2: **Working on the remote machine,** mount the Binary Distribution tape and load the fifth **tar** set into */usr/afsws*.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/afsws

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 4

# tar xvf /dev/<device>
```

1.5. How to Continue

If you are installing the first AFS machine in your cell, proceed now to chapter 2. The instructions there make the first machine both a file server and a client.

If you are installing an additional file server machine or installing/removing database server machine functionality, proceed to chapter 3.

If you are are installing an additional client machine, proceed to chapter 4.

2. Installing the First AFS Machine

This chapter describes how to install the first AFS machine in your cell, setting it up as both an AFS file server machine and a client machine. After completing all procedures in this chapter, you can remove the client functionality, if desired.

To install additional file server machines, follow the instructions in chapter 3, "Installing Additional File Server Machines," after completing this chapter.

To install additional client machines, follow the instructions in chapter 4, "Installing Additional Client Machines," after completing this chapter.

Assumptions

The instructions in this chapter require that:

- you type the instructions at the console of the machine you are installing
- you are logged in as "root"
- you have already installed a standard UNIX kernel on the machine being installed

Configuration Decisions

During the installation, you must make several decisions about how you to set up your cell. They include:

- choosing the first AFS machine
- choosing a cell name
- determining the cache size on client machines
- defining partition names
- setting up your cell's AFS tree structure

Chapter 2 of the *AFS System Administrator's Guide*, "Issues in Cell Set-up and Administration," provides guidelines. It is recommended that you read it before beginning the installation.

How to Use This Chapter

This chapter is divided into three large sections corresponding to the three different functions you must perform to install the first AFS machine in your cell. For the installation to be complete and correct, you must perform all steps in the order they appear. A summary of the procedures in each functional section appears at the beginning of the section. The functions are:

- Installing file server machine functionality (beginning in section 2.1)
- Installing client machine functionality (beginning in section 2.18)
- Setting up your cell's file tree, establishing further security mechanisms and enabling access to foreign cells, among other tasks (beginning in section 2.24)

2.1. Overview: Installing File Server Functionality

The first phase of the installation is to install file server machine functionality, by performing the following procedures:

1. Choose which machine to install as the first AFS machine.
2. Incorporate AFS modifications into the machine's kernel (using either a dynamic kernel loader or the kernel build procedure).
3. Load the file server binaries to the local disk.
4. Set up AFS partitions suitable for storing volumes.
5. Replace the standard **fsck** with the AFS version of **fsck**.
6. Start the Basic OverSeer (BOS) Server.
7. Define the cell name and the machine's cell membership.
8. Start the Authentication Server.
9. Set up initial security mechanisms, including Authentication Database entries.
10. Start the Protection Server.
11. Start the Volume Location Server.
12. Start the Backup Server.
13. Start the *fs* process, which incorporates three component processes: the File Server, Volume Server and Salvager.
14. Start the server portion of the Update Server.
15. Start the controller process for the Network Time Protocol Daemon which synchronizes clocks (called *runntp*).

To start and configure server processes, you will issue commands from several of the AFS ***command suites***. These reside in the */usr/afs/bin* directory on the file server machine. Specifically, you will use commands

- from the **bos** suite to contact the BOS Server.
- from the **kas** suite to contact the Authentication Server.
- from the **pts** suite to contact the Protection Server.
- from the **vos** suite to contact the Volume and Volume Location Servers.

The instructions in this chapter do not explain the function of each command or argument; for complete information, consult the *AFS Command Reference Manual*.

2.2. Choosing the First AFS Machine

The first AFS machine you install will function as an AFS file server, so it should have sufficient disk space to store AFS volumes. It is recommended that you store not only user files in volumes on a file server, but also AFS client binaries. When you later install additional file server machines in your cell, you can distribute these volumes among the different machines as you see fit.

If you follow these installation procedures, the first AFS file server machine in the cell will be designated as a ***database server machine***, the ***binary distribution machine*** for its CPU/operating system type, and (if you are using the United States edition of AFS) your cell's ***system control machine***. For more information on the roles of these machine types, refer to the *AFS System Administrator's Guide*.

Since it will be your cell's first database server machine, the first machine you install should, if possible, have the lowest IP address of any file server machine you currently plan to install. As explained in more detail in the introduction to section 3.3, if you later install database server functionality on a machine with a lower Internet address, you must first update the */usr/vice/etc/CellServDB* file on all of your cell's client machines.

The instructions in this chapter make the machine into both an AFS file server machine and an AFS client machine. Setting up the machine as a client makes it easier to create and view AFS volumes and to set up your cell's file tree under */afs/**cellname***. Section 2.34 explains how to remove the client functionality after you have completed the installation, if you wish to do so.

2.3. Incorporating AFS Modifications into the Kernel

All AFS machines must run a kernel that includes AFS modifications. There are two possible ways to add these modifications into the kernel: dynamic loading or kernel building. The following table summarizes which method is available for each system type.

If dynamic loading is possible for your system type, it is the recommended method, as it is significantly quicker and easier than kernel building. A dynamic loader adds the AFS modifications to the memory version of the kernel created at each reboot, without altering the disk version (*/vmunix* or equivalent). First follow the instructions in section 2.3.1 to load the necessary files to the local disk. Then proceed to the section listed in the table to learn how to run the loader.

If you must (or choose to) build a kernel, see chapter 5 for complete instructions for each system type. When you are finished building the kernel, return to section 2.3.8.

Operating system	Dynamic loading	Building
AIX 2.2.1	Not possible	See section 5.3
AIX 3.2	Use AIX kernel extension facility, provided by IBM (see section 2.3.3)	Not possible
AOS	Use **dkload** (see section 2.3.2)	See section 5.4
HP-UX for HP 9000 Series 300, 400 and 700 machines	Use **dkload** (see section 2.3.2)	See section 5.5
HP-UX for HP 9000 Series 800 machines	Use **dkload** (see section 2.3.2)	See section 5.6
IRIX	Use **ml**, provided by Silicon Graphics (see section 2.3.4)	See section 5.7
NeXT OS	Use **kl_util**, provided by NeXT (see section 2.3.5)	Not possible
Solaris	Use **modload**, provided by Sun (see section 2.3.6)	Not possible
SunOS	Use **dkload** (see section 2.3.2) or **modload**, provided by Sun (see section 2.3.7)	See section 5.8
Ultrix	Use **dkload** (see section 2.3.2)	Possible only with Ultrix source license (see section 5.9 for DECstations, section 5.10 for VAX systems)

2.3.1. Loading Dynamic Kernel Loader Files to the Local Disk

Before you can run a dynamic kernel loader, certain files must reside on the local disk, in

- */usr/afs/bin/modload* for Solaris, and for SunOS systems using **modload**
- */usr/afs/bin/sgiload* for IRIX systems
- */usr/afs/bin/dkload* for all other loaders

Follow the instructions for the machine configuration you are using, as discussed in section 1.4.2.

2.3.1.1. Loading Dynamic Kernel Loader Files Using a Local Tape Drive

If the machine you are installing has a tape drive, you can load the first **tar** set from the Binary Distribution tape directly onto the local disk.

Step 1: Create the */usr/afs* and */usr/afs/bin* directories.

```
# mkdir /usr

# mkdir /usr/afs

# mkdir /usr/afs/bin
```

Step 2: Mount the tape and extract the first **tar** set into */usr/afs/bin*. On Solaris and SunOS systems, the *modload* directory is created automatically; on IRIX systems, the *sgiload* directory is created automatically; on all systems except Solaris and IRIX, the *dkload* directory is created automatically.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/afs/bin

# mt -f /dev/<device> rewind

# tar xvf /dev/<device>
```

2.3.1.2. Loading Dynamic Kernel Loader Files from a Remote Machine

If the machine you are installing (the "local" machine) does not have a tape drive, you must transfer the needed files onto the local disk from an existing subdirectory of */usr/afsws*:

- */usr/afsws/root.client/usr/vice/etc/modload* contains the files necessary for running **modload** on Solaris and SunOS systems.
- */usr/afsws/root.client/usr/vice/etc/sgiload* contains the files necessary for running **ml** on IRIX systems.
- */usr/afsws/root.client/usr/vice/etc/dkload* contains the files necessary for running other dynamic kernel loaders.

Step 1: Working on the local machine, create the directory for housing dynamic kernel loader files on the local disk (*/usr/afs/bin/dkload*, */usr/afs/bin/modload*, or */usr/afs/bin/sgiload* as indicated).

```
# mkdir /usr

# mkdir /usr/afs

# mkdir /usr/afs/bin

# mkdir /usr/afs/bin/modload on Solaris, and SunOS systems using modload

# mkdir /usr/afs/bin/sgiload on IRIX systems

# mkdir /usr/afs/bin/dkload on other system types
```

Step 2: Using **ftp**, NFS, or another network file transfer method, copy the files from the remote machine's */usr/afsws/root.client/usr/vice/etc/dkload*, *modload*, or *sgiload* directory to the directory on the local machine created in the previous step.

2.3.2. Using Transarc's Dynamic Kernel Loader, dkload

The **dkload** program is the dynamic kernel loader provided by Transarc. It is available for the following operating systems:

- AOS
- HP-UX
- SunOS
- Ultrix

For operating systems not listed here, consult the table at the beginning of section 2.3 to learn what loading and building options are available.

For this machine to remain an AFS machine, **dkload** must run each time the machine reboots. You can invoke **dkload** automatically in the machine's initialization file (*/etc/rc* or equivalent), as explained in step 3 below.

The files containing the AFS kernel modifications are delivered in standard and "stripped" versions. The standard versions are called *libafs.a* and *libafs.nonfs.a* (the latter is for machines that do not support NFS). The corresponding stripped versions are called *libafss.a* and *libafss.nonfs.a*; because they lack the symbol table found in the standard versions, they require somewhat less disk and core memory, but do not permit debugging. If you wish to use the stripped versions, substitute them for the standard versions in instructions below.

The following instructions are brief. More detailed instructions appear in section 5.13.

To invoke **dkload** manually:

Step 1: Verify that

- there is at least one spare megabyte of space in */tmp* for temporary files created as **dkload** runs
- the following are in */bin* on the local disk (not as symbolic links): **as**, **ld**, and **nm**.
- the */usr/afs/bin/dkload* directory on the local disk contains: *dkload* (the binary), *libafs.a*, *libafs.nonfs.a*, *libcommon.a*, and *kalloc.o*
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time it should not have, but reboot without invoking **afsd** if necessary.

Step 2: Invoke **dkload**. On SunOS and Ultrix systems only, you must run **ranlib** first.

> **Note:** If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.a* for *libafs.a*. Either use the **mv** command to replace *libafs.a* with *libafs.nonfs.a* in the */usr/afs/bin/dkload* directory (before issuing these commands), or make the substitution on the command line. If you wish to use the "stripped" version of the AFS library file, substitute '*libafss*' for '*libafs*' in the same fashion.
>
> # **cd /usr/afs/bin/dkload**
>
> # **ranlib libafs.a** on SunOS and Ultrix systems only
>
> # **ranlib libcommon.a** on SunOS and Ultrix systems only
>
> # **./dkload libafs.a**

Step 3: If you wish **dkload** to run each time the machine reboots, verify that it is invoked in the machine's initialization file (*/etc/rc* or equivalent). An easy way to add the needed commands is to copy the contents of */usr/afs/bin/dkload/rc.dkload.server*; place them in the file after the instructions that mount the file systems. The contents of this file appear in full in section 5.13.3. If the machine's kernel does not support NFS, remember to substitute *libafs.nonfs.a* for *libafs.a*. If you wish to use the "stripped" version of the AFS library file, substitute *libafss.a* for *libafs.a* or *libafss.nonfs.a* for *libafs.nonfs.a*.

2.3.3. Using the Kernel Extension Facility on AIX 3.2 Systems

The AIX kernel extension facility is a dynamic kernel loader provided by IBM Corporation for AIX 3.2. The **dkload** program is not available for this system type, nor is it possible to add AFS during a kernel build.

For this machine to remain an AFS machine, the kernel extension facility must run each time it reboots. You can invoke the facility automatically in the machine's initialization files, as explained in step 3 below.

The following instructions are brief. More detailed instructions appear in section 5.14.

To invoke the kernel extension facility manually:

Step 1: Verify that

- the */usr/afs/bin/dkload* directory on the local disk contains: *afs.ext*, *cfgafs*, *cfgexport*, *export.ext*, and *export.ext.nonfs*.
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time it should not have, but reboot without invoking **afsd** if necessary.
- NFS is already in the kernel (*nfs.ext* is already loaded), if you wish NFS to run on this machine. NFS must run for the machine to function as an NFS/AFS Translator machine.

Step 2: Invoke **cfgexport** and **cfgafs**.

Notes: If this machine will act as an NFS/AFS Translator machine, you must make a substitution in this step. For details, consult the section entitled "Setting Up an NFS/AFS Translator Machine" in the NFS/AFS Translator Supplement to the *AFS System Administrator's Guide*.

If NFS will not run on this machine, then you must substitute *export.ext.nonfs* for *export.ext* in this step. The machine cannot function as an NFS/AFS Translator in this case.

If the machine is to act as an NFS/AFS Translator, remember to make the necessary substitution. If the machine will not run NFS, substitute *export.ext.nonfs* for *export.ext*.

```
# cd /usr/afs/bin/dkload

# ./cfgexport -a export.ext

# ./cfgafs -a afs.ext
```

Step 3: IBM delivers several function-specific initialization files for AIX 3.2 systems, rather than the single file used on some other systems. If you want the kernel extension facility to run each time the machine reboots, verify that it is invoked in the appropriate place in these initialization files. An easy way to add the needed commands is to copy the contents of */usr/afs/bin/dkload/rc.dkload.server*, which appear in section 5.14.3.

The following list summarizes the order in which the indicated commands must appear in the initialization files for the machine to function properly (you will add some of the commands in later sections). Note particularly that you should not invoke **nfsd** at the same place in the initialization files as the commands that load *nfs.ext*; **nfsd** must follow the call to **afsd**.

- command(s) mounting the file systems
- command(s) loading *nfs.ext*, if the machine will run NFS, which an NFS/AFS Translator machine must (but only for AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)
- the contents of *rc.dkload.server*, to invoke the kernel extension facility. If the machine will act as an NFS/AFS Translator machine, be sure to make the same substitution as you made in step 2.
- **bosserver** (you will be instructed to add this command in section 2.17)
- **afsd** (you will be instructed to add this command in section 2.23)
- **nfsd** (if the machine is to be an NFS server)

2.3.4. Using ml on IRIX Systems

The **ml** program is the dynamic kernel loader provided by Silicon Graphics, Inc. for IRIX systems. The **dkload** program is not available for these system types. It is possible to build AFS into the kernel for these system types; see section 2.3.8.

For this machine to remain an AFS machine, **ml** must run each time the machine reboots. Step 4 below explains how to ensure this.

The following instructions are brief. More detailed instructions appear in section 5.15.

To invoke **ml** manually:

Step 1: Verify that

- the */usr/afs/bin/sgiload* directory on the local disk contains: *afs*, *afs.rc*, *afs.sm*, *libafs.R3000.o.dl*, *libafs.R3000.nonfs.o.dl*, *libafs.R4000.o.dl*, *libafs.R4000.nonfs.o.dl*.
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time it should not have, but reboot without invoking **afsd** if necessary.

Step 2: Remove the *.dl* extension from the AFS library file that is appropriate for this machine, selecting R3000 versus R4000 processor, and no NFS support versus NFS support.

If you do not know which processor your machine has, issue IRIX's **hinv** command and check the line in the output that begins "CPU."

Remove the *.dl* extension from one of the AFS library files by issuing the (one) appropriate command.

cd /usr/afs/bin/sgiload

For R3000 processor with NFS support:

cp libafs.R3000.o.dl libafs.R3000.o

For R3000 processor without NFS support:

cp libafs.R3000.nonfs.o.dl libafs.R3000.nonfs.o

For R4000 processor with NFS support:

cp libafs.R4000.o.dl libafs.R4000.o

For R4000 processor without NFS support:

cp libafs.R4000.nonfs.o.dl libafs.R4000.nonfs.o

Step 3: Issue the **ml** command.

Issue the following command, replacing *library file* with the name of the library file whose *.dl* extension who removed in step 2.

ml ld -v -j /usr/afs/bin/sgiload/<*library file*> **-p afs_ -a afs**

Step 4: Transarc provides an initialization script for IRIX systems as */usr/afs/bin/sgiload/afs.rc* that includes an **ml** instruction for loading AFS as the machine reboots. This script also includes automatic tests for whether the machine has the R3000 or R4000 processor and NFS support or no NFS support.

To ensure that *afs.rc* is invoked properly at each reboot, you must copy it from */usr/afs/bin/sgiload* to the initialization files directory (standardly, */etc/init.d*), make sure it is executable, and link it to the two locations where **ml** expects to find it. Then use the **chkconfig** command to activate configuration files that tell **ml** that this machine will run AFS and serve as a file server machine.

```
# cd /etc/init.d

# cp /usr/afs/bin/afs.rc afs.rc

# chmod 555 afs.rc

# ln -s afs.rc /etc/rc0.d/K35afs.rc

# ln -s afs.rc /etc/rc2.d/S35afs.rc

# cd /etc/config

# /etc/chkconfig -f afsml on

# /etc/chkconfig -f afsserver on
```

2.3.5. Using kl_util on NeXT OS Systems

The **kl_util** program is the dynamic kernel loader provided by NeXT, Inc. for NeXT OS. The **dkload** program is not available for this system type, nor is it possible to add AFS during a kernel build.

For the machine to remain an AFS machine, the AFS modifications must be added to the kernel each time the machine reboots. This happens automatically if you add the appropriate line to */etc/kern_loader.conf*, as instructed in step 4 below.

The following instructions for running **kl_util** manually are brief. More detailed instructions appear in section 5.16.

To invoke **kl_util** manually:

Step 1: Verify that

- *afs_reloc* exists in */usr/lib/kern_loader/AFS* on the local disk
- the **kl_util** binary is available on the local disk (standard location is */usr/etc*)
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time it should not have, but reboot without invoking **afsd** if necessary.

Step 2: Copy the *afs_reloc* file provided under */usr/afs/bin/dkload* to the location where **kl_util** expects to find it, */usr/lib/kern_loader/AFS/afs_reloc*).

Create the */usr/lib/kern_loader/AFS* directory and copy the *afs_reloc* file into it.

mkdir /usr/lib/kern_loader/AFS

cp /usr/afs/bin/dkload/afs_reloc /usr/lib/kern_loader/AFS

Step 3: Invoke **kl_util**.

/usr/etc/kl_util -a /usr/lib/kern_loader/AFS/afs_reloc

Step 4: If you wish **kl_util** to run each time the machine reboots, verify that the following line appears in */etc/kern_loader.conf* (any ordering with other lines is acceptable):

```
/usr/lib/kern_loader/AFS/afs_reloc
```

2.3.6. Using modload on Solaris Systems

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for Solaris systems. The **dkload** program is not available for these system types, nor it it possible to build AFS into the kernel.

For this machine to remain an AFS machine, **modload** must run each time the machine reboots. Step 5 below explains how to ensure this.

The following instructions are brief. More detailed instructions appear in section 5.17.

To invoke **modload** manually:

Step 1: Verify that

- the **modload** binary is available on the local disk (standard location is */usr/sbin*)
- the */usr/afs/bin/modload* directory on the local disk contains *libafs.o* and *libafs.nonfs.o*
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time, it should not have, but reboot without invoking **afsd** if necessary.

Step 2: Create the file */kernel/fs/afs* as a copy of the appropriate AFS library file. If the machine is to function as an NFS/AFS translator machine, you must copy *libafs.o* (as well as run NFS).

```
# cd /usr/afs/bin/modload

# cp libafs.o  /kernel/fs/afs if the machine supports NFS

# cp libafs.nonfs.o  /kernel/fs/afs if the machine does not support NFS
```

Step 3: Create an entry for AFS in the */etc/name_to_sysnum* file, to allow the kernel to make AFS system calls.

In the file */etc/name_to_sysnum*, create an "afs" entry in slot 105 (the slot just before the "nfs" entry), so that the file looks like:

```
reexit          1
fork            2
 .              .
 .              .
 .              .
sigpending      99
setcontext      100
statvfs         103
fstatvfs        104
afs             105
nfs             106
```

Step 4: Invoke **modload**.

```
# /usr/sbin/modload  /kernel/fs/afs
```

If you wish to verify that AFS loaded correctly, use the **modinfo** command.

```
# /usr/sbin/modinfo | egrep afs
```

The following output indicates that AFS loaded successfully:

```
69 fc71f000 4bc15 105   1  afs (afs syscall interface)
69 fc71f000 4bc15  15   1  afs (afs file system)
```

Step 5: Transarc provides an initialization script for Solaris systems as */usr/afs/bin/modload/afs.rc* that includes a **modload** instruction for loading AFS as the machine reboots.

To ensure that *afs.rc* is invoked properly at each reboot, you must copy it from */usr/afs/bin/modload* to the initialization files directory (standardly, */etc/init.d*), make sure it is executable, and link it to the two locations where **modload** expects to find it.

```
# cd  /etc/init.d

# cp  /usr/afs/bin/modload/afs.rc  afs.rc

# chmod  555  afs.rc

# ln -s afs.rc  /etc/rc3.d/S14afs

# ln -s afs.rc  /etc/rc2.d/K66afs
```

2.3.7. Using modload on SunOS Systems

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for SunOS systems. The **dkload** program is also available for these system types, if you prefer to follow the instructions in section 2.3.2. It is also possible to build AFS into the kernel for these system types; see section 2.3.8.

For this machine to remain an AFS machine, **modload** must run each time the machine reboots. You can invoke **modload** automatically in the machine's initialization file (*/etc/rc* or equivalent), as explained in step 3 below.

The following instructions are brief. More detailed instructions appear in section 5.18.

To invoke **modload** manually:

Step 1: Verify that

- the */usr/afs/bin/modload* directory on the local disk contains *libafs.o*
- the **modload** binary is available on the local disk (standard location is */usr/etc*)
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time, it should not have, but reboot without invoking **afsd** if necessary.

Step 2: Invoke **modload**.

> **Note:** If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.o* for *libafs.o*. Either use the **mv** command to replace *libafs.o* with *libafs.nonfs.o* in the */usr/afs/bin/modload* directory (before issuing these commands), or make the substitution on the command line.
>
> # **cd /usr/afs/bin/modload**
>
> # **/usr/etc/modload ./libafs.o**

Step 3: If you wish **modload** to run each time the machine reboots, verify that it is invoked in the machine's initialization file (*/etc/rc* or equivalent). An easy way to add the needed commands is to copy the contents of */usr/afs/bin/modload/rc.modload.server*; place them in the file after the instructions that mount the file systems. The contents of this file appear in full in section 5.18.3. If the machine's kernel does not support NFS, remember to substitute *libafs.nonfs.o* for *libafs.o*.

2.3.8. Installing a Previously-Built AFS-Modified Kernel

This section describes how to install a new kernel on the first AFS machine. You may skip it if you dynamically loaded the kernel already.

Step 1: Follow the instructions in chapter 5 to build AFS modifications into a kernel for this machine's system type.

Step 2: Move the existing kernel on the local machine to a safe location.

```
# mv  /vmunix  /vmunix_save
```

Step 3: Use a copying program (either **cp** or a remote program such as **ftp** or NFS) to copy the AFS-modified kernel to */vmunix*. A standard location for the AFS-modified kernel is */usr/sys/**kernel_name**/vmunix*.

Step 4: Reboot the machine to start using the new kernel. This example instruction shows the rebooting command appropriate for most system types.

```
# sync

# /etc/reboot
```

2.4. Loading the File Server Process Binaries

Now that you have incorporated AFS modifications into the kernel, the next step is to load the AFS server process binaries and configuration files from the Binary Distribution onto the machine's local disk. Follow the instructions suitable for your machine configuration, as discussed in section 1.4.2).

2.4.1. Loading Files From a Local Tape Drive

The third **tar** set on the Binary Distribution tape contains the AFS server process binaries and configuration files. The server process binaries are in the **tar** set's *bin* directory and the configuration files in the *etc* directory. The following procedure loads these directories into the local disk directories */usr/afs/bin* and */usr/afs/etc* respectively.

Step 1: Create the */usr/afs* directory, if necessary. If you used a dynamic kernel loader in section 2.3, the directory should already exist.

```
# mkdir /usr/afs
```

Step 2: Mount the Binary Distribution tape and load the third **tar** set into */usr/afs*. The *bin* and *etc* directories will be loaded automatically.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/afs

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 2

# tar xvf /dev/<device>
```

2.4.2. Loading Files From a Remote Machine

This section assumes that the file server binaries are stored in */usr/afsws/root.server/usr/afs* on the remote machine.

Step 1: **Working on the local machine,** create the */usr/afs* directory if necessary. If you used a dynamic kernel loader in section 2.3, the directory should already exist.

```
# mkdir /usr/afs
```

Step 2: Using a file transfer program such as **ftp** or NFS, copy the files from */usr/afsws/root.server/usr/afs* on the remote machine to */usr/afs* on the local machine.

2.5. Setting Up AFS Partitions

AFS volumes must reside on partitions associated with directories named */vicep**x***, where ***x*** is a lowercase letter of the alphabet. By convention, the first directory created on a machine to house AFS volumes is called */vicepa*, the second directory */vicepb*, etc. For most system types, you must also

- alter the "file systems registry" file (*/etc/fstab* or equivalent) to mount the partitions at boot up
- create a file system on each partition
- mount the partitions manually

Every file server machine must have at least one partition devoted exclusively to storing AFS volumes (preferably associated with the directory */vicepa*). You cannot simply create a directory under an existing directory (for example, */usr/vicepa* is not legal).

You can also perform the steps in this section when you want to set up a new */vicep**x*** partition on an existing file server machine. In that case, you must also restart the BOS Server (which restarts the *fs* process) to force recognition of the new partition. Complete instructions also appear in chapter 11 of the *AFS System Administrator's Guide*.

Note for HP-UX systems: Disk striping is supported for the *hp700_ux90* system type, but not for *hp300_ux90* (so only one partition per disk is allowed on that system type). The *hp800_ux90* system type uses logical volumes rather than disk striping.

Step 1: Decide how many partitions to devote to storage of AFS volumes. There must be at least one.

Step 2: Create a directory called */vicep**x*** for each partition chosen in step 1. The example instruction creates three directories.

Create a directory for each partition to be used for storing AFS volumes.

```
# mkdir /vicepa

# mkdir /vicepb

# mkdir /vicepc

   and so on
```

Step 3: For each directory created in step 2, take the action necessary to associate the directory with one of the partitions chosen in step 1. For many system types, this requires adding a line to the machine's "file systems registry" file (*/etc/fstab* or equivalent).

Add a line to */etc/fstab* or equivalent for each directory you created in step 2. Use the format appropriate for your system type, substituting the correct directory name for */vicep***x**, and the appropriate SCSI disk name for *disk*. Some system types use spaces rather than colons as field separators; consult the existing file for clarification.

Examples appear following the box.

For AIX 2.2.1, use the **devices** program to edit */etc/filesystems*. Consult the operating system documentation for syntax.

For AIX 3.2, use the **SMIT** program to create a journalled file system. Consult the operating system documentation for syntax.

For AOS: `/dev/`<*disk*> `/vicep`<*x*> `ufs rw 1 2`

For HP-UX on Series 300, 400 and 700 machines, place the following in */etc/checklist*:
`/dev/dsk/`<*disk*> `/vicep`<*x*> `hfs defaults 0 2`

For HP-UX on Series 800 machines using logical volumes, use the **SAM** program to create a file system. Consult the operating system documentation for syntax. (If not using logical volumes, use the same instruction as for Series 300, 400 and 700 machines.)

For IRIX: `/dev/vicep`<*x*> `/vicep`<*x*> `efs rw,raw=/dev/rvicep`<*x*> `0 0`

For NeXT OS: `/dev/`<*disk*> `/vicep`<*x*> `4.3 rw,noquota 0 2`

For Solaris, place the following in */etc/vfstab*:
`/dev/dsk/`<*disk*> `/dev/rdsk/`<*disk*> `/vicep`<*x*> `ufs` <*boot order*> `yes`

For SunOS: `/dev/`<*disk*> `/vicep`<*x*> `4.2 rw 1 2`

For Ultrix: `/dev/`<*disk*>`:/vicep`<*x*>`:rw:1:2:ufs::`

Following are examples of file registry entries (in */etc/fstab* except for HP-UX, which uses */etc/checklist*, and Solaris, which uses */etc/vfstab*).

```
AOS:      /dev/hd1g /vicepa ufs rw 1 2
HP-UX:    /dev/dsk/1s0  /vicepa hfs  defaults  0 2
IRIX:     /dev/vicepa /vicepa efs rw,raw=/dev/rvicepa 0 0
NeXT OS:  /dev/sd0b /vicepa 4.3 rw,noquota 0 2
Solaris:  /dev/dsk/c0t6d0s1 /dev/rdsk/c0t6d0s1 /vicepa ufs 3 yes
SunOS:    /dev/sd0g /vicepa 4.2 rw 1 2
Ultrix:   /dev/rz4a:/vicepa:rw:1:2:ufs::
```

Step 4: Create a file system on each partition associated with a directory in step 3. The commands shown should be suitable for the indicated system types, but consult your operating system documentation for more information.

Create a file system on each partition, using the command appropriate for your system type. **Repeat this command for each partition**.

For AIX 2.2.1: # **minidisks**

For AIX 3.2, your creation of a journalled file system (with **SMIT** in step 3) makes this step unnecessary.

For AOS: # **minidisk**

For HP-UX on Series 300, 400 and 700 machines, use the **newfs** or **makefs** command; consult the operating system documentation for syntax.

For HP-UX on Series 800 machines, your use of **SAM** to create a file system in step 3) makes this step unnecessary. If you did not use **SAM**, use **newfs** or **makefs** as directed for Series 300, 400 and 700 machines.

For IRIX: # **mkfs /dev/rvicep**<*x*>

For NeXT OS, use the **disk** command or **BuildDisk** program; consult the operating system documentation for syntax.

For Solaris: # **newfs -v /dev/rdsk/**<*xxxxxxxx*>

For SunOS: # **newfs -v /dev/rsd**<*xx*>

For Ultrix: # **newfs -v /dev/rhd**<*xx*> <*disk type*>

Step 5: Mount the partition(s), using the command appropriate to your operating system. For many BSD-based systems, the **mount -a** command is appropriate and mounts all partitions at once. For many System V-based systems, the **mountall** command is appropriate instead.

For NeXT OS systems, no "mount all" command is appropriate. You must either reboot the machine or mount each */vicep**x*** partition by hand, because the **mount** command consults NetInfo rather than */etc/fstab* once the machine initializes. If you reboot, remember to log in again as "root."

For AIX 3.2 systems, it is recommended that you also add the following line to */etc/vfs* at this point:

```
afs     4     none     none
```

If you do not add this line, you will receive an error message from the **mount** command any time you use it to list the mounted file systems (but note that the **mount** command is working properly even if you receive the error message).

2.6. Replacing fsck

You should never run the standard **fsck** program on an AFS file server machine. It will discard the files that make up AFS volumes on the partitions associated with the */vicepx* directories, because they are not standard UNIX directories. In this step, you replace standard **fsck** with a modified **fsck** provided by Transarc. It properly checks both AFS and standard UNIX partitions. To repeat:

NEVER run the standard vendor-supplied fsck program on an AFS file server machine. It discards AFS volumes.

You can tell you are running the correct, AFS version when it displays the banner:

```
[Transarc AFS (R) 3.3 fsck]
```

Follow the instructions below that are appropriate for your system type:

- For AIX 3.2 systems, see section 2.6.1.
- For DECstations, see section 2.6.2.
- For IRIX systems, see section 2.6.3.
- For NeXT OS systems, see section 2.6.4.
- For Solaris systems, see section 2.6.5.
- For SunOS systems, see section 2.6.6.
- For all other system types, see section 2.6.7.

2.6.1. Replacing fsck on AIX 3.2 Systems

Follow the instructions in this section to replace **fsck** on AIX 3.2 systems.

For this system type, you do not replace **fsck** itself, but rather the "program helper" distributed as */sbin/helpers/v3fshelper*.

Step 1: Move the standard **fsck** program helper to a save file and install the AFS-modified helper (*/usr/afs/bin/v3fshelper*) in the standard location.

```
# cd  /sbin/helpers

# mv  v3fshelper  v3fshelper.noafs

# cp  /usr/afs/bin/v3fshelper  v3fshelper
```

2.6.2. Replacing fsck on DECstations

Follow the instructions in this section to replace **fsck** on DECstations running Ultrix.

Step 1: Move standard **fsck** to a save file, install the AFS-modified **fsck** ("**vfsck**") to the standard location and link standard **fsck** to it.

```
# mv  /bin/fsck  /bin/fsck.old

# cp  /usr/afs/bin/vfsck  /bin/vfsck

# rm  /etc/fsck

# ln  -s  /bin/vfsck  /etc/fsck
```

2.6.3. Replacing fsck on IRIX Systems

Silicon Graphics, Inc. has modified the IRIX **fsck** program to handle AFS volumes properly, so it is not necessary to replace standard **fsck** on IRIX systems. Transarc does not provide **vfsck** for these systems.

2.6.4. Replacing fsck on NeXT OS Systems

Follow the instructions in this section to replace **fsck** on NeXT OS systems.

Step 1: Move standard **fsck** to a save file, install the AFS-modified **fsck** ("**vfsck**") to the standard location and link standard **fsck** to it.

```
# mv  /usr/etc/fsck  /usr/etc/fsck.old

# cp  /usr/afs/bin/vfsck  /usr/etc/vfsck

# ln  -s  /usr/etc/vfsck  /usr/etc/fsck
```

2.6.5. Replacing fsck on Solaris Systems

Follow the instructions in this section to replace **fsck** on Solaris systems.

For this system type, */etc/fsck* is a link to the driver program distributed as */usr/sbin/fsck*. Rather than replacing either of them, you replace the actual binary distributed as */usr/lib/fs/ufs/fsck*.

Step 1: Move the distributed **fsck** to a save file, install the AFS-modified **fsck** ("**vfsck**") to the standard location and link the distributed **fsck** to it.

```
# mv  /usr/lib/fs/ufs/fsck  /usr/lib/fs/ufs/fsck.old
# cp  /usr/afs/bin/vfsck  /usr/lib/fs/ufs/vfsck
# ln  -s  /usr/lib/fs/ufs/vfsck  /usr/lib/fs/ufs/fsck
```

2.6.6. Replacing fsck on SunOS Systems

Follow the instructions in this section to replace **fsck** on Sun OS systems.

Step 1: Move standard **fsck** to a save file, install the AFS-modified **fsck** ("**vfsck**") to the standard location and link standard **fsck** to it.

```
# mv  /usr/etc/fsck  /usr/etc/fsck.old
# cp  /usr/afs/bin/vfsck  /usr/etc/vfsck
# rm  /etc/fsck
# ln  -s  /usr/etc/vfsck  /etc/fsck
# ln  -s  /usr/etc/vfsck  /usr/etc/fsck
```

2.6.7. Replacing fsck on Other System Types

Follow the instructions in this section to replace **fsck** on any system types other than those already discussed.

Step 1: Move standard **fsck** to a save file, install the AFS-modified **fsck** ("**vfsck**") to the standard location and link standard **fsck** to it.

```
# mv /etc/fsck /etc/fsck.old
# cp /usr/afs/bin/vfsck /etc/vfsck
# ln -s /etc/vfsck /etc/fsck
```

2.7. Starting the BOS Server

You are now ready to start the AFS server processes on this machine. The Basic OverSeer (BOS) Server monitors other AFS server processes on its file server machine, and controls several aspects of their behavior. You therefore must start it first, using the **bosserver** command.

You will invoke **bosserver** with authorization checking turned off. In this mode anyone can issue any **bos** command, because the BOS Server does not check that the issuer has the privileges normally required for issuing the commands. You must work with authorization checking turned off because you have not yet created the Authentication Database entries necessary for authentication; you will do this in section 2.10.

Working with authorization checking turned off is a grave security risk, because it means that anyone who can log onto the machine can issue any AFS command. **Complete all steps in this chapter in one uninterrupted pass and do not leave the machine unattended until you turn on authorization checking in section 2.26.**

As it initializes for the first time, the BOS Server creates the following directories and files. It sets their owner to "root" and sets their mode bits so that no one but the owner can write them; in some cases, it also disables reading. For explanations of the contents and function of these directories and files, refer to chapter 3 of the *AFS System Administrator's Guide*. For further discussion of the mode bit settings, see section 2.31.3 at the end of this chapter.

- */usr/afs/db*
- */usr/afs/etc/CellServDB*
- */usr/afs/etc/ThisCell*
- */usr/afs/local*
- */usr/afs/logs*

The BOS Server also creates a link from */usr/vice* to */usr/afs*. This is necessary because the AFS command interpreters (which generally run on client machines) consult the *CellServDB* and *ThisCell* files in */usr/vice/etc* for information about which cell's server processes they should contact. Because you are installing a file server machine, these files reside under */usr/afs* rather than */usr/vice*. The link from */usr/vice* to */usr/afs* enables the command interpreters to retrieve the information they need from the files. You will replace this link with an actual directory when you make this machine into a client (in section 2.19).

Step 1: Transarc provides your AFS license number in a letter that comes with the Binary Distribution. Copy this number into */usr/afs/etc/License*. The absence of that file prevents the BOS Server from starting, so if you do not know your license number you must contact AFS Product Support before continuing.

Copy your AFS license number into */usr/afs/etc/License*.

echo "<*license number*>**" > /usr/afs/etc/License**

Step 2: Start the BOS Server with authorization checking turned off by invoking **bosserver** with the **-noauth** flag.

/usr/afs/bin/bosserver -noauth &

Step 3: Verify that the BOS Server created the link from */usr/vice* to */usr/afs* that is described above. If not, create the link manually.

ls -ld /usr/vice

If */usr/vice* does not exist, or is not a link to */usr/afs*:

ln -s /usr/afs /usr/vice

2.8. Defining the Cell Name and the Machine's Cell Membership

Now assign your cell's name. You should already have selected a cell name that follows the ARPA Internet Domain System conventions. Select your cell name carefully because it is very complicated to change once set. Chapter 2 of the *AFS System Administrator's Guide* discusses the issues surrounding cell name selection, and explains why the cell name is difficult to change. That chapter also lists important restrictions on the format of a cell name; two of the most important restrictions to note are that the name cannot include uppercase letters or more than 64 characters.

Use the **bos setcellname** command to assign the cell name. It creates two files:

- */usr/afs/etc/ThisCell*, which defines this machine's cell membership and determines which cell's server processes the command interpreters on this machine contact by default
- */usr/afs/etc/CellServDB* file, which lists the cell's database server machines. This machine (being installed) is automatically placed on the list.

Note: Throughout this chapter, substitute the complete Internet host name of the machine you are installing (the local machine) for *machine name*. By convention, machine names use their cell's name as a suffix (for example, *fs1.transarc.com*).

Step 1: Invoke **bos setcellname** to set the cell name.

Issue the **bos setcellname** command, substituting this machine's complete Internet host name for *machine name* and the chosen cell name for *cell name*.

```
# cd /usr/afs/bin

# bos setcellname <machine name> <cell name>
```

You can safely ignore any warnings about **bos** being unable to obtain tickets.

Step 2: Check the cell name as it appears in *CellServDB*. The output also lists the local machine's name as the cell's only database server machine.

```
# bos listhosts <machine name>
```

2.9. Starting the Authentication Server

Start the Authentication Server, *kaserver*, by using the **bos create** command to create an entry for it in */usr/afs/local/BosConfig*. This process runs on database server machines only.

If you are running a previous version of AFS, contact Transarc Corporation for conversion instructions before proceeding.

2.9.1. A Note on Kerberos

AFS's authentication and protection protocols are based on algorithms and other procedures known as "Kerberos," as originally developed by Project Athena at the Massachusetts Institute of Technology. Some cells choose to replace AFS's protocols with Kerberos as obtained directly from Project Athena or other sources. If you wish to do this, contact your AFS Product Support Representative now to learn about necessary modifications to the installation.

2.9.2. Instructions for Installing the Authentication Server

The remaining instructions in these server installation sections direct you to supply the **-cell** argument on all applicable commands. Provide the cell name you assigned in section 2.8. This is mostly a precaution, since the cell is already defined in */usr/afs/etc/ThisCell*.

The remaining instructions also assume that the current working directory is */usr/afs/bin* (you moved there in section 2.8).

Step 1: Start the Authentication Server by issuing the **bos create** command to create a *BosConfig* entry for the *kaserver* process.

Type the following **bos create** command on a single line:

bos create <*machine name*> **kaserver simple /usr/afs/bin/kaserver**
-cell <*cell name*>

Messages may appear periodically on your console, indicating that AFS's distributed database technology, Ubik, is electing a quorum (quorum election is necessary even with a single server). After a few minutes, a final message indicates that the Authentication Server is ready to process requests.

You can safely ignore any messages that tell you to add Kerberos to the */etc/services* file; AFS uses a default value that makes the addition unnecessary.

2.10. Initializing Security Mechanisms

The Authentication Server process you just created maintains the AFS Authentication Database, which records a password entry for each user in your cell and an entry called *afs* for the AFS server processes. In this section you will create initial entries in the Authentication Database and in some related configuration files. These entries are crucial to security in your cell.

Right now you will create two entries in the Authentication Database:

- *admin*, a "generic" administrative account. You can name this account something other than *admin*; if you do so, substitute that name throughout this chapter.

  Use of a generic administrative account means you do not need to grant privilege separately to each system administrator. Instead, each administrator knows *admin*'s password and authenticates under that identity when performing tasks that require administrative privilege.

  Like all user entries in the Authentication Database, *admin*'s entry records the user name (*admin*) and a password, scrambled into an encryption key.

- *afs*, the entry for the AFS server processes. No one will ever login as *afs*, but the Authentication Server's Ticket Granting Service (TGS) module uses the password field to encrypt the server tickets that AFS clients must present to servers during mutual authentication. (See chapter 1 in the *AFS System Administrator's Guide* to review the role of server encryption keys in mutual authentication).

After creating the accounts, you will then grant administrative privilege to *admin*. Issuing the **kas setfields** command in step 2 makes *admin* able to issue any **kas** command. Issuing **bos adduser** in step 4 adds *admin* to */usr/afs/etc/UserList*, which makes it able to issue any **bos** or **vos** command (the BOS Server, Volume Server and Volume Location Server consult *UserList* to determine privilege).

In step 5, you will place the AFS server encryption key into */usr/afs/etc/KeyFile*. The AFS server processes refer to this file to learn the server encryption key when they need to decrypt server tickets.

Note: Most **kas** commands always prompt for a password (in the following steps it is `Password for root`). Because authorization checking is currently disabled, the Authentication Server does not check the validity of the password provided. You can type any character string, such as a space, and press <**Return**>.

You can also ignore the error messages that appear after you provide the password. They appear only because authorization checking is disabled, and do not indicate actual errors.

Step 1: Create Authentication Database entries for *afs* and *admin*.

Issue the **kas create** command to create Authentication Database entries for *admin* and *afs*. Substitute the desired passwords for *afs_passwd* and *admin_passwd*, using any string longer than six characters. **Be sure to remember both passwords.**

In the following example commands, the passwords are not provided on the command line, so **kas** prompts for them. The advantage is that the passwords do not echo visibly on the screen, but if you wish to type them on the command line, place them between the account name and the **-cell** flag.

Note: at the `Password for root` prompt, press space and <**Return**>. Ignore any error messages that appear.

kas create afs -cell <*cell name*>
```
Password for root:
initial_password: <afs_passwd>
Verifying, please re-enter initial_password: <afs_passwd>
```

kas create admin -cell <*cell name*>
```
Password for root:
initial_password: <admin_passwd>
Verifying, please re-enter initial_password: <admin_passwd>
```

Step 2: Define *admin* to be a privileged issuer of **kas** commands.

Issue the **kas setfields** command to designate *admin* as a privileged administrator with respect to the Authentication Database.

Note: at the `Password for root` prompt, press space and <**Return**>. Ignore any error messages that appear.

kas setfields admin -flags admin -cell <*cell name*>
```
Password for root:
```

Step 3: Verify that *admin* has the `ADMIN` flag in its Authentication Database entry.

Issue the **kas examine** command. The `ADMIN` flag should appear near the top of the output, indicating that *admin* is privileged.

Note: at the `Password for root` prompt, press space and <**Return**>. Ignore any error messages that appear.

kas examine admin -cell <*cell name*>
```
Password for root:
```

Step 4: Define *admin* to be a privileged issuer of **bos** and **vos** commands. Doing so has no immediate consequence, but becomes important after you enable authorization checking in section 2.26.

Issue **bos adduser** to add *admin* to */usr/afs/etc/UserList*.

bos adduser <*machine name*> **admin -cell** <*cell name*>

Step 5: Add the AFS server encryption key (*afs_passwd*) to */usr/afs/etc/KeyFile*.

Issue the **bos addkey** command to define *afs_passwd* as the first server encryption key, with key version number 0. **Be sure** to use the same *afs_passwd* as in step 1.

In the following example commands, *afs_passwd* is not provided on the command line, so **bos** prompts for it. The advantage is that the password does not echo visibly on the screen, but if you wish to type it on the command line, place it between *machine name* and the **-kvno** switch.

bos addkey <*machine name*> **-kvno 0 -cell** <*cell name*>
`input key:` <*afs_passwd*>

Step 6: Use **kas examine** and **bos listkeys** to verify that the key derived from *afs_passwd* is the same in the Authentication Database *afs* entry and in the *KeyFile*. Note that you can see the actual octal numbers making up the key only because authorization checking is temporarily disabled. If the keys are different, perform steps 1 and 5 again, making sure that *afs_passwd* is the same in both cases.

Note: You should change the server encryption key at least once per month from now on. When you do, follow the instructions in chapter 10 of the *AFS System Administrator's Guide*, being sure to change the key both in the *KeyFile* and in the Authentication Database.

Note: at the `Password for root` prompt, press space and <**Return**>. Ignore any error messages that appear.

kas examine afs
```
Password for root:
```

bos listkeys <*machine name*> **-cell** <*cell name*>

2.11. Starting the Protection Server

Start the Protection Server, *ptserver*, which maintains the Protection Database. This process runs on database server machines only.

As it initializes for the first time, the Protection Server automatically creates "system" entries in the Database, including the *system:administrators* group, and assigns them AFS UIDs. You will add *admin* to *system:administrators* to enable it to issue privileged **pts** and **fs** commands.

If you are running a previous version of AFS, contact Transarc Corporation for conversion instructions before proceeding.

Step 1: Start the Protection Server by issuing the **bos create** command to create a *BosConfig* entry for it.

Type the following **bos create** command on a single line:

bos create <*machine name*> **ptserver simple /usr/afs/bin/ptserver -cell** <*cell name*>

Messages may appear periodically on your console, indicating that AFS's distributed database technology, Ubik, is electing a quorum (quorum election is necessary even with a single server). Wait a few minutes after the last message, to guarantee that the Protection Server is ready to execute requests.

Step 2: Add *admin* to the *system:administrators* group.

Create a Protection Database entry for *admin* and add its name to the *system:administrators* group. If you substituted a name other than *admin* in section 2.10, use that name here. Presuming *admin* is the first Protection Database entry you have created, it will be assigned the AFS UID 1. If you wish to assign a different AFS UID, use the **-id** argument on **pts createuser**.

If your cell's local password file (*/etc/passwd* or equivalent) already has an entry for *admin*, you should use the **-id** argument to make *admin*'s AFS UID the same as its UNIX UID. See chapters 16 and 17 of the *AFS System Administrator's Guide* for further discussion of matching AFS and UNIX UIDs.

pts createuser -name admin -cell <*cell name*> **[-id** <*AFS UID*>**]**

pts add admin system:administrators -cell <*cell name*>

Step 3: Verify that *admin* was correctly added to *system:administrators*.

Check that the output lists *admin* as a member of *system:administrators*.

pts membership system:administrators -cell <*cell name*>

2.12. Starting the Volume Location Server

Start the Volume Location Server, *vlserver*, by using the **bos create** command to create an entry for it in */usr/afs/local/BosConfig*. This process maintains the Volume Location Database (VLDB), and runs on database server machines only.

If you are running a previous version of AFS, contact Transarc Corporation for conversion instructions before proceeding.

Step 1: Start the Volume Location Server by issuing the **bos create** command to create a *BosConfig* entry for it.

Type the following **bos create** command on a single line:

bos create <*machine name*> **vlserver simple /usr/afs/bin/vlserver -cell** <*cell name*>

Messages may appear periodically on your console, indicating that AFS's distributed database technology, Ubik, is electing a quorum (quorum election is necessary even with a single server). Wait a few minutes after the last message, to guarantee that the VL Server is ready to execute requests.

2.13. Starting the Backup Server

Start the Backup Server, *buserver*, by using the **bos create** command to create an entry for it in */usr/afs/local/BosConfig*. This process maintains the Backup Database, and runs on database server machines only. The chapter entitled "Backing Up the System" in the *AFS System Administrator's Guide* details the other instructions you must perform before actually using the Backup System.

If you are running a previous version of AFS, contact Transarc Corporation for conversion instructions before proceeding.

Step 1: Start the Backup Server by issuing the **bos create** command to create a *BosConfig* entry for it.

Type the following **bos create** command on a single line:

bos create <*machine name*> **buserver simple /usr/afs/bin/buserver**
-cell <*cell name*>

Messages may appear periodically on your console, indicating that AFS's distributed database technology, Ubik, is electing a quorum (quorum election is necessary even with a single server). Wait a few minutes after the last message, to guarantee that the Backup Server is ready to execute requests.

2.14. Starting the File Server, Volume Server, and Salvager

Start the *fs* process, which "binds together" the File Server, Volume Server and Salvager (*fileserver*, *volserver* and *salvager* processes).

Step 1: Start the *fs* process by issuing the **bos create** command to create a *BosConfig* entry for it.

Type the following **bos create** command on a single line:

bos create <*machine name*> **fs fs /usr/afs/bin/fileserver /usr/afs/bin/volserver /usr/afs/bin/salvager -cell** <*cell name*>

You may see a message regarding the initialization of the VLDB and a few error messages of the form

```
fsync_clientinit failed (will sleep and retry): connection refused
```

These messages appear because the *volserver* process cannot start operation until the *fileserver* process initializes. Initialization can take a while, especially if you already have a large number of existing AFS files. Wait a few minutes after the last such message, to guarantee that both component processes have started successfully.

One way to check that the *fs* process has started successfully is to issue the following command. Check that the entry for `fs` in the output reports two "proc starts."

bos status <*machine name*> **-long**

Step 2: What you do in this step depends on whether you have previously run AFS on this machine or not. If you have not run AFS before, you will create your cell's root volume. If you have run AFS before, you will synchronize the VLDB with the volumes that already exist.

If you have not previously run AFS on this file server machine, create the root volume for AFS, `root.afs`. For <*partition name*>, substitute the name of one of the AFS partitions you created in section 2.5 (such as */vicepa*).

vos create <*machine name*> <*partition name*> **root.afs -cell** <*cell name*>

Messages may appear indicating that the Volume Server created `root.afs` on the indicated partition on the local machine, and created a VLDB entry for it.

OR

If you have previously run AFS on this file server machine, synchronize the VLDB to reflect the state of volumes on the local machine. The synchronization may take several minutes; to follow its progress, use the **-v** flag.

vos syncvldb <*machine name*> **-cell** <*cell name*> **-v**

vos syncserv <*machine name*> **-cell** <*cell name*> **-v**

2.15. Starting the Server Portion of the Update Server

Start the server portion of the Update Server, *upserver*, to distribute this machine's copies of files from

- */usr/afs/bin* to other file server machines of its system type.

  Distributing its copies of the binary files in */usr/afs/bin* is what makes this machine the ***binary distribution machine*** for its system type. The other file server machines of its system type run the appropriate client portion of the Update Server, *upclientbin*. This means that the administrator should install updated binaries only on the binary distribution machine of each system type.

- */usr/afs/etc* to all other file server machines, if you are running the United States edition of AFS. AFS customers in the United States and Canada receive this edition.

  Distributing its copies of the configuration files in */usr/afs/etc* is what makes this machine the ***system control machine***. The other file server machines in the cell run the appropriate client portion of the Update Server, *upclientetc*. This means that administrators at sites using the United States edition of AFS should update the files in */usr/afs/etc* only on the system control machine.

  Some of the files in */usr/afs/etc*, particularly the *KeyFile*, are very sensitive (crucial to cell security), so it is important that the *upserver* process encrypt them before distribution across the network. To guarantee this, use the **-crypt** flag as indicated on the following **upserver** initialization command.

  Sites using the international edition of AFS should not use the Update Server to distribute the contents of */usr/afs/etc* (thus international cells do not run a system control machine). Due to United States government export restrictions, the international edition of AFS does not include the encryption routines that the Update Server uses to encrypt information before sending it across the network. Instead of using the *upserver* process to distribute the contents of */usr/afs/etc*, you must update the configuration files on each file server machine individually (for example, when you update the *KeyFile*). The **bos** commands you use to update the configuration files are able to encrypt data to protect it as it crosses the network. Instructions in the *AFS System Administrator's Guide* explain how to update configuration files on individual machines.

The binaries in */usr/afs/bin* are not important to cell security, so it is not necessary to encrypt them before distribution across the network. With both editions of AFS, you will mark */usr/afs/bin* with the **upserver** initialization command's **-clear** flag, to indicate that the *upserver* process should encrypt the contents of */usr/afs/bin* only if an *upclientbin* process requests them that way.

Note that the server and client portions of the Update Server always mutually authenticate with one another, in both the United States and international versions of AFS, and regardless of whether you use the **-clear** or **-crypt** flag. This protects their communications from eavesdropping to some degree.

See the *AFS Command Reference Manual* entries on **upclient** and **upserver** for more information on these commands.

Step 1: Start the server portion of the Update Server by issuing the **bos create** command to create a *BosConfig* entry for it.

If using the United States edition of AFS, type the following command on a single line:

bos create <*machine name*> **upserver simple "/usr/afs/bin/upserver -crypt /usr/afs/etc -clear /usr/afs/bin" -cell** <*cell name*>

If using the international edition of AFS, type the following command on a single line:

bos create <*machine name*> **upserver simple "/usr/afs/bin/upserver -clear /usr/afs/bin" -cell** <*cell name*>

2.16. Starting the Controller for NTPD

In this section you start the AFS process, *runntp*, that controls the Network Time Protocol Daemon (NTPD). This daemon runs on all of on your cell's file server machines, and keeps their internal clocks synchronized. As the cell's system control machine, this machine is special: it is the only one that refers to a machine outside the cell as its time standard or source. All other file server machines in the cell will refer to this machine as their time source.

Keeping the clocks on your cell's file server machines synchronized is crucial to the correct operation of AFS's distributed database technology (Ubik). Chapter 2 of the *AFS System Administrator's Guide* explains in some detail how unsynchronized clocks can disturb Ubik's performance and cause service outages in your cell.

Choosing an appropriate external time source is important, but involves more considerations than can be discussed in the space available here. If you need help in selecting a source, contact your AFS Product Support Representative. The *AFS Command Reference Manual* provides more information on the **runntp** command's arguments.

As the *runntp* process initializes NTPD, trace messages may appear on the console. You may safely ignore these messages, but might find them interesting if you understand how NTPD works.

Step 1: Verify that **ntpd** and **ntpdc** exist in */usr/afs/bin*.

```
# ls /usr/afs/bin
```

Step 2: Start the *runntp* process on the system control machine to control NTPD.

Initialize *runntp* by issuing the following command. For *time server machine(s)*, substitute the IP address of one or more machines that will serve as an external time source, separating each name with a space. Include this argument on this machine only. For help in selecting appropriate machine(s), contact your AFS Product Support Representative.

The **-localclock** flag tells NTPD to rely on its internal clock during periods when the machine loses contact with its external time source(s). You should only use it on this machine, and only if your cell is subject to frequent network outages that might separate this machine from its external time source.

Type the command on a single line.

```
# bos create <system_control_machine> runntp simple
                "/usr/afs/bin/runntp -localclock <time server machine(s)>"
```

2.17. Completing the Installation of Server Functionality

You have now started all the AFS server processes that will run on this machine. In this section, you will reboot the machine to make sure that all processes will restart successfully.

The first step is to make sure that the **bosserver** initialization command appears in the machine's initialization file. At each reboot, the BOS Server starts up in response to this command, and starts all processes in */usr/afs/local/BosConfig* that have status flag `Run` (each **bos create** command you issued in this chapter set the process' status flag to `Run`).

Step 1: Verify that the machine's initialization file invokes **bosserver**, so that the BOS Server starts automatically at each file server reboot.

If the machine does not run IRIX or Solaris, add the following lines to */etc/rc* or equivalent, after the lines that configure the network, mount all file systems, and invoke a kernel dynamic loader.

```
if [ -f /usr/afs/bin/bosserver ]; then
   echo 'Starting bosserver' > /dev/console
      /usr/afs/bin/bosserver &
fi
```

If the machine runs IRIX or Solaris, no action is necessary. The *afs.rc* script includes tests that result in automatic BOS Server start up if appropriate (you installed *afs.rc* as part of dynamic kernel loading, in section 2.3.4 for IRIX or section 2.3.6 for Solaris).

Step 2: Shutdown AFS server processes in preparation for reboot. You may wish to check if any users are logged on and notify them first.

Using the **-wait** flag on the **bos shutdown** command guarantees that all processes shut down before the command line prompt returns.

Shutdown the server processes on the machine in preparation for reboot.

cd /usr/afs/bin

bos shutdown <*machine name*> **-wait**

Step 3: Reboot the machine and login again as "root."

The BOS Server starts automatically because of the **bosserver** command you added to */etc/rc* or equivalent in step 1.

sync

/etc/reboot

`login:` **root**
`Password:`

Step 4: Verify that all server processes are running normally. The output for each process should read "`Currently running normally`."

```
# bos status <machine name>
```

Step 5: Turn off authorization checking once again, in preparation for installing the client functionality on this machine. Authorization checking is currently on because the **bosserver** command in */etc/rc* (or equivalent) does not—and should not—use the **-noauth** flag.

Remember that turning off authorization checking is a grave security risk. Perform the remaining steps in this chapter in one uninterrupted session. You will turn on authorization checking in section 2.26.

Turn off authorization checking, and then proceed immediately to the remainder of this chapter.

```
# echo "" > /usr/afs/local/NoAuth
```

2.18. Overview: Installing Client Functionality

Now that the machine is an AFS file server machine, database server machine, system control machine, and binary distribution machine, you will also make it a client machine.

In the following sections, you will:

1. Load the client binaries onto the local disk.
2. Define the machine's cell membership for client processes.
3. Create the client version of *CellServDB*.
4. Define cache location and size.
5. Create the */afs* directory and start the Cache Manager.

2.19. Loading Client Files on the First AFS Machine

Several client binaries must reside on the local disk of every AFS client machine, including the **afsd** binary which initializes the Cache Manager. Follow the instructions suitable for the machine configuration you are are using.

2.19.1. Loading Client Files Using a Local Tape Drive

The fourth **tar** set on the tape contains the */usr/vice/etc* directory, consisting of **afsd** and the files needed to incorporate AFS using the dynamic kernel loader for this system type (*dkload* or equivalent). The following steps load it into */usr/vice/etc*.

Step 1: Remove the link from */usr/vice* to */usr/afs*, which was created in section 2.7.

```
# rm /usr/vice
```

Step 2: Create the */usr/vice/etc* directory.

```
# mkdir /usr/vice

# mkdir /usr/vice/etc
```

Step 3: Mount the Binary Distribution tape and load the fourth **tar** set into */usr/vice/etc*.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/vice/etc

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 3

# tar xvf /dev/<device>
```

Step 4: The directory for dynamic kernel loader files now resides in two places: */usr/afs/bin* (appropriate for a file server) and */usr/vice/etc* (appropriate for a client). Because this machine will definitely remain a file server, you should remove the dynamic loader directory under */usr/vice/etc/* to avoid confusion. The machine's initialization file should also refer to the directory under */usr/afs/bin*.

Remove the client version of dynamic kernel loader files.

cd /usr/vice/etc

For IRIX systems:

rm -rf sgiload

ln -s /usr/afs/bin/sgiload sgiload

For Solaris and SunOS systems:

rm -rf modload

ln -s /usr/afs/bin/modload modload

For all other system types that use a dynamic loader, including SunOS if you use **dkload**:

rm -rf dkload

ln -s /usr/afs/bin/dkload dkload

2.19.2. Loading Client Files from a Remote Machine

This section assumes that the needed client files already reside in the */usr/afsws/root.client/usr/vice/etc* directory on a remote machine. In this section you will copy the files from that directory into */usr/vice/etc* on the local machine using **ftp**, NFS or another file transfer method.

Step 1: **Working on the local machine,** remove the link from */usr/vice* to */usr/afs*, which was created in section 2.7.

rm /usr/vice

Step 2: **Working on the local machine,** create the */usr/vice/etc* directory.

mkdir /usr/vice

mkdir /usr/vice/etc

Step 3: Copy the files in */usr/afsws/root.client/usr/vice/etc* (on the remote machine) to */usr/vice/etc* (on the local machine) using **ftp**, NFS, or another network file transfer method.

Step 4: The directory for dynamic kernel loader files now resides in two places: */usr/afs/bin* (appropriate for a file server) and */usr/vice/etc* (appropriate for a client). Because this machine will definitely remain a file server, you should remove the dynamic loader directory under */usr/vice/etc/* to avoid confusion. The machine's initialization file should also refer to the directory under */usr/afs/bin*.

Remove the client version of dynamic kernel loader files.

cd /usr/vice/etc

For IRIX systems:

rm -rf sgiload

ln -s /usr/afs/bin/sgiload sgiload

For Solaris and SunOS systems:

rm -rf modload

ln -s /usr/afs/bin/modload modload

For all other system types that use a dynamic loader, including SunOS if you use **dkload**:

rm -rf dkload

ln -s /usr/afs/bin/dkload dkload

2.20. Defining the Client Machine's Cell Membership

Every AFS client machine must have a copy of the */usr/vice/etc/ThisCell* file on its local disk, to define its cell membership for the AFS client programs that run on it. The */usr/afs/etc/ThisCell* file you created in section 2.8 is used only by server processes.

Among other functions, on a client *ThisCell* determines:

- the cell in which users authenticate by default
- which cell's file server processes this machine's AFS command interpreters contact by default

Step 1: Place your cell name into */usr/vice/etc/ThisCell*:

```
# echo "<cellname> " >  /usr/vice/etc/ThisCell
```

For example, the appropriate command in the Transarc Corporation cell is

```
# echo "transarc.com" > /usr/vice/etc/ThisCell
```

2.21. Creating the Client Version of CellServDB

Every client machine's */usr/vice/etc/CellServDB* file lists the database server machines in each cell that the Cache Manager can contact. If a cell is not listed in this file, or its list of database server machines is wrong, then users working on this machine will be unable to access the cell's file tree. For the Cache Manager to perform properly, the *CellServDB* file must be accurate at all times. Refer to the *AFS System Administrator's Guide* for instructions on keeping this file up-to-date after initial creation.

The Cache Manager consults */usr/vice/etc/CellServDB* only once per reboot. As **afsd** initializes the Cache Manager, it copies the contents of *CellServDB* into the kernel; until the next reboot, the Cache Manager consults this in-kernel list of database server machines. Section 2.30 and the *AFS System Administrator's Guide* explain how to use the **fs newcell** command to update the kernel list without rebooting.

A sample *CellServDB* file is included in the AFS Binary Distribution as */usr/vice/etc/CellServDB.sample*. It includes all of the cells that had agreed to be advertised at the time your Binary Distribution tape was made; the majority of the list is ordered by Internet address. Follow the instructions in this section to add your local cell's database server machines to this machine's *CellServDB*. Section 2.29 provides instructions for making all of the cells accessible, and section 2.30 shows how to add more foreign cells in the future.

Each cell's entry in *CellServDB* must have the following format.

- The first line must begin with the > character, followed by the cell's Internet domain name. The domain name can be followed by a # sign (indicating a comment) and a comment that explains the name.
- Each subsequent line lists one database server machine in the cell. Each database server machine line must contain the Internet address of the server in the standard four-component decimal form (e.g., 158.98.3.2), followed by a # sign and the machine's Internet host name. In this case, the # sign does not indicate a comment; the hostname that follows is a required field. The Cache Manager attempts to contact the cell using the specified name, referring to the Internet address only if contacting by name fails.

An example extract from *CellServDB.sample* appears after the instructions.

Step 1: Working in the */usr/vice/etc* directory, rename *CellServDB.sample* to *CellServDB*.

```
# cd /usr/vice/etc

# mv CellServDB.sample CellServDB
```

Step 2: Use a text editor to add an entry for your cell to *CellServDB*. It does not matter where you put the entry, but for ease of future access you may wish to place it either at the beginning of the file or in Internet address order.

Be sure to format your cell's entry as described above. The Cache Manager cannot read *CellServDB* if it contains formatting errors (such as extra blank lines between the database server machine lines).

Here is an example for a cell called *school.edu* with a single database server machine called *first.fs.school.edu* at address 128.2.9.7:

```
>school.edu      #our home cell
128.2.9.7        #first.fs.school.edu
```

Step 3: If *CellServDB* includes cells that you do not wish users of this machine to access, remove their entries.

To make all of the cells in *CellServDB* accessible to users of this machine, see the instructions in section 2.29.

The following extract from *CellServDB.sample* illustrates correct format. (However, the machine names and addresses are subject to change. Section 2.30 explains how to obtain current information about database server machines.)

```
>athena.mit.edu           #MIT/Athena cell
18.72.0.43                      #orf.mit.edu
18.80.0.2                       #maeander.mit.edu
18.70.0.6                       #prill.mit.edu
>andrew.cmu.edu           #Carnegie Mellon University - Campus
128.2.10.2                      #vice2.fs.andrew.cmu.edu
128.2.10.7                      #vice7.fs.andrew.cmu.edu
128.2.10.11                     #vice11.fs.andrew.cmu.edu
>cs.cmu.edu               #Carnegie Mellon University - SCS
128.2.242.86                    #lemon.srv.cs.cmu.edu
128.2.217.45                    #apple.srv.cs.cmu.edu
128.2.222.199                   #papaya.srv.cs.cmu.edu
>umich.edu                #University of Michigan - Campus
141.211.168.24                  #beachhead.ifs.umich.edu
141.211.168.25                  #toehold.ifs.umich.edu
141.211.168.28                  #bastion.ifs.umich.edu
>transarc.com             #Transarc Corporation
158.98.3.2                      #ernie.transarc.com
158.98.3.3                      #bigbird.transarc.com
158.98.14.3                     #oscar.transarc.com
```

2.22. Setting Up the Cache

Every AFS client must have a cache in which to store local copies of files brought over from file server machines. The Cache Manager can cache either on disk or in machine memory.

For both types of caching, **afsd** consults the */usr/vice/etc/cacheinfo* file as it initializes the Cache Manager and cache, to learn the defaults for cache size and where to mount AFS locally. For disk caches, it also consults the file to learn cache location. You must create this file for both types of caching.

The file has three fields:

- the first field defines where to mount AFS on the local disk. The standard choice is */afs*.
- the second field defines the local disk directory to use for caching, in the case of a disk cache. The standard choice is */usr/vice/cache*, but you could specify a different directory to take advantage of more space on other partitions. Something must appear in this field even if the machine uses memory caching.
- the third field defines cache size as a number of kilobyte (1024 byte) blocks. Make it as large as possible, but do not make the cache larger than 90% to 95% of the space available on the partition housing */usr/vice/cache* or in memory: the cache implementation itself requires a small amount of space. For AIX 3.2 systems using a disk cache, cache size cannot exceed 85% of the disk capacity reported by the **df** command. This difference between AIX 3.2 and other systems is because the output of AIX 3.2 **df** shows actual disk capacity and use, whereas BSD-based versions "hide" about 10% of disk capacity to allow for overage.

  Violating this restriction on cache size can cause errors or worsened performance.

  AFS versions 3.2 and 3.3 allow a cache as large as 1 gigabyte. Earlier versions of AFS impose a cache size limit of 700 megabytes.

  Disk caches smaller than 5 megabytes do not generally perform well, and you may find the performance of caches smaller than 10 megabytes unsatisfactory, particularly on system types that have large binary files. Deciding on a suitable upper limit is more difficult. The point at which enlarging the cache does not really improve performance depends on the number of users on the machine, the size of the files they are accessing, and other factors. A cache larger than 40 megabytes is probably unnecessary on a machine serving only a few users accessing files that are not huge. Machines serving multiple users may perform better with a cache of 60 to 70 megabytes.

  Memory caches smaller than 1 megabyte are nonfunctional, and most users find the performance of caches smaller than 5 megabytes to be unsatisfactory. Again, this depends on the number of users working on the machine, and the number of processes running. Machines running only a few processes may be able to use a smaller memory cache.

2.22.1. Setting Up a Disk Cache

Step 1: Create the cache directory. This example instruction shows the standard location, */usr/vice/cache*.

```
# mkdir /usr/vice/cache
```

Step 2: Create the *cacheinfo* file to define the boot-time defaults discussed above. This example instruction shows the standard mount location, */afs*, and the standard cache location, */usr/vice/cache*.

```
# echo "/afs:/usr/vice/cache:<#blocks>" > /usr/vice/etc/cacheinfo
```

For example, to devote 10000 one-kilobyte blocks to the cache directory on this machine, type:

```
# echo "/afs:/usr/vice/cache:10000" >/usr/vice/etc/cacheinfo
```

2.22.2. Setting Up a Memory Cache

Step 1: Create the *cacheinfo* file to define the boot-time defaults discussed above. This example instruction shows the standard mount location, */afs*, and the standard cache location, */usr/vice/cache*. The location specified is irrelevant for a memory cache, but a value must be provided.

```
# echo "/afs:/usr/vice/cache:<#blocks>" > /usr/vice/etc/cacheinfo
```

For example, to devote 10000 kilobytes of memory to caching on this client machine, type:

```
# echo "/afs:/usr/vice/cache:10000" >/usr/vice/etc/cacheinfo
```

2.23. Creating */afs* and Starting the Cache Manager

As mentioned previously, the Cache Manager mounts AFS at the local */afs* directory. In this section you create that directory and then run **afsd** to initialize the Cache Manager.

You should also add **afsd** to the machine's initialization file (*/etc/rc* or equivalent), so that it runs automatically at each reboot. If **afsd** does not run at each reboot, the Cache Manager will not exist on this machine, and it will not function as an AFS client.

The **afsd** program sets several cache configuration parameters as it initializes, and starts up daemons that improve performance. As described completely in the *AFS Command Reference Manual*, you can use the **afsd** command's arguments to alter these parameters and/or the number of daemons. Depending on the machine's cache size, its amount of RAM and how many people work on it, you may be able to improve its performance as a client by overriding default values.

AFS also provides a simpler alternative to setting **afsd**'s arguments individually, in the form of sets of parameter settings that are appropriate for client machines of three different sizes. For system types other than IRIX and Solaris, the settings are specified in three initialization scripts distributed in */usr/vice/etc/dkload*. The scripts are appropriate only for machines with a disk cache. Both the *AFS Command Reference Manual* and chapter 13 of the *AFS System Administrator's Guide* discuss these scripts in more detail. They are called:

- *rc.afsd.small*, which configures the Cache Manager appropriately for a "small" machine with a single user, about 8 megabytes of RAM and a 20-megabyte cache. It sets **-stat** to 300, **-dcache** to 100, **-daemons** to 2, and **-volumes** to 50.
- *rc.afsd.medium*, which configures the Cache Manager appropriately for a "medium" machine with two to six users, about 16 megabytes of RAM and a 40-megabyte cache. It sets **-stat** to 2000, **-dcache** to 800, **-daemons** to 3, and **-volumes** to 70.
- *rc.afsd.large*, which configures the Cache Manager appropriately for "large" machine with five to ten users, about 32 megabytes of RAM and a 100-megabyte cache. It sets **-stat** to 2800, **-dcache** to 2400, **-daemons** to 5, and **-volumes** to 128.

For IRIX and Solaris systems, the parameter settings are defined in the *afs.rc* initialization script, which you installed into */etc/init.d* as part of dynamic kernel loading (in section 2.3.4 for IRIX or section 2.3.6 for Solaris). The script defines `LARGE`, `MEDIUM` and `SMALL` values for the `OPTIONS` variable, which is then included on the `afsd` command line in the script. The *afs.rc* script is distributed with `OPTIONS` set to `$MEDIUM`, but you may change this as desired.

Step 1: Create the */afs* directory and verify that it is empty.

```
# mkdir /afs

# ls /afs
```

Step 2: Invoke **afsd**. Use the **-nosettime** flag because this is a file server machine that is also a client. The flag prevents the machine from picking a file server machine in the cell as its source for the correct time, which client machines normally do. File server machines synchronize their clocks using the *runntp* process instead.

For a memory cache, also use the **-memcache** flag to indicate that the cache should be in memory rather than on disk.

With a disk cache, starting up **afsd** for the first time on a machine can take up to 10 minutes, because the Cache Manager has to create all of the structures needed for caching (V files). Starting up **afsd** at future reboots does not take nearly this long, since the structures already exist. With a memory cache, the memory structures must be allocated at each reboot, but the process is equally quick each time.

Because of the potentially long start up, you may wish to put the following commands in the background. Even if you do, **afsd** must initialize completely before you issue the **klog** command in step 4. Console messages will trace the progress of the initialization and indicate when it is complete.

For a disk cache:

Invoke **afsd** with the **-nosettime** flag. On system types other than IRIX and Solaris, you can substitute one of the configuration scripts (such as *rc.afsd.medium*) for **afsd** in the following command if desired (but you still must type the **-nosettime** and **-verbose** flags). Since the three scripts are not available for IRIX or Solaris systems, you must type on the command line any additional configuration parameters you wish to set.

```
# /usr/vice/etc/afsd -nosettime -verbose [&]
```

For a memory cache:

Invoke **afsd** with the **-nosettime** and **-memcache** flags. You may specify values for other parameters if desired.

```
# /usr/vice/etc/afsd -nosettime -memcache -verbose [&]
```

Step 3: If you intend for the machine to remain an AFS client after you complete its installation, invoke **afsd** in its initialization file.

For a disk cache:

If the machine does not run IRIX or Solaris, add **afsd** to its initialization file (*/etc/rc* or equivalent), after the commands that invoke a dynamic kernel loader and the BOS Server, but before any NFS daemons (**nfsd**) are started. To invoke one of the configuration scripts described in the introduction to this section (such as *rc.afsd.medium*), substitute it for `afsd`:

```
/usr/vice/etc/afsd -nosettime > /dev/console
```

If the machine runs IRIX or Solaris, verify that the `OPTIONS` variable in */etc/init.d/afs.rc* is set to the appropriate value; as distributed, it is `$MEDIUM`. For Solaris systems, remember to add the `-nosettime` flag. For IRIX systems, there is a test for the `TIMESYNC` variable that automatically adds the `-nosettime` flag if appropriate.

For a memory cache:

If the machine does not run IRIX or Solaris, add the following to */etc/rc* or equivalent. Add other configuration parameters if desired, but remember that the "*large*," "*medium*" and "*small*" scripts cannot be used with a memory cache.

```
/usr/vice/etc/afsd -nosettime -memcache > /dev/console
```

If the machine runs IRIX or Solaris, verify that the `OPTIONS` variable in */etc/init.d/afs.rc* is not set to any of `$LARGE`, `$MEDIUM` or `$SMALL`; these values cannot be used with a memory cache. For Solaris systems, remember to add the `-nosettime` flag to the `afsd` command line. For IRIX systems, there is a test for the `TIMESYNC` variable that automatically adds the `-nosettime` flag if appropriate.

For system types other than IRIX and Solaris, the following should now appear in the machine's initialization file in the indicated order:

- command(s) mounting the file systems
- command(s) loading *nfs.ext* (only for RS/6000s running NFS and AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)
- dynamic kernel loader command(s), unless AFS was built into the kernel
- **bosserver**
- **afsd**
- **nfsd** (if the machine is to be an NFS server)

Step 4: To verify that the Cache Manager and other client programs and processes are functioning correctly, authenticate as *admin*, using the password (*admin_passwd*) you defined in section 2.10. If you named your generic administrative account something other than *admin*, substitute that name here.

Wait to issue this command until console messages indicate that **afsd** has finished initializing.

```
# /usr/afs/bin/klog admin
Password: <admin_passwd>
```

Step 5: Use the **tokens** command to verify that **klog** worked correctly. The output should list tokens for *admin*'s AFS UID that are good for your cell. If either **klog** or **tokens** seems to have malfunctioned, you may wish to contact AFS Product Support before proceeding.

```
# /usr/afs/bin/tokens
```

Step 6: Issue the **fs checkvolumes** command.

```
# /usr/afs/bin/fs checkvolumes
```

2.24. Overview: Completing the Installation of the First AFS Machine

The machine is now a fully functional AFS file server and client machine. However, your cell's file tree does not exist yet. In this section you will create the upper levels of the tree, among other procedures. The procedures are:

1. Create and mount top-level volumes.
2. Turn on authorization checking.
3. Create and mount volumes to store system binaries in AFS.
4. Enable access to Transarc and other cells.
5. Institute additional security measures.
6. Replace the standard **login** binary with a version that both authenticates with AFS and logs into the local UNIX file system, if the machine will remain an AFS client machine.
7. Alter file system clean-up scripts on some system types, if the machine will remain an AFS client machine.
8. Remove client functionality if desired.

2.25. Setting Up the Top Levels of the AFS Tree

If this is the first time you have installed AFS, you should now set up the top levels of your cell's AFS file tree.

If you have run a previous version of AFS in your cell before, your file tree should already exist. Proceed to section 2.26.

You created the `root.afs` volume in section 2.14, and the Cache Manager mounted it automatically at */afs* on this machine when you ran **afsd** in section 2.23. You now set the access control list (ACL) on */afs*—creation, mounting, and setting the ACL are the three steps required in creating any volume. The default ACL on a new volume grants all seven access rights to *system:administrators*. In this section you add the READ and LOOKUP rights for *system:anyuser*; users must have these rights for the Cache Manager to "pass through" `root.afs` when accessing volumes mounted lower in the file tree.

After setting the ACL on `root.afs`, you will create your cell's `root.cell` volume, mount it at the second level in the file tree and set the ACL. Actually, you will create both a ReadWrite and a regular mount point for `root.cell`, so that Cache Managers can access your cell's file tree via both a "ReadWrite" and a "ReadOnly" path. This essentially creates separate read-only and read-write trees. Chapter 5 of the *AFS System Administrator's Guide* further explains the concept of ReadOnly and ReadWrite paths (mount point traversal).

After that, you will replicate both `root.afs` and `root.cell`. This is required if you want to replicate any other volumes in your cell, because all volumes mounted above a replicated volume must themselves be replicated in order for the Cache Manager to access the replica.

Note: Once you replicate `root.afs`, the Cache Manager will access the ReadOnly version of the volume—`root.afs.readonly`—if it is available. Whenever you want to make changes to the contents of this volume (when, for example, you mount another cell's `root.cell` volume at the second level in your file tree), you must create a temporary mount point for the ReadWrite version (`root.afs`), make the changes, re-release the volume and remove the temporary mount point. See section 2.29 for instructions.

Step 1: Grant *system:anyuser* the READ and LOOKUP rights on the ACL for */afs*.

```
# /usr/afs/bin/fs  sa  /afs  system:anyuser  rl
```

Step 2: Create the `root.cell` volume and mount it at the second level in the file tree to serve as the root of your cell's tree. Then grant *system:anyuser* the READ and LOOKUP rights on the ACL.

Type each of the following commands on a single line.

/usr/afs/bin/vos create <*machine name*> <*partition name*> **root.cell -cell** <*cellname*>

/usr/afs/bin/fs mkm /afs/<*cellname*> **root.cell**

/usr/afs/bin/fs sa /afs/<*cellname*> **system:anyuser rl**

Step 3: **(Optional).** You may wish to link your full cell name to a shorter cell name, so that users have less to type on the command line. For example, in Transarc's file tree, */afs/transarc.com* is linked to the shorter */afs/tr*. Do this for your home cell only.

Working in the */afs* directory, link */afs/**cellname*** to an abbreviated name, if desired.

cd /afs

ln -s <*full cellname*> <*short cellname*>

Step 4: Mount `root.cell` with a ReadWrite mount point (you already mounted it with a regular mount point in step 2). This mount point tells the Cache Manager to access only the ReadWrite version of `root.cell`.

By convention, a ReadWrite mount point has a period in front of the directory name, as shown in the instruction (i.e., */afs/**.cellname***).

Create a ReadWrite mount point to `root.cell`.

/usr/afs/bin/fs mkm /afs/.<*cellname*> **root.cell -rw**

Step 5: Define a replication site for both `root.afs` and `root.cell`. Since this is your cell's first machine, the sites will presumably be on this machine. You may wish to place each site on a partition other than the one where the ReadWrite version resides, but that is not necessary. Note that multiple sites on one machine may speed access to a volume, but do not preserve the volume's availability if the file server machine goes down. It is recommended that you also replicate `root.afs` and `root.cell` on the next few additional file server machines you install.

Define replication sites for `root.afs` and `root.cell`.

/usr/afs/bin/vos addsite <*machine name*> <*partition name*> **root.afs**

/usr/afs/bin/vos addsite <*machine name*> <*partition name*> **root.cell**

Step 6: Make sure that `root.afs` and `root.cell` exist and are accessible before you attempt to replicate them. The output from **fs examine** should list their name, volumeID number, quota, and size, and then the size of the partition. If you get an error message instead, do not continue before taking corrective action.

Use **fs examine** to verify that `root.afs` and `root.cell` exist.

/usr/afs/bin/fs examine /afs

/usr/afs/bin/fs examine /afs/<*cellname*>

Step 7: Release replicas of `root.afs` and `root.cell` to the sites you defined in step 5.

Use **vos release** to release replicas of `root.afs` and `root.cell` to the ReadOnly sites.

/usr/afs/bin/vos release root.afs

/usr/afs/bin/vos release root.cell

Step 8: Verify that both volumes are now released. First issue **fs checkvolumes** to force the Cache Manager to notice that you have released ReadOnly versions of the volumes, then issue **fs examine** again. This time the output should mention `root.afs.readonly` and `root.cell.readonly` instead of the ReadWrite versions, since the Cache Manager has a built-in bias to access the ReadOnly version of `root.afs` if it exists.

Verify that the local Cache Manager now sees the ReadOnly versions of `root.afs` and `root.cell`.

/usr/afs/bin/fs checkvolumes

/usr/afs/bin/fs examine /afs

/usr/afs/bin/fs examine /afs/<*cellname*>

2.26. Turning On Authorization Checking

Now turn on authorization checking so that the server processes on this machine will only perform privileged actions for authorized users. You should already have tokens as *admin*, from having issued **klog** in section 2.23. Before turning on authorization checking, use the **tokens** command to verify that you have tokens. If you do not have tokens, or they are expired, reissue **klog**.

Step 1: Verify that you are authenticated as *admin*. The output of the **tokens** command should indicate that there are tokens for *admin*'s AFS UID in your cell. If not, reissue **klog** as shown in section 2.23.

```
# /usr/afs/bin/tokens
```

Step 2: Turn on authorization checking on the file server machine.

Issue **bos setauth** to turn on authorization checking.

```
# /usr/afs/bin/bos setauth <machine name> on -cell <cellname>
```

Step 3: Restart all server processes on the machine, including the BOS Server, to ensure that they establish new connections with one another that obey the new authorization checking requirement. This command can take several seconds to complete. You may see several messages about the BOS Server and other processes restarting.

Use **bos restart** to restart all server process on this machine. The **-bosserver** flag stops and restarts the BOS Server, which then starts up all other processes listed with status flag Run in */usr/afs/local/BosConfig*.

```
# /usr/afs/bin/bos restart <machine name> -bosserver -cell <cellname>
```

2.27. Setting up Volumes to House AFS Binaries

AFS client binaries and configuration files must be available in the subdirectories under */usr/afsws* on each client machine (*afsws* is an acronym for "AFS workstation binaries"). To save disk space, you should create */usr/afsws* on the local disk as a link to the volume that houses the AFS client binaries and configuration files for this system type.

In this section you will create and mount volumes for housing the binaries. The recommended location for mounting the volume is */afs/**cellname**/**sysname**/usr/afsws*, where ***sysname*** is the Transarc system name for this machine's system type (as listed in section 1.3). The instructions in chapter 4 for installing additional client machines of this machine type assume that you have followed the instructions in this section.

As you install client machines of different system types, you will need to create new volumes and directories for each type. Follow the instructions in chapter 4 for installing additional clients.

If you have previously run a version of AFS in your cell, you may already have created volumes for housing AFS binaries for this machine type. If so, the only step you need to perform in this section is step 3 in section 2.27.1 (creating the */usr/afsws* link).

Step 1: Create volumes to store AFS client binaries for this machine's system type. The following example instructions create three volumes called "***sysname***", "***sysname***.usr" and "***sysname***.usr.afsws", enough to permit loading AFS client binaries into */afs/**cellname**/**sysname**/usr/afsws*. Refer to section 1.3 to learn the proper value of ***sysname*** for this machine type.

```
# cd /usr/afs/bin

# vos create <machine name> <partition name> <sysname>

# vos create <machine name> <partition name> <sysname>.usr

# vos create <machine name> <partition name> <sysname>.usr.afsws
```

Step 2: Mount the volumes created in step 1 at the indicated place in the AFS file tree, just below */afs/**cellname***. Because `root.cell` is now replicated, you must make the mount points in its ReadWrite version, by preceding ***cellname*** with a period as shown. You then issue **vos release** to release new replicas of `root.cell`, and **fs checkvolumes** to force the local Cache Manager to access them.

```
# fs  mkmount  /afs/.<cellname>/<sysname>  <sysname>

# fs  mkmount  /afs/.<cellname>/<sysname>/usr  <sysname>.usr

# fs  mkmount  /afs/.<cellname>/<sysname>/usr/afsws  <sysname>.usr.afsws

# vos  release  root.cell

# fs  checkvolumes
```

Step 3: Set the ACL on the mount points created in step 2 to grant *system:anyuser* the READ and LOOKUP rights.

```
Set the ACL on the mount points created in step 2.

# cd  /afs/<cellname>/<sysname>

# fs  sa  -dir  .  ./usr  ./usr/afsws  -acl  system:anyuser  rl
```

Step 4: Set the quota on */afs/**cellname**/**sysname**/usr/afsws* according to the following chart. The values include a safety margin.

Operating system	**Quota in kilobyte blocks**
AOS	25000
AIX	30000
HP-UX for Series 300 and 400	25000
HP-UX for Series 700 and 800	35000
IRIX	45000
NeXT OS	50000
Solaris	30000
SunOS	25000
Ultrix (Vaxen)	25000
Ultrix (DECstations)	45000

```
# fs setquota /afs/<cellname>/<sysname>/usr/afsws  <quota>
```

2.27.1. Loading AFS Binaries into a Volume and Creating a Link to the Local Disk

Now load the AFS binaries from the fifth **tar** set on the AFS Binary Distribution tape into the volume mounted at */afs/**cellname**/**sysname**/usr/afsws*.

If disk space (on the file server machine housing the volume) allows, you should load the complete contents of the */usr/afsws* into AFS. If space does not permit, you should at minimum load the client binaries found under */usr/afsws/root.client*.

Step 1: **If the local machine has a tape drive,** mount the Binary Distribution tape and load the fifth **tar** set into */afs/**cellname**/**sysname**/usr/afsws*.

The appropriate subdirectories are created automatically, and have their ACL set to match that on */afs/**cellname**/**sysname**/usr/afsws* (which at this point grants all rights to *system:administrators* and READ and LOOKUP to *system:anyuser*).

> **Note:** If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.
>
> For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.
>
> # **cd /afs/**<*cellname*>**/**<*sysname*>**/usr/afsws**
>
> # **mt -f /dev/**<*device*> **rewind**
>
> # **mt -f /dev/**<*device*> **fsf 4**
>
> # **tar xvf /dev/**<*device*>

If loading the binaries from a remote machine's */usr/afsws* **directory,** change directory (**cd**) to */afs/**cellname**/**sysname**/usr/afsws* on the local machine and then use **ftp**, NFS, or another network transfer program to copy in the contents of the remote machine's */usr/afsws* directory.

Step 2: You may make AFS software available to users only in accordance with the terms of your AFS License agreement. To prevent access by unauthorized users, you should change the ACL on some of the subdirectories of */afs/**cellname**/**sysname**/usr/afsws*, granting the READ and LOOKUP rights to *system:authuser* instead of *system:anyuser*. This way, only users who are authenticated in your cell can access AFS binaries. The ACL on the *bin* subdirectory must continue to grant the READ and LOOKUP rights to *system:anyuser*, because unauthenticated users must be able to access the **klog** binary stored there.

To be sure that unauthorized users are not accessing AFS software, you should periodically check that the ACL on these directories is set properly.

To limit access to AFS binaries to users authenticated in your cell, issue the following commands. The ACL on the *bin* subdirectory must continue to grant the READ and LOOKUP rights to *system:anyuser*.

cd /afs/<*cellname*>/<*sysname*>**/usr/afsws**

fs sa -dir ./* -acl system:authuser rl system:anyuser none

fs sa -dir bin -acl system:anyuser rl

Step 3: Create a symbolic link from */usr/afsws* (a local directory) to */afs/**cellname**/@sys/usr/afsws*. You could also substitute the machine's Transarc system name for *@sys* (make the link to */afs/**cellname**/**sysname**/usr/afsws*). The advantage of using *@sys* is that it automatically adjusts in case you upgrade this machine to a different system type.

ln -s /afs/<*cellname*>**/@sys/usr/afsws /usr/afsws**

You should include */usr/afsws/bin* and */usr/afsws/etc* in the PATH variable for each user account, so that users can issue commands from the AFS suites (such as **fs**).

2.28. Storing System Binaries in AFS

In addition to AFS binaries, you may wish to store other system binaries in AFS volumes, such as the standard UNIX programs found under */etc*, */bin*, and */lib*. This kind of central storage eliminates the need to keep copies of common programs and files on every machine's local disk.

This section summarizes the recommended scheme for storing system binaries in AFS. A more extended discussion appears in chapter 2 of the *AFS System Administrator's Guide*. No instructions appear here, since you may not wish to create the necessary volumes at this time. If you do wish to create them, use the instructions in section 2.27 (which are for AFS-specific binaries) as a template.

You cannot link all system binaries to AFS directories; some files must remain on the local disk for use during times when AFS is inaccessible (bootup and outages). They include:

- a basic text editor, network commands, etc.
- boot sequence files executed before AFS is accessible (before **afsd** is executed), such as startup command files, mount commands, and configuration files
- files needed by dynamic kernel loaders. For example, the **dkload** process requires */bin/nm*, */bin/as*, */bin/ld* to be on the local disk.

2.28.1. Setting the ACL on System Binary Volumes

It is recommended that you restrict access to most system binaries by granting the READ and LOOKUP rights to *system:authuser* instead of *system:anyuser*. This limits access to users who are authenticated in your cell. However, the ACL on binaries that users must access while unauthenticated should grant READ and LOOKUP to *system:anyuser*.

2.28.2. Volume and Directory Naming Scheme

It is recommended that you create a volume called ***sysname*** for each supported system type you use as an AFS machine, where ***sysname*** is a system name listed in section 1.3. Mount each such volume at */afs/**cellname**/**sysname***; you already did so for this machine's system type, in section 2.27. You must use the value for ***sysname*** listed in section 1.3 if you wish to use the *@sys* variable in pathnames.

Then create a volume for each system distribution directory (such as */bin*, */etc*, */lib*, */usr*) and mount it under */afs/**cellname**/**sysname***. As an example, you would name the volume containing */bin* files as ***sysname***`.bin` and mount it at */afs/**cellname**/**sysname**/bin*.

You can name volumes in any way you wish, and mount them at other locations than those suggested here. However, the following scheme is recommended because it:

- clearly identifies the volume contents
- makes it easy to back up related volumes together because the AFS Backup System uses a wild card notation to take advantage of volume name prefixes common to multiple volumes
- makes it easy to track related volumes, keeping them together on the same file server machine if desired
- correlates volume name and mount point name

The following is an suggested scheme for naming volumes and directories to house system binaries. (Certain of these volumes—such as ***sysname***.`usr.afsws`—already exist as a result of previous instructions in this chapter.)

Volume Name	**Mount Point**
sysname	*/afs/**cellname**/**sysname***
sysname.`bin`	*/afs/**cellname**/**sysname**/bin*
sysname.`etc`	*/afs/**cellname**/**sysname**/etc*
sysname.`usr`	*/afs/**cellname**/**sysname**/usr*
sysname.`usr.afsws`	*/afs/**cellname**/**sysname**/usr/afsws*
sysname.`usr.bin`	*/afs/**cellname**/**sysname**/usr/bin*
sysname.`usr.etc`	*/afs/**cellname**/**sysname**/usr/etc*
sysname.`usr.inc`	*/afs/**cellname**/**sysname**/usr/include*
sysname.`usr.lib`	*/afs/**cellname**/**sysname**/usr/lib*
sysname.`usr.loc`	*/afs/**cellname**/**sysname**/usr/local*
sysname.`usr.man`	*/afs/**cellname**/**sysname**/usr/man*
sysname.`usr.sys`	*/afs/**cellname**/**sysname**/usr/sys*

2.29. Enabling Access to Transarc and Other Cells

At this point, the client version of *CellServDB* (in */usr/vice/etc*) on this machine lists the local cell and all of the foreign cells from the sample *CellServDB* that you decided to retain. The cells are still not accessible, however, because their `root.cell` volumes are not yet mounted in your file tree. You create the mount points in this section.

Repeat these instructions for each foreign cell that you wish to enable your cell's client machines to access. You do not have to perform these instructions now if you do not wish to, but the foreign cell(s) will not be accessible until you do. Keep in mind that a client machine's *CellServDB* file must also list the database server machines for any cell it is to access. Instructions for making new cells accessible in future appear in section *in.first.newcell* in chapter 15 of the *AFS System Administrator's Guide*.

Step 1: Verify that */usr/vice/etc/CellServDB* includes an entry for the cell.

Step 2: Mount the cell's `root.cell` volume at */afs/**cellname*** in your file tree. The following series of steps is necessary because `root.afs` is replicated. You must alter its ReadWrite version by temporarily mounting it in a writable directory (*/afs/.**cellname***) and making your changes. Then release the changed volume to the ReadOnly sites, where client machines normally access it. The **fs checkvolumes** command forces the local Cache Manager to notice the release of the new replica. The final **ls** command is optional, but allows you to verify that the new cell's mount point is visible in your file tree. The output should show the directories at the top level of the new cell's tree.

Substitute your cell's name for *cellname* in the **cd** commands.

cd /afs/.<*cellname*>

fs mkmount temp root.afs

cd temp

Repeat the following **fs mkmount** command for each foreign cell you wish to mount at this time; issue the subsequent commands only once.

fs mkmount <*foreign cell*> **root.cell -c** <*foreign cell*>

cd /afs/.<*cellname*>

fs rmmount temp

vos release root.afs

fs checkvolumes

ls /afs/<*foreign cell*>

2.30. Enabling Access to New Cells in the Future

The instructions in this section enable your cell's client machines to access a foreign cell that does not already appear in *CellServDB*, without having to reboot. You can follow these instructions any time you wish to add access to a new foreign cell. You do not have to follow them now if there are no new foreign cells you want to make accessible from this machine.

Transarc Corporation maintains the file */afs/transarc.com/service/etc/CellServDB.export* as a list of all cells that have agreed to advertise their database server machines. Every effort is made to keep this file updated with the most current information available. You may wish to check the file periodically for the existence of new cells.

Transarc also maintains a *CellServDB* of test and private cells, which it does not make available to other cells. Even if you do not wish your cell's database server machines to be advertised to everyone, please register your cell with Transarc for inclusion in this file.

There are three things you can do to facilitate maintenance of *CellServDB*:

- You may want to store in your local AFS tree a central copy of the *CellServDB* appropriate for your cell's client machines, perhaps as */afs/**cellname**/common/etc/CellServDB*, rather than relying on the global version exported by Transarc. Whenever you install a new client, you can copy this file over to it.
- Transarc requests that you list your cell's database server machines in a file called */afs/**cellname**/service/etc/CellServDB.local*, and make it readable by foreign users. This is similar to Transarc's global */afs/transarc.com/service/etc/CellServDB.export*, but allows foreign cells to get information about your cell's database server machines directly from you. Include only the information that can be exposed to other cells. Private or test cells you administer and do not wish to make accessible to foreign users should be listed in another version of *CellServDB* not readable by foreign users (perhaps as */afs/**cellname**/common/etc/CellServDB.private*).
- It is important for Transarc to have up-to-date information about AFS cells. When you finish setting up your cell, or whenever you change your cell's database server machines, please inform your AFS Product Support Representative of the names and Internet addresses of all of your database server machines. By default, this information is made available to other cells.

Step 1: If you maintain a central file such as */afs/**cellname**/common/etc/CellServDB*, update it to include the new cell. It may be easiest to copy it from */afs/transarc.com/service/etc/CellServDB.export*. If you type it yourself, be sure to maintain proper file and entry format, as described in section 2.21.

Step 2: Add the foreign cell's entry to the local */usr/vice/etc/CellServDB* file. If you maintain a central copy of *CellServDB*, you can use AFS to copy it from there. If not, it may be easiest to copy the cell's entry directly from */afs/transarc.com/service/etc/CellServDB.export*. If you type it yourself, be sure to maintain proper file and entry format, as described in section 2.21.

Step 3: Add the cell to the in-kernel list that the Cache Manager consults for cell information (recall that it only looks at */usr/vice/etc/CellServDB* as **afsd** runs at reboot, to transfer the contents into the kernel record).

Issue **fs newcell** for the cell you want to add to the Cache Manager's in-kernel list. Provide the complete Internet host name of each database server machine.

fs newcell <*cellname*> <*dbserver1*> **[**<*dbserver2*>**] [**<*dbserver3*>**]**

For example, to add the list of database server machines for the Transarc Corporation cell, you would issue on a single line

fs newcell transarc.com bigbird.transarc.com ernie.transarc.com oscar.transarc.com

Step 4: Mount the cell's `root.cell` volume at */afs/**cellname*** in your file tree. Because `root.afs` is replicated, you must first temporarily mount the ReadWrite version of `root.afs` in a writable directory (such as your cell's */afs/.**cellname*** directory). Make your changes and then release new replica to the ReadOnly sites. This is the only way to make the change visible to client machines when they access the ReadOnly version of `root.afs`, as they normally do.

Note: You only need to mount a cell's `root.cell` volume once, not on each client machine.

The **fs checkvolumes** command forces the local Cache Manager to notice the release of new replica.

Substitute your cell's name for *cellname* in the **cd** commands.

cd /afs/.<*cellname*>

fs mkmount temp root.afs

cd temp

Repeat the following **fs mkmount** command for each foreign cell you wish to mount at this time; issue the subsequent commands only once.

fs mkmount <*foreign cell*> **root.cell -c** <*foreign cell*> **-root**

cd /afs/.<*cellname*>

fs rmmount temp

vos release root.afs

fs checkvolumes

Step 5: Verify that the new cell's mount point is visible in your file tree. The output should show the directories at the top level of the new cell's tree.

ls /afs/<*cellname*>

Step 6: Repeat steps 2 and 3 on every client machine in your cell that needs to access the new cell's file tree.

2.31. Improving Your Cell's Security

This section discusses some measures you should implement immediately to improve your cell's security. For additional information, refer to chapter 2 of the *AFS System Administrator's Guide*, especially the sections titled "Setting Up File Server Machines" and "Security and Authorization in AFS".

2.31.1. Controlling root Access

As in standard UNIX administration, you should not allow unauthorized users to learn the "root" password. Although "root" does not have special access to files in AFS, it does have

- the privilege necessary to issue certain commands that affect workstation performance (these are in the **fs** suite)
- the ability to turn off authorization checking on a file server machine, allowing anyone to perform otherwise privileged actions

2.31.2. Controlling System Administrator Access

You should also limit the number of people who have system administrator privilege. Follow these guidelines:

- Create generic administrative accounts like *admin*. Only authenticate under these identities when performing administrative tasks, and destroy the administrative tokens immediately after finishing the task (either by issuing **unlog** or issuing **klog** to adopt your regular identity).
- Set a short ticket lifetime for system administrator accounts (for example, 20 minutes), using the **-lifetime** argument to the **kas setfields** command (see the *AFS Command Reference Manual*). You should not make the lifetime this short on the account used for system backups.
- Restrict the number of people with administrative privilege, especially the number of members in *system:administrators*.

  Members of *system:administrators* by default have all ACL rights on all directories in the file tree, and therefore must be trusted not to examine private files. The members of this group can also issue all **pts** commands, create users and groups, and change volume quotas.
- Limit the use of system administrator accounts on public workstations.

2.31.3. Protecting Sensitive AFS Directories

The subdirectories of */usr/afs* on a file server machine contain sensitive files (such as *UserList* and *KeyFile* in the *etc* subdirectory). Users should not see or be able to write to these files, since they could potentially figure out to use them to gain access to many administrator privileges.

The first time it initializes on a file server machine, the BOS Server creates several of these directories and files. It sets their owner to "root" and sets their mode bits so that no one but the owner can write them; in some cases, it also disables reading.

At each subsequent restart, the BOS Server checks that the protections on the following files still comply with the chart, and that "root" owns each file or directory. A question mark in place of a right indicates that the BOS Server does not care what it is.

/usr/afs	`drwxr?xr-x`	*/usr/afs/etc/KeyFile*	`-rw????---`
/usr/afs/bin	`drwxr?xr-x`	*/usr/afs/local*	`drwx???---`
/usr/afs/db	`drwx???---`	*/usr/afs/etc/UserList*	`-rw?????--`
/usr/afs/etc	`drwxr?xr-x`	*/usr/afs/logs*	`drwxr?xr-x`

In case of incorrect mode bits or ownership, the BOS Server adds a warning to */usr/afs/logs/BosLog*; the warning also appears in the output of the **bos status** command when the **-long** flag is used. The BOS Server does not reset the mode bits, which allows you to set them in alternate ways if that is appropriate.

To correct the mode bits to comply with the chart, you could issue one or more of the following commands.

```
# chmod 755 /usr/afs
# chmod 755 /usr/afs/bin
# chmod 700 /usr/afs/db
# chmod 755 /usr/afs/etc
# chmod 600 /usr/afs/etc/KeyFile
# chmod 600 /usr/afs/etc/UserList
# chmod 700 /usr/afs/local
# chmod 755 /usr/afs/logs
```

You should never need to modify files in these directories directly. Use the appropriate AFS commands (such as the **bos addkey** and **bos removekey** commands to add and remove server encryption keys in */usr/afs/etc/KeyFile*).

2.32. Replacing login

Note: If you plan to remove the client functionality from this file server machine, skip this section and proceed to section 2.34.

Transarc provides a version of **login** that both authenticates the issuer with AFS and logs him or her into the local UNIX file system. It is strongly recommended that you replace standard **login** with the AFS-authenticating version so that your cell's users automatically receive PAG-based tokens when they log in. If you do not replace the standard **login**, then your users must use the two-step login procedure (log into the local UNIX file system followed by **pagsh** and **klog** to authenticate with AFS). For more details, see the section titled "Login and Authentication in AFS" in chapter 2 of the *AFS System Administrator's Guide*.

If you are using Kerberos authentication rather that AFS's protocols, you must install AFS's **login.krb** instead of regular AFS **login**. Contact your AFS Product Support Representative for further details.

You can tell you are running AFS **login** if the following banner appears after you provide your password:

```
AFS 3.3 (R) login
```

To install AFS **login**, follow the instructions below that are appropriate for your system type:

- For AIX 3.2 systems, see section 2.32.1.
- For IRIX systems, see section 2.32.2.
- For NeXT systems using the Login Window, see section 2.32.3.
- For all other system types, see section 2.32.4.

2.32.1. Replacing login for AIX 3.2

Follow the instructions in this section to replace **login** on AIX 3.2 systems (system type *rs_aix32*).

For this system type, Transarc supplies both **login.noafs**, which is invoked when AFS is not running on the machine, and **login.afs**, which is invoked when AFS is running; these files are found in */usr/afsws/bin*, assuming you have followed the instructions in section 2.27.1. Note that standard AIX 3.2 **login** is normally installed as */usr/sbin/login*, with links to */etc/tsm*, */etc/getty* and */bin/login*. You will install the replacement AFS binaries into the */bin* directory.

Step 1: Replace the link to standard **login** in */bin* with **login.noafs**.

```
# mv  /bin/login  /bin/login.orig

# cp  /usr/afsws/bin/login.noafs  /bin/login
```

Step 2: Replace the links from */etc/getty* and */etc/tsm* to standard **login** with links to */bin/login*.

```
# mv  /etc/getty  /etc/getty.orig

# mv  /etc/tsm  /etc/tsm.orig

# ln -s  /bin/login  /etc/getty

# ln -s  /bin/login  /etc/tsm
```

Step 3: Install **login.afs** into */bin*, and create a symbolic link to */etc/afsok*.

```
# cp  /usr/afsws/bin/login.afs  /bin/login.afs

# ln -s  /bin/login.afs  /etc/afsok
```

2.32.2. Replacing login for IRIX Systems

Silicon Graphics, Inc. has modified the IRIX **login** program to operate in an AFS environment in the same manner as the AFS version. It is not necessary to replace standard **login** on IRIX systems. Transarc does not provide a modified **login** program for these systems.

2.32.3. Replacing the Login Window binary for NeXT OS Systems

Follow the instructions in this section to replace the binary for the "Login Window" on NeXT OS systems (system type *next_mach30*) with a version that performs AFS **login**.

Note that on this system type the `AFS 3.3 (R) login` banner does not appear.

Step 1: Replace the standard **loginwindow** binary with the AFS version.

```
# cd  /usr/lib/NextStep/loginwindow.app

# mv  loginwindow  loginwindow.NextStep

# cp  /usr/afsws/bin/loginwindow  loginwindow
```

2.32.4. Replacing login for Other System Types

Follow the instructions in this section to replace **login** on other system types.

For these system types, the replacement AFS **login** binary resides in */usr/afsws/bin*, assuming you have followed the instructions in section 2.27.1. You install it as */bin/login*.

Step 1: Replace standard **login** with AFS **login**.

```
# mv  /bin/login  /bin/login.orig

# cp  /usr/afsws/bin/login  /bin/login
```

2.33. Altering File System Clean-Up Scripts

Note: If you plan to remove the client functionality from this file server machine, skip this section and proceed to section 2.34.

NeXT systems and many Sun systems are distributed with a crontab file that contains a command for removing unneeded files from the file system (it usually begins with the **find(1)** command). On NeXT systems, the standard location for this file is */usr/adm/daily*; on Sun systems, it is standardly */usr/spool/cron/crontabs/root*.

Once this machine is an AFS client, you must modify the pathname specification in this cron command to exclude */afs*. Otherwise, the command will traverse the entire portion of the AFS tree accessible from this machine, which includes every cell whose database server machines appear in the machine's kernel list (derived from */usr/vice/etc/CellServDB*). The traversal could take many hours.

Use care in altering the pathname specification, so that you do not accidentally exclude directories that you wish to be searched. The following may be suitable alterations, but are suggestions only—you must verify that they are appropriate for your system.

The first possible alteration requires that you list all file system directories to be searched.

On SunOS and NeXT OS systems, use:

```
find / /usr /<other partitions> -xdev remainder of existing command
```

On Solaris systems, use:

```
find / /usr /<other partitions> -mount remainder of existing command
```

The second possibility excludes any directories whose names begin with "*a*" or a non-alphabetic character.

```
find /[A-Zb-z]*  remainder of existing command
```

Note that you should not use the following, because it still searches under */afs*, looking for a subdirectory of type "4.2".

```
find / -fstype 4.2      /* do not use */
```

2.34. Removing Client Functionality

Follow the instructions in this section only if you do not wish this machine to remain an AFS client. Removing client functionality will make the machine unable to access files in AFS.

Step 1: Remove the call to **afsd** from the machine's initialization file (*/etc/rc* or equivalent).

Step 2: Remove the */usr/vice* directory and everything under it.

Step 3: Reboot the machine, and login again as "root." The following sequence is appropriate on most machine types.

```
# sync

# /etc/reboot

login: root
Password:
```

3. Installing Additional Servers

This chapter explains how to

- install a file server machine of the same AFS system type (one that uses the same AFS binaries) as an existing file server machine. See section 3.1.
- install the first file server machine of a new system type (one that uses different binaries than an existing file server machine). See section 3.2.
- add database server functionality to an existing file server machine. See section 3.3.
- remove database server functionality from a machine. See section 3.4.

Assumptions

These instructions require that:

- you have already installed your cell’s first file server machine by following the instructions in chapter 2
- you are working at the console of the machine you are installing, logged into the local UNIX file system as "root"
- you have already installed a standard, vendor-supplied UNIX kernel on the machine you are installing

3.1. Installing a File Server Machine of an Existing System Type

If the machine you wish to install as a file server machine is the same AFS system type as an existing file server machine, you can copy the binaries it needs from the existing machine rather than from the Binary Distribution tape. If you also loaded the contents of the fifth **tar** set (the */usr/afsws* directory) into AFS, as recommended in section 2.27.1, you can set up this machine as an AFS client and copy the needed files from the AFS directory (the recommended name for which is */afs/**cellname**/**sysname**/root.server/usr/afs*.) To install client functionality on a file server machine, see sections 2.18 through 2.23.

The instructions in this section assume that the first machine of this system type was designated as the binary distribution machine for its type. Therefore, the following steps instruct you to install the client portion of the Update Server.

These instructions are brief; refer to the corresponding steps in chapter 2 for more detailed information. If you are copying files from a remote file server machine, the instructions assume they reside in */usr/afsws* on that machine.

3.1.1. Procedure Overview

To install a file server machine of an existing type, you will

1. copy needed binaries and files onto this machine's local disk
2. replace the standard kernel with one that includes AFS modifications (either using a dynamic kernel loader or with a previously built kernel)
3. configure partitions to house AFS volumes
4. replace the standard **fsck** with the AFS version of **fsck**
5. start the Basic OverSeer (BOS) Server
6. start client portion(s) of the Update Server
7. start the controller for the NTP Daemon
8. start the process that binds the File Server, Volume Server and Salvager
9. continue to section 3.3 if this machine will be a database server machine

3.1.2. Instructions

Step 1: Create the following directories as appropriate.

```
# mkdir /usr/afs

# mkdir /usr/afs/etc

# mkdir /usr/afs/bin

# mkdir /usr/afs/bin/sgiload (if you will use ml)

# mkdir /usr/afs/bin/modload (if you will use modload)

# mkdir /usr/afs/bin/dkload (if you will use any other dynamic kernel loader)
```

Step 2: Using **ftp**, NFS, or another file transfer method, copy everything from the system control machine's */usr/afs/etc* directory to the local */usr/afs/etc*.

Similarly, copy the following files from an existing file server machine of the same system type to this machine's local disk.

- Copy everything from */usr/afs/bin*.
- If this machine runs IRIX, copy */usr/afs/bin/sgiload*.

  If this machine runs Solaris, or SunOS and will use **modload**, copy */usr/afs/bin/modload*.

  If this machine will use any other dynamic kernel loader, copy */usr/afs/bin/dkload*.

  If this machine will use a previously built AFS-modified kernel (instead of a dynamic kernel loader), copy */vmunix* or equivalent.

Step 3: Incorporate AFS modifications into the machine's kernel. Follow the instructions for the appropriate dynamic kernel loader, or for copying over a previously built kernel.

- If you use **dkload**, perform the following. This loader is available on the system types listed in section 2.3.2.

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.13.1.
  2. Run **dkload** manually.

     Note: If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.a* for *libafs.a*. Either use the **mv** command to replace *libafs.a* with *libafs.nonfs.a* in the */usr/afs/bin/dkload* directory (before issuing these commands), or make the substitution on the command line. If you wish to use the "stripped" version of the AFS library file, substitute '*libafss*' for '*libafs*' in the same fashion.

     # **cd /usr/afs/bin/dkload**

     # **ranlib libafs.a** on SunOS and Ultrix systems only

     # **ranlib libcommon.a** on SunOS and Ultrix systems only

     # **./dkload libafs.a**

  3. Modify the machine's initialization file (*/etc/rc* or equivalent) to invoke **dkload**, by copying in the contents of */usr/afs/bin/dkload/rc.dkload.server*. Place the commands after the commands that mount the file systems (but before **afsd** is invoked, if this machine is also a client). Remember to substitute *libafs.nonfs.a* for *libafs.a*, if the machine's kernel does not support NFS. If you wish to use the "stripped" version of the AFS library file, substitute *libafss.a* for *libafs.a* or *libafss.nonfs.a* for *libafs.nonfs.a*.

- To use the kernel extension facility on AIX 3.2 systems:

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.14.1.
  2. Invoke **cfgexport** and **cfgafs**.

     Note: If this machine will act as an NFS/AFS Translator machine, you must make a substitution in this step. For details, consult the section entitled "Setting Up an NFS/AFS Translator Machine" in the NFS/AFS Translator Supplement to the *AFS System Administrator's Guide*.

     If the machine is to act as an NFS/AFS Translator, remember to make the necessary substitution. If the machine will not run NFS at all, substitute *export.ext.nonfs* for *export.ext*.

     # **cd /usr/afs/bin/dkload**

     # **./cfgexport -a export.ext**

     # **./cfgafs -a afs.ext**

3. IBM delivers several function-specific initialization files for AIX 3.2 systems, rather than the single file used on some other systems. If you want the kernel extension facility to run each time the machine reboots, copy the contents of */usr/afs/bin/dkload/rc.dkload.server* into these initialization files such that the indicated commands appear in the order shown here (you will add some of the commands in later steps). Note particularly that if the machine will remain a client as well as a file server, then you should not invoke **nfsd** at the same place in the initialization files as the commands that load *nfs.ext*; **nfsd** must follow the call to **afsd**.

   – command(s) mounting the file systems

   – command(s) loading *nfs.ext*, if the machine will run NFS, which an NFS/AFS Translator machine must (but only for AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)

   – the contents of *rc.dkload.server*, to invoke the kernel extension facility. If the machine will act as an NFS/AFS Translator machine, be sure to make the same substitution as you made in the preceding step.

   – **bosserver** (you will be instructed to add this command in step 10)

   – **afsd** (if this machine will remain a client after you finish this installation; see step 11)

   – **nfsd** (if the machine is to be an NFS server)

- To use **ml** on IRIX systems:

1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.15.1.

   Also remove the *.dl* extension from the AFS library file that is appropriate for this machine. If you do not know which processor the machine has, issue IRIX's **hinv** command and check the line in the output that begins "CPU."

   Remove the *.dl* extension from one of the AFS library files by issuing the (one) appropriate command.

   # **cd /usr/afs/bin/sgiload**

   For R3000 processor with NFS support:

   # **cp libafs.R3000.o.dl libafs.R3000.o**

   For R3000 processor without NFS support:

   # **cp libafs.R3000.nonfs.o.dl libafs.R3000.nonfs.o**

   For R4000 processor with NFS support:

   # **cp libafs.R4000.o.dl libafs.R4000.o**

   For R4000 processor, without NFS support:

   # **cp libafs.R4000.nonfs.o.dl libafs.R4000.nonfs.o**

2. Invoke **ml**, replacing *library file* with the name of the library file appropriate for this machine (the one you selected by removing its *.dl* extension).

```
# ml ld -v -j /usr/afs/bin/sgiload/<library file> -p afs_ -a afs
```

3. Copy the initialization script that Transarc provides for IRIX systems as */usr/afs/bin/sgiload/afs.rc* to the */etc/init.d* directory, make sure it is executable, and link it to the two locations where **ml** expects to find it. Then use the **chkconfig** command to activate configuration files that tell **ml** that this machine will run AFS and serve as a file server machine.

```
# cd /etc/init.d

# cp /usr/afs/bin/afs.rc afs.rc

# chmod 555 afs.rc

# ln -s afs.rc /etc/rc0.d/K35afs.rc

# ln -s afs.rc /etc/rc2.d/S35afs.rc

# cd /etc/config

# chkconfig -f afsml on

# chkconfig -f afsserver on
```

- To use **kl_util** on NeXT OS systems:

1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.16.1.

   Also copy the *afs_reloc* file from its distribution directory to the location where **kl_util** expects to find it.

```
# mkdir /usr/lib/kern_loader/AFS

# cp /usr/afs/bin/dkload/afs_reloc /usr/lib/kern_loader/AFS
```

2. Invoke **kl_util**.

```
# /usr/etc/kl_util -a /usr/lib/kern_loader/AFS/afs_reloc
```

3. Add the following line to */etc/kern_loader.conf* (any ordering with other lines in the file is acceptable):

```
/usr/lib/kern_loader/AFS/afs_reloc
```

- To use **modload** on Solaris systems:

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.17.1.

     Also create the file */kernel/fs/afs* as a copy of the appropriate AFS library file.

     # **cd /usr/afs/bin/modload**

     # **cp libafs.o /kernel/fs/afs** if the machine supports NFS

     # **cp libafs.nonfs.o /kernel/fs/afs** if the machine does not support NFS

     Then create an entry for AFS in the */etc/name_to_sysnum* file, to allow the kernel to make AFS system calls.

     In the file */etc/name_to_sysnum*, create an "afs" entry in slot 105 (the slot just before the "nfs" entry), so that the file looks like:

     ```
     reexit          1
     fork            2
      .              .
      .              .
      .              .
     sigpending      99
     setcontext      100
     statvfs         103
     fstatvfs        104
     afs             105
     nfs             106
     ```

  2. Invoke **modload**.

     # **/usr/sbin/modload /kernel/fs/afs**

  3. Copy the initialization script that Transarc provides for Solaris systems as */usr/afs/bin/modload/afs.rc* to the */etc/init.d* directory, make sure it is executable, and link it to the two locations where **modload** expects to find it.

     # **cd /etc/init.d**

     # **cp /usr/afs/bin/modload/afs.rc afs.rc**

     # **chmod 555 afs.rc**

     # **ln -s afs.rc /etc/rc3.d/S14afs**

     # **ln -s afs.rc /etc/rc2.d/K66afs**

- If you use **modload** on SunOS systems:

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.18.1.
  2. Invoke **modload**.

     ```
     # cd /usr/afs/bin/modload

     # /usr/etc/modload ./libafs.o if the machine supports NFS

     # /usr/etc/modload ./libafs.nonfs.o if the machine does not support NFS
     ```

  3. Modify the machine's initialization file (*/etc/rc* or equivalent) to invoke **modload**, by copying in the contents of */usr/afs/bin/modload/rc.modload.server*. Place the commands after the commands that mount the file systems (but before **afsd** is invoked, if this machine is also a client). Remember to substitute *libafs.nonfs.o* for *libafs.o* if the machine's kernel does not support NFS.

- If you use an AFS-modified kernel already built for this system type:

  1. Move the existing kernel to a save-file.

     ```
     # mv /vmunix /vmunix_save
     ```

  2. Copy the AFS-modified kernel from an existing file server of the same system type to */vmunix* on the local disk, using **ftp**, NFS, or AFS.
  3. Reboot the machine after warning users of the reboot. The following commands are appropriate on most system types.

     ```
     # sync

     # /etc/reboot
     ```

Step 4: Configure partitions to house AFS volumes.

1. Create a */vicep**x*** directory for each partition that will house AFS volumes. The example instruction creates three directories.

   Important note for HP-UX systems: Disk striping is supported for the *hp700_ux90* system type, but not for *hp300_ux90* (so only one partition per disk is allowed on that system type). The *hp800_ux90* system type uses logical volumes rather than disk striping.

   ```
   # mkdir /vicepa

   # mkdir /vicepb

   # mkdir /vicepc

     and so on
   ```

2. Take the action necessary to associate each */vicep**x*** directory just created with a partition on the machine that will house AFS volumes. For many system types, this requires adding a line to the machine's "file systems registry" file (*/etc/fstab* or equivalent.)

   Unless otherwise indicated, use the syntax shown for each line you add (to */etc/fstab*, unless otherwise noted). Examples appear following the instruction box.

   For AIX 2.2.1, use the **devices** program to edit */etc/filesystems*. Consult the operating system documentation for syntax.

   For AIX 3.2, use the **SMIT** program to create a journalled file system. Consult the operating system documentation for syntax.

   For AOS: `/dev/`<*disk*> `/vicep`<*x*> `ufs rw 1 2`

   For HP-UX on Series 300, 400 and 700 machines, place the following in */etc/checklist*:
   `/dev/dsk/`<*disk*> `/vicep`<*x*> `hfs defaults 0 2`

   For HP-UX on Series 800 machines using logical volumes, use the **SAM** program to create a file system. Consult the operating system documentation for syntax. (If not using logical volumes, use the same instruction as for Series 300, 400 and 700 machines.)

   For IRIX: `/dev/vicep`<*x*> `/vicep`<*x*> `efs rw,raw=/dev/rvicep`<*x*> `0 0`

   For NeXT OS: `/dev/`<*disk*> `/vicep`<*x*> `4.3 rw,noquota 0 2`

   For Solaris, place the following in */etc/vfstab*:
   `/dev/dsk/`<*disk*> `/dev/rdsk/`<*disk*> `/vicep`<*x*> `ufs` <*boot order*> `yes`

   For SunOS: `/dev/`<*disk*> `/vicep`<*x*> `4.2 rw 1 2`

   For Ultrix: `/dev/`<*disk*>`:/vicep`<*x*>`:rw:1:2:ufs::`

Examples of file registry entries (in */etc/fstab* except for HP-UX, which uses */etc/checklist*, and Solaris, which uses */etc/vfstab*):

```
AOS:    /dev/hd1g /vicepa ufs rw 1 2
HP-UX:  /dev/dsk/1s0  /vicepa hfs  defaults  0 2
IRIX:   /dev/vicepa /vicepa efs rw,raw=/dev/rvicepa 0 0
NeXT    /dev/sd0b /vicepa 4.3 rw,noquota 0 2
Solaris: /dev/dsk/c0t6d0s1 /dev/rdsk/c0t6d0s1 /vicepa ufs 3 yes
SunOS:  /dev/sd0g /vicepa 4.2 rw 1 2
Ultrix:  /dev/rz4a:/vicepa:rw:1:2:ufs::
```

3. Create a file system on each partition. The commands shown should be suitable for the indicated system types, but consult your operating system documentation for more information.

   Create a file system on each partition, using the command appropriate to your system type. **Repeat this command for each partition**.

   For AIX 2.2.1: # **minidisks**

   For AIX 3.2, your creation of a journalled file system (with **SMIT** in the previous step) makes this step unnecessary.

   For AOS: # **minidisk**

   For HP-UX on Series 300, 400 and 700 machines, use the **newfs** or **makefs** command; consult the operating system documentation for syntax.

   For HP-UX on Series 800 machines, your use of **SAM** to create a file system in the previous step makes this step unnecessary. If you did not use **SAM**, use **newfs** or **makefs** as directed for Series 300, 400 and 700 machines.

   For IRIX: # **mkfs /dev/rvicep**<*x*>

   For NeXT OS, use the **disk** command or **BuildDisk** program; consult the operating system documentation for syntax.

   For Solaris: # **newfs -v /dev/rdsk**<*xxxxxxxx*>

   For SunOS: # **newfs -v /dev/rsd**<*xx*>

   For Ultrix: # **newfs -v /dev/rhd**<*xx*> <*disk_type*>

4. Mount the partition(s), using the command appropriate to your operating system. For many BSD-based systems, the **mount -a** command is appropriate and mounts all partitions at once. For many System V-based systems, the **mountall** command is appropriate instead.

   For NeXT OS systems, no "mount all" command is appropriate. You must either reboot the machine or mount each */vicep**x*** partition by hand, because the **mount** command consults NetInfo rather than */etc/fstab* once the machine initializes. If you reboot, remember to log in again as "root."

For AIX 3.2 systems, it is recommended that you also add the following line to */etc/vfs* at this point:

```
afs     4       none        none
```

If you do not add this line, you will receive an error message from the **mount** command when you use it to list the mounted file systems (but note that the **mount** command is working properly even if you receive the error message).

Step 5: Replace the standard version of **fsck** with a version that handles AFS volumes properly.

- For AIX 3.2 systems:

```
# cd  /sbin/helpers

# mv  v3fshelper  v3fshelper.noafs

# cp  /usr/afs/bin/v3fshelper  v3fshelper
```

- For DECstations:

```
# mv  /bin/fsck  /bin/fsck.old

# cp  /usr/afs/bin/vfsck  /bin/vfsck

# rm  /etc/fsck

# ln  -s  /bin/vfsck  /etc/fsck
```

- For IRIX systems, no action is necessary. Silicon Graphics, Inc. has modified IRIX **fsck** to handle AFS volumes properly.
- For NeXT OS systems:

```
# mv   /usr/etc/fsck  /usr/etc/fsck.old

# cp  /usr/afs/bin/vfsck  /usr/etc/vfsck

# ln  -s  /usr/etc/vfsck  /usr/etc/fsck
```

- For Solaris systems:

```
# mv  /usr/lib/fs/ufs/fsck  /usr/lib/fs/ufs/fsck.old

# cp  /usr/afs/bin/vfsck  /usr/lib/fs/ufs/vfsck

# ln  -s  /usr/lib/fs/ufs/vfsck  /usr/lib/fs/ufs/fsck
```

- For SunOS systems:

```
# mv   /usr/etc/fsck  /usr/etc/fsck.old
# cp  /usr/afs/bin/vfsck  /usr/etc/vfsck
# rm  /etc/fsck
# ln  -s  /usr/etc/vfsck  /etc/fsck
# ln  -s  /usr/etc/vfsck  /usr/etc/fsck
```

- For any other system type:

```
# mv /etc/fsck /etc/fsck.old
# cp /usr/afs/bin/vfsck /etc/vfsck
# ln -s /etc/vfsck /etc/fsck
```

Step 6: Start the BOS Server (*bosserver*), using the **-noauth** flag to prevent the AFS processes from performing authorization checking. Remember that this is a grave compromise of security; finish the remaining instructions in this section in an uninterrupted pass.

```
# /usr/afs/bin/bosserver -noauth &
```

Step 7: Create one or two instances of the client portion of the Update Server (*upclient*), depending on whether you are using the international or United States edition of AFS:

- With both editions, create *upclientbin* to accept the AFS binaries stored in */usr/afs/bin*, from the *upserver* process on the binary distribution machine for this machine's system type. You should already have followed the instructions in chapter 2 or section 3.2 to install the first machine of this system type as the binary distribution machine.
- If using the United States edition, create *upclientetc* to accept the common configuration files stored in */usr/afs/etc* from the *upserver* process (server portion of the Update Server) on the system control machine. You should already have followed the instructions in chapter 2 to install your cell's first file server machine as the system control machine.

In the United States edition of AFS, instances of the *upclient* process by default requests that *upserver* encrypt files before transferring them across the network. This is appropriate for */usr/afs/etc*, because its contents are sensitive. It is not appropriate for */usr/afs/bin*, because they are not sensitive and encrypting binaries takes a long time. You will use the **-clear** flag on */usr/afs/bin* so that *upclientbin* does not request encryption for it.

In the international edition of AFS, the *upserver* cannot access the routines it needs to encrypt user-level data. The *upclient* process therefore must request that *upserver* send files in unencrypted form. This is acceptable for */usr/afs/bin*, but not */usr/afs/etc*. In the international edition of AFS, do not use the Update Server to update */usr/afs/etc*; you will have to update the contents of */usr/afs/etc* by hand on each file server machine.

By default, the Update Server performs updates every 300 seconds (five minutes). Use the **-t** argument to specify an different number of seconds.

Type each command on a single line. Substitute this machine's name for *machine name* and this cell's name for *cellname*, in this and all remaining commands in this section.

Issue this command with both the United States and international editions:

```
# bos create <machine name> upclientbin simple "/usr/afs/bin/upclient
                <binary distribution machine> [-t <time>] -clear /usr/afs/bin"
                -cell <cellname>
```

Issue this command only with the United States edition:

```
# bos create <machine name> upclientetc simple "/usr/afs/bin/upclient
                <system control machine> [-t <time>] /usr/afs/etc"
                -cell <cellname>
```

Step 8: Start the *runntp* process, which controls the Network Time Protocol Daemon (NTPD), after verifying that the **ntpd** and **ntpdc** binaries exist in */usr/afs/bin*. NTPD keeps the internal clock on this machine synchronized with the clock on other file server machines in the cell.

Type each command on a single line.

```
# ls /usr/afs/bin

# bos create <machine name> runntp simple /usr/afs/bin/runntp
                -cell <cell name>
```

Step 9: Start the *fs* process, which binds together the File Server, Volume Server and Salvager.

Type the following command on a single line.

```
# bos create <machine name>  fs  fs  /usr/afs/bin/fileserver
                /usr/afs/bin/volserver  /usr/afs/bin/salvager  -cell <cellname>
```

Step 10: Verify that the machine's initialization file invokes **bosserver**, so that the BOS Server starts automatically at each file server reboot.

If the machine does not run IRIX or Solaris, add the following lines to */etc/rc* or equivalent, after the lines that configure the network, mount all file systems,and invoke a kernel dynamic loader.

```
if [ -f /usr/afs/bin/bosserver ]; then
   echo 'Starting bosserver' > /dev/console
      /usr/afs/bin/bosserver &
fi
```

If the machine runs IRIX or Solaris, no action is necessary. The *afs.rc* script includes tests that result in automatic BOS Server start up if appropriate (you installed *afs.rc* as part of dynamic kernel loading, in section 2.3.4 for IRIX or section 2.3.6 for Solaris).

Step 11: The commands in several of the following steps (13, 14, and 17) must be issued on an AFS client machine. You must either

- make this machine a client before continuing, by following the instructions in sections 2.18 through 2.23.
- issue the remaining commands on an existing AFS client machine (of any system type), either at the console or over a remote connection.

Since this is an additional file server machine of an existing type, the AFS binaries for this system type presumably already reside in AFS, as recommended in section 2.27. If they do not, before continuing you should follow the instructions in that section too. If you decide to work on another client machine, remember to perform the final step in section 2.27—linking the local directory */usr/afsws* to the appropriate location in the AFS file tree—on the new file server machine itself.

In addition, the following should now appear in the machine's initialization file(s) in the indicated order:

- command(s) mounting the file systems
- command(s) loading *nfs.ext* (but only for AIX systems running NFS and AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)
- dynamic kernel loader command(s), unless AFS was built into the kernel
- **bosserver**
- **afsd** (if the machine will remain a client after you complete this installation)
- **nfsd** (if the machine is to be an NFS server)

Step 12: If this machine will remain an AFS client after you complete this installation, replace standard **login** with the AFS version.

- For AIX 3.2 systems:

```
# mv  /bin/login  /bin/login.orig
# cp  /usr/afsws/bin/login.noafs  /bin/login
# mv  /etc/getty  /etc/getty.orig
# mv  /etc/tsm  /etc/tsm.orig
# ln -s  /bin/login  /etc/getty
# ln -s  /bin/login  /etc/tsm
# cp  /usr/afsws/bin/login.afs  /bin/login.afs
# ln -s  /bin/login.afs  /etc/afsok
```

- For IRIX systems, no action is necessary. Silicon Graphics, Inc. has modified IRIX **login** to operate the same as AFS **login** when the machine's kernel includes AFS.
- For NeXT OS systems:

```
# cd  /usr/lib/NextStep/loginwindow.app
# mv  loginwindow  loginwindow.NextStep
# cp  /usr/afsws/bin/loginwindow  loginwindow
```

- For any other system type:

```
# mv  /bin/login  /bin/login.orig
# cp  /usr/afsws/bin/login  /bin/login
```

Step 13: Authenticate as *admin* (or whatever name you assigned the administrative account in section 2.10).

```
# klog admin
Password:
```

Step 14: Turn on authorization checking.

Substitute the new file server machine's name for *machine name*.

/usr/afs/bin/bos setauth <*machine name*> **on -cell** <*cellname*>

Step 15: Verify that */usr/afs* and its subdirectories on the new file server machine meet the ownership and mode bit requirements outlined in section 2.31.3. If necessary, use the **chmod** commands shown there to correct the mode bits.

Step 16: If you made this machine an AFS client in step 11 and want it to remain one, then follow the instructions in section 2.33 to alter file system clean-up files on NeXT and Sun systems. If you wish to remove the client functionality from this machine, see section 2.34.

Step 17: Restart the BOS Server, which will start all processes you created with **bos create** commands in this section.

Substitute the new file server machine's name for *machine name*.

/usr/afs/bin/bos restart <*machine name*> **-bosserver -cell** <*cellname*>

Step 18: If you want this machine be a database server machine, proceed to section 3.3.

3.2. Installing a File Server Machine of a New System Type

The instructions in this section are similar to those for installing the first file server machine in a cell. As the first file server machine of a new system type, this machine will be designated as the ***binary distribution machine*** for its type.

3.2.1. Procedure Overview

To install the first file server machine of a new type, you will

1. load the file server binaries and the files needed for incorporating an AFS-modified kernel onto the local disk
2. incorporate AFS modifications into the machine's kernel (using either a dynamic kernel loader or the kernel build procedure)
3. configure partitions to house AFS volumes
4. replace the standard **fsck** with the AFS version of **fsck**
5. start the Basic OverSeer (BOS) Server
6. start the server portion of the Update Server
7. start client portion(s) of the Update Server
8. start the controller for the NTP Daemon
9. start the process that binds the File Server, Volume Server and Salvager
10. continue to section 3.3 if this machine will be a database server machine

The following instructions are brief; refer to the procedures in chapter 2 for more information. If you are copying files from a remote machine, it is assumed they reside in */usr/afsws* on that machine.

3.2.2. Instructions

Step 1: Create the following directories as appropriate.

```
# mkdir /usr/afs

# mkdir /usr/afs/etc

# mkdir /usr/afs/bin/sgiload (if you will use ml)

# mkdir /usr/afs/bin/modload (if you will use modload)

# mkdir /usr/afs/bin/dkload (if you will use any other dynamic kernel loader)

# mkdir /usr/afs/sys (if you will build a new kernel)
```

Step 2: Using **ftp**, NFS, or another file transfer method, copy everything from the system control machine's */usr/afs/etc* directory to the local */usr/afs/etc*.

Step 3: Load the appropriate files onto the local disk, using the appropriate method:

If the machine you are installing is attached to a tape drive, load the contents of **tar** sets indicated:

- Load the third **tar** set into */usr/afs* (instructions appear in section 2.4). The *bin* subdirectory is created automatically; it contains file server process binaries.
- If using a dynamic kernel loader, load the first **tar** set into */usr/afs/bin* (instructions appear in section 2.3.1). On IRIX systems, the *sgiload* directory is created automatically. On Solaris and SunOS systems, the *modload* directory is created automatically. On all other systems for which a loader is available, the *dkload* directory is created automatically.
- If building AFS into a new kernel for this system type, load the second **tar** set into */usr/afs/sys* (instructions appear in section 5.2).

If the machine you are installing is not attached to a tape drive, the AFS Binary Distribution files must reside in */usr/afsws* on a remote machine, so that you can copy them from there onto the local disk using a network transfer program. For detailed instructions on loading the files into */usr/afsws* in the first place, see section 1.4.2.2.

- Copy the file server process binaries and configuration files from */usr/afsws/root.server/usr/afs/bin* into the local */usr/afs/bin* directory.
- If this machine runs IRIX, copy the contents of */usr/afsws/root.client/usr/vice/etc/sgiload* into the local */usr/afs/bin/sgiload* directory.

  If this machine runs Solaris, or SunOS and will use **modload**, copy the contents of */usr/afsws/root.client/usr/vice/etc/modload* into the local */usr/afs/bin/modload* directory.

  If this machine will use any other dynamic kernel loader, copy the contents of

/usr/afsws/root.client/usr/vice/etc/dkload into the local */usr/afs/bin/dkload* directory.

If you will build AFS into a new kernel for this system type, copy the contents of */usr/afsws/root.client/bin* into the local */usr/afs/sys* directory.

Step 4: Incorporate AFS modifications into the machine's kernel. Follow the instructions for the appropriate dynamic kernel loader, or for copying over a previously built kernel.

- If you use **dkload**, perform the following. This loader is available on the system types listed in section 2.3.2.

1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.13.1.
2. Run **dkload** manually.

> **Note:** If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.a* for *libafs.a*. Either use the **mv** command to replace *libafs.a* with *libafs.nonfs.a* in the */usr/afs/bin/dkload* directory (before issuing these commands), or make the substitution on the command line. If you wish to use the "stripped" version of the AFS library file, substitute '*libafss*' for '*libafs*' in the same fashion.
>
> # **cd /usr/afs/bin/dkload**
>
> # **ranlib libafs.a** on SunOS and Ultrix systems only
>
> # **ranlib libcommon.a** on SunOS and Ultrix systems only
>
> # **./dkload libafs.a**

3. Modify the machine's initialization file (*/etc/rc* or equivalent) to invoke **dkload**, by copying in the contents of */usr/afs/bin/dkload/rc.dkload.server*. Place the commands after the commands that mount the file systems (but before **afsd** is invoked, if this machine is also a client). Remember to substitute *libafs.nonfs.a* for *libafs.a*, if the machine's kernel does not support NFS. If you wish to use the "stripped" version of the AFS library file, substitute *libafss.a* for *libafs.a* or *libafss.nonfs.a* for *libafs.nonfs.a*.

- To use the kernel extension facility on AIX 3.2 systems:

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.14.1.

  2. Invoke **cfgexport** and **cfgafs**.

     Note: If this machine will act as an NFS/AFS Translator machine, you must make a substitution in this step. For details, consult the section entitled "Setting Up an NFS/AFS Translator Machine" in the NFS/AFS Translator Supplement to the *AFS System Administrator's Guide*.

     If the machine is to act as an NFS/AFS Translator, remember to make the necessary substitution. If the machine will not run NFS at all, substitute *export.ext.nonfs* for *export.ext*.

     # **cd /usr/afs/bin/dkload**

     # **./cfgexport -a export.ext**

     # **./cfgafs -a afs.ext**

  3. IBM delivers several function-specific initialization files for AIX 3.2 systems, rather than the single file used on some other systems. If you want the kernel extension facility to run each time the machine reboots, copy the contents of */usr/afs/bin/dkload/rc.dkload.server* into these initialization files such that the indicated commands appear in the order shown here (you will add some of the commands in later steps). Note particularly that if the machine will remain a client as well as a file server, then you should not invoke **nfsd** at the same place in the initialization files as the commands that load *nfs.ext*; **nfsd** must follow the call to **afsd**.

     – command(s) mounting the file systems

     – command(s) loading *nfs.ext*, if the machine will run NFS, which an NFS/AFS Translator machine must (but only for AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)

     – the contents of *rc.dkload.server*, to invoke the kernel extension facility. If the machine will act as an NFS/AFS Translator machine, be sure to make the same substitution as you made in the preceding step.

     – **bosserver** (you will be instructed to add this command in step 10)

     – **afsd** (if this machine will remain a client after you finish this installation; see step 11)

     – **nfsd** (if the machine is to be an NFS server)

- To use **ml** on IRIX systems:

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.15.1.

     Also remove the *.dl* extension from the AFS library file that is appropriate for this machine. If you do not know which processor the machine has, issue IRIX's **hinv** command and check the line in the output that begins "CPU."

     Remove the *.dl* extension from one of the AFS library files by issuing the (one) appropriate command.

     # **cd /usr/afs/bin/sgiload**

     For R3000 processor with NFS support:

     # **cp libafs.R3000.o.dl libafs.R3000.o**

     For R3000 processor without NFS support:

     # **cp libafs.R3000.nonfs.o.dl libafs.R3000.nonfs.o**

     For R4000 processor with NFS support:

     # **cp libafs.R4000.o.dl libafs.R4000.o**

     For R4000 processor, without NFS support:

     # **cp libafs.R4000.nonfs.o.dl libafs.R4000.nonfs.o**

  2. Invoke **ml**, replacing *library file* with the name of the library file appropriate for this machine (the one you selected by removing its *.dl* extension).

     # **ml ld -v -j /usr/afs/bin/sgiload/**<*library file*> **-p afs_ -a afs**

3. Copy the initialization script that Transarc provides for IRIX systems as */usr/afs/bin/sgiload/afs.rc* to the */etc/init.d* directory, make sure it is executable, and link it to the two locations where **ml** expects to find it. Then use the **chkconfig** command to activate configuration files that tell **ml** that this machine will run AFS and serve as a file server machine.

```
# cd  /etc/init.d

# cp  /usr/afs/bin/afs.rc  afs.rc

# chmod  555  afs.rc

# ln -s afs.rc  /etc/rc0.d/K35afs.rc

# ln -s afs.rc  /etc/rc2.d/S35afs.rc

# cd /etc/config

# chkconfig  -f  afsml  on

# chkconfig  -f  afsserver  on
```

- To use **kl_util** on NeXT OS systems:

1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.16.1.

   Also copy the *afs_reloc* file from its distribution directory to the location where **kl_util** expects to find it.

```
# mkdir  /usr/lib/kern_loader/AFS

# cp  /usr/afs/bin/dkload/afs_reloc  /usr/lib/kern_loader/AFS
```

2. Invoke **kl_util**.

```
# /usr/etc/kl_util  -a  /usr/lib/kern_loader/AFS/afs_reloc
```

3. Add the following line to */etc/kern_loader.conf* (any ordering with other lines in the file is acceptable):

```
/usr/lib/kern_loader/AFS/afs_reloc
```

- To use **modload** on Solaris systems:

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.17.1.

     Also create the file */kernel/fs/afs* as a copy of the appropriate AFS library file.

     # **cd /usr/afs/bin/modload**

     # **cp libafs.o /kernel/fs/afs** if the machine supports NFS

     # **cp libafs.nonfs.o /kernel/fs/afs** if the machine does not support NFS

     Then create an entry for AFS in the */etc/name_to_sysnum* file, to allow the kernel to make AFS system calls.

     In the file */etc/name_to_sysnum*, create an "afs" entry in slot 105 (the slot just before the "nfs" entry), so that the file looks like:

     ```
     reexit          1
     fork            2
      .              .
      .              .
      .              .
     sigpending      99
     setcontext      100
     statvfs         103
     fstatvfs        104
     afs             105
     nfs             106
     ```

  2. Invoke **modload**.

     # **/usr/sbin/modload /kernel/fs/afs**

  3. Copy the initialization script that Transarc provides for Solaris systems as */usr/afs/bin/modload/afs.rc* to the */etc/init.d* directory, make sure it is executable, and link it to the two locations where **modload** expects to find it.

     # **cd /etc/init.d**

     # **cp /usr/afs/bin/modload/afs.rc afs.rc**

     # **chmod 555 afs.rc**

     # **ln -s afs.rc /etc/rc3.d/S14afs**

     # **ln -s afs.rc /etc/rc2.d/K66afs**

- If you use **modload** on SunOS systems:

  1. Verify that the machine's local disk houses the needed files and directories, as listed in section 5.18.1.
  2. Invoke **modload**.

     ```
     # cd /usr/afs/bin/modload

     # /usr/etc/modload ./libafs.o if the machine supports NFS

     # /usr/etc/modload ./libafs.nonfs.o if the machine does not support NFS
     ```

  3. Modify the machine's initialization file (*/etc/rc* or equivalent) to invoke **modload**, by copying in the contents of */usr/afs/bin/modload/rc.modload.server*. Place the commands after the commands that mount the file systems (but before **afsd** is invoked, if this machine is also a client). Remember to substitute *libafs.nonfs.o* for *libafs.o* if the machine's kernel does not support NFS.

- If you use an AFS-modified kernel already built for this system type:

  1. Move the existing kernel to a save-file.

     ```
     # mv /vmunix /vmunix_save
     ```

  2. Copy the AFS-modified kernel from an existing file server of the same system type to */vmunix* on the local disk, using **ftp**, NFS, or AFS.
  3. Reboot the machine after warning users of the reboot. The following commands are appropriate on most system types.

     ```
     # sync

     # /etc/reboot
     ```

Step 5: Configure partitions to house AFS volumes.

1. Create a *vicepx* directory for each partition that will house AFS volumes. The example instruction creates three directories.

   Important note for HP-UX systems: Disk striping is supported for the *hp700_ux90* system type, but not for *hp300_ux90* (so only one partition per disk is allowed on that system type). The *hp800_ux90* system type uses logical volumes rather than disk striping.

   ```
   # mkdir /vicepa

   # mkdir /vicepb

   # mkdir /vicepc

     and so on
   ```

2. Take the action necessary to associate each */vicep**x*** directory just created with a partition on the machine that will house AFS volumes. For many system types, this requires adding a line to the machine's "file systems registry" file (*/etc/fstab* or equivalent.)

Unless otherwise indicated, use the syntax shown for each line you add (to */etc/fstab*, unless otherwise noted). Examples appear following the instruction box.

For AIX 2.2.1, use the **devices** program to edit */etc/filesystems*. Consult the operating system documentation for syntax.

For AIX 3.2, use the **SMIT** program to create a journalled file system. Consult the operating system documentation for syntax.

For AOS: `/dev/`<*disk*> `/vicep`<*x*> `ufs rw 1 2`

For HP-UX on Series 300, 400 and 700 machines, place the following in */etc/checklist*:
`/dev/dsk/`<*disk*> `/vicep`<*x*> `hfs defaults 0 2`

For HP-UX on Series 800 machines using logical volumes, use the **SAM** program to create a file system. Consult the operating system documentation for syntax. (If not using logical volumes, use the same instruction as for Series 300, 400 and 700 machines.)

For IRIX: `/dev/vicep`<*x*> `/vicep`<*x*> `efs rw,raw=/dev/rvicep`<*x*> `0 0`

For NeXT OS: `/dev/`<*disk*> `/vicep`<*x*> `4.3 rw,noquota 0 2`

For Solaris, place the following in */etc/vfstab*:
`/dev/dsk/`<*disk*> `/dev/rdsk/`<*disk*> `/vicep`<*x*> `ufs` <*boot order*> `yes`

For SunOS: `/dev/`<*disk*> `/vicep`<*x*> `4.2 rw 1 2`

For Ultrix: `/dev/`<*disk*>`:/vicep`<*x*>`:rw:1:2:ufs::`

Examples of file registry entries (in */etc/fstab* except for HP-UX, which uses */etc/checklist*, and Solaris, which uses */etc/vfstab*):

```
AOS:    /dev/hd1g /vicepa ufs rw 1 2
HP-UX:  /dev/dsk/1s0  /vicepa hfs  defaults  0 2
IRIX:   /dev/vicepa /vicepa efs rw,raw=/dev/rvicepa 0 0
NeXT:   /dev/sd0b /vicepa 4.3 rw,noquota 0 2
Solaris: /dev/dsk/c0t6d0s1 /dev/rdsk/c0t6d0s1 /vicepa ufs 3 yes
SunOS:  /dev/sd0g /vicepa 4.2 rw 1 2
Ultrix:  /dev/rz4a:/vicepa:rw:1:2:ufs::
```

3. Create a file system on each partition. The commands shown should be suitable for the indicated system types, but consult your operating system documentation for more information.

   Create a file system on each partition, using the command appropriate to your system type. **Repeat this command for each partition**.

   For AIX 2.2.1: # **minidisks**

   For AIX 3.2, your creation of a journalled file system (with **SMIT** in the previous step) makes this step unnecessary.

   For AOS: # **minidisk**

   For HP-UX on Series 300, 400 and 700 machines, use the **newfs** or **makefs** command; consult the operating system documentation for syntax.

   For HP-UX on Series 800 machines, your use of **SAM** to create a file system in the previous step makes this step unnecessary. If you did not use **SAM**, use **newfs** or **makefs** as directed for Series 300, 400 and 700 machines.

   For IRIX: # **mkfs /dev/rvicep**<*x*>

   For NeXT OS, use the **disk** command or **BuildDisk** program; consult the operating system documentation for syntax.

   For Solaris: # **newfs -v /dev/rdsk**<*xxxxxxxx*>

   For SunOS: # **newfs -v /dev/rsd**<*xx*>

   For Ultrix: # **newfs -v /dev/rhd**<*xx*> <*disk_type*>

4. Mount the partition(s), using the command appropriate to your operating system. For many BSD-based systems, the **mount -a** command is appropriate and mounts all partitions at once. For many System V-based systems, the **mountall** command is appropriate instead.

   For NeXT OS systems, no "mount all" command is appropriate. You must either reboot the machine or mount each */vicep**x*** partition by hand, because the **mount** command consults NetInfo rather than */etc/fstab* once the machine initializes. If you reboot, remember to log in again as "root."

   For AIX 3.2 systems, it is recommended that you also add the following line to */etc/vfs* at this point:

   ```
   afs     4     none     none
   ```

   If you do not add this line, you will receive an error message from the **mount** command when you use it to list the mounted file systems (but note that the **mount** command is working properly even if you receive the error message).

Step 6: Replace the standard version of **fsck** with a version that handles AFS volumes properly.

- For AIX 3.2 systems:

```
# cd  /sbin/helpers

# mv  v3fshelper  v3fshelper.noafs

# cp  /usr/afs/bin/v3fshelper  v3fshelper
```

- For DECstations:

```
# mv  /bin/fsck  /bin/fsck.old

# cp  /usr/afs/bin/vfsck  /bin/vfsck

# rm  /etc/fsck

# ln  -s  /bin/vfsck  /etc/fsck
```

- For IRIX systems, no action is necessary. Silicon Graphics, Inc. has modified IRIX **fsck** to handle AFS volumes properly.
- For NeXT OS systems:

```
# mv   /usr/etc/fsck  /usr/etc/fsck.old

# cp  /usr/afs/bin/vfsck  /usr/etc/vfsck

# ln  -s  /usr/etc/vfsck  /usr/etc/fsck
```

- For Solaris systems:

```
# mv  /usr/lib/fs/ufs/fsck  /usr/lib/fs/ufs/fsck.old

# cp  /usr/afs/bin/vfsck  /usr/lib/fs/ufs/vfsck

# ln  -s  /usr/lib/fs/ufs/vfsck  /usr/lib/fs/ufs/fsck
```

- For Solaris systems:

```
# mv  /usr/lib/fs/ufs/fsck  /usr/lib/fs/ufs/fsck.old

# cp  /usr/afs/bin/vfsck  /usr/lib/fs/ufs/vfsck

# ln  -s  /usr/lib/fs/ufs/vfsck  /usr/lib/fs/ufs/fsck
```

- For SunOS systems:

```
# mv   /usr/etc/fsck  /usr/etc/fsck.old
# cp  /usr/afs/bin/vfsck  /usr/etc/vfsck
# rm  /etc/fsck
# ln  -s  /usr/etc/vfsck  /etc/fsck
# ln  -s  /usr/etc/vfsck  /usr/etc/fsck
```

- For any other system type:

```
# mv /etc/fsck /etc/fsck.old
# cp /usr/afs/bin/vfsck /etc/vfsck
# ln -s /etc/vfsck /etc/fsck
```

Step 7: Start the BOS Server (*bosserver*), using the **-noauth** flag to prevent the AFS processes from performing authorization checking. Remember that this is a grave compromise of security; finish the remaining instructions in this section in an uninterrupted pass.

```
# /usr/afs/bin/bosserver -noauth &
```

Step 8: Create an instance of the server portion of the Update Server (*upserver*), to distribute its copy of the file server process binaries in */usr/afs/bin* to the other file server machines of this system type that you may install in future. Creating this process makes this machine the binary distribution machine for its type.

Type the following on a single line.

```
# bos create <machine name> upserver simple
                "/usr/afs/bin/upserver -clear /usr/afs/bin" -cell <cellname>
```

Step 9: If using the United States edition of AFS, create an instance of the client portion of the Update Server (*upclient*), to accept updates of the common configuration files stored in */usr/afs/etc* from the server portion of the Update Server on the system control machine. The recommended name for this process is *upclientetc*. You should already have followed the instructions in chapter 2 to install your cell's first file server machine as the system control machine.

In the United States edition of AFS, *upclient* requests that *upserver* transfer files in encrypted form. This is appropriate for */usr/afs/etc*.

Do not issue this command with the international edition of AFS, because encryption of user-level data is not possible. The Update Server cannot encrypt */usr/afs/etc*, and

its contents are too sensitive to cross the network unencrypted. You will have to update the contents of */usr/afs/etc* by hand on each file server machine.

By default, the Update Server performs updates every 300 seconds (five minutes). Use the **-t** argument to specify an different number of seconds.

If using the United States edition of AFS, type the following on a single line. Substitute this machine's name for *machine name* and this cell's name for *cell name*, in this and all remaining commands in this section.

bos create <*machine name*> **upclientetc simple "/usr/afs/bin/upclient** <*system control machine*> **[-t** <*time*>**] /usr/afs/etc"** **-cell** <*cellname*>

Step 10: Start the *runntp* process, which controls the Network Time Protocol Daemon (NTPD), after verifying that the **ntpd** and **ntpdc** binaries exist in */usr/afs/bin*. NTPD keeps the internal clock on this machine synchronized with the clock on other file server machines in the cell.

Type each command on a single line.

ls /usr/afs/bin

bos create <*machine name*> **runntp simple /usr/afs/bin/runntp** **-cell** <*cellname*>

Step 11: Start the *fs* process, which binds together the File Server, Volume Server and Salvager.

Type the following command on a single line.

bos create <*machine name*> **fs fs /usr/afs/bin/fileserver** **/usr/afs/bin/volserver /usr/afs/bin/salvager -cell** <*cell name*>

Step 12: Verify that the machine's initialization file invokes **bosserver**, so that the BOS Server starts automatically at each file server reboot.

If the machine does not run IRIX or Solaris, add the following lines to */etc/rc* or equivalent, after the lines that configure the network, mount all file systems,and invoke a kernel dynamic loader.

```
if [ -f /usr/afs/bin/bosserver ]; then
   echo 'Starting bosserver' > /dev/console
      /usr/afs/bin/bosserver &
fi
```

If the machine runs IRIX or Solaris, no action is necessary. The *afs.rc* script includes tests that result in automatic BOS Server start up if appropriate (you installed *afs.rc* as part of dynamic kernel loading, in section 2.3.4 for IRIX or section 2.3.6 for Solaris).

Step 13: The commands in several of the following steps (14, 16, and 19) must be issued on an AFS client machine. You must either

- make this machine a client before continuing, by following the instructions in sections 2.18 through 2.23.
- issue the remaining commands on an existing AFS client machine (of any system type), either at the console or over a remote connection.

Since this is the first file server machine of its type, you should also follow the instructions in section 2.27 to copy the AFS binaries for this system type into an AFS volume. Whether or not you perform the steps on another client machine, remember to perform the final step in section 2.27—linking the local directory */usr/afsws* to the appropriate location in the AFS file tree—on the new file server machine itself.

You may also wish to create AFS volumes to house UNIX system binaries for the new system type, as discussed in section 2.28.

In addition, the following should now appear in the machine's initialization file(s) in the indicated order:

- command(s) mounting the file systems
- command(s) loading *nfs.ext* (but only for AIX systems running NFS and AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)
- dynamic kernel loader command(s), unless AFS was built into the kernel
- **bosserver**
- **afsd** (if the machine will remain a client after you complete this installation)
- **nfsd** (if the machine is to be an NFS server)

Step 14: Authenticate as *admin* (or whatever name you assigned the administrative account in section 2.10).

```
# klog admin
Password:
```

Step 15: If this machine will remain an AFS client after you complete this installation, replace standard **login** with the AFS version.

- For AIX 3.2 systems:

```
# mv  /bin/login  /bin/login.orig
# cp  /usr/afsws/bin/login.noafs  /bin/login
# mv  /etc/getty  /etc/getty.orig
# mv  /etc/tsm  /etc/tsm.orig
# ln -s  /bin/login  /etc/getty
# ln -s  /bin/login  /etc/tsm
# cp  /usr/afsws/bin/login.afs  /bin/login.afs
# ln -s  /bin/login.afs  /etc/afsok
```

- For IRIX systems, no action is necessary. Silicon Graphics, Inc. has modified IRIX **login** to operate the same as AFS **login** when the machine's kernel includes AFS.
- For NeXT OS systems:

```
# cd  /usr/lib/NextStep/loginwindow.app
# mv  loginwindow  loginwindow.NextStep
# cp  /usr/afsws/bin/loginwindow  loginwindow
```

- For any other system type:

```
# mv  /bin/login  /bin/login.orig
# cp  /usr/afsws/bin/login  /bin/login
```

Step 16: Turn on authorization checking.

```
Substitute the new file server machine's name for machine name.
# /usr/afs/bin/bos setauth <machine name> on  -cell <cell name>
```

Step 17: Verify that */usr/afs* and its subdirectories on the new file server machine meet the ownership and mode bit requirements outlined in section 2.31.3. If necessary, use the **chmod** commands shown there to correct the mode bits.

Step 18: If you made this machine an AFS client in step 13 and want it to remain one, then follow the instructions in section 2.33 to alter file system clean-up files on NeXT and Sun systems. If you wish to remove the client functionality from this machine, see section 2.34.

Step 19: Restart the BOS Server, which will start all processes you created with **bos create** commands in this section.

Substitute the new file server machine's name for *machine name*.

/usr/afs/bin/bos restart <*machine name*> **-bosserver -cell** <*cell name*>

Step 20: If you want this server be a database server machine, proceed to section 3.3.

3.3. Installing Database Server Functionality on an Existing File Server Machine

Read the following information before you install a database server machine:

- Database server machines are unique because they

  – run the Authentication Server, Protection Server, and Volume Location Server. They also run the Backup Server if the cell uses the AFS Backup System, as is assumed in these instructions.

  – must appear in the *CellServDB* file of every machine in the cell (and of client machines in foreign cells, if they are to access files in this cell)

- It is recommended, but not required, that you configure your database server machines as AFS file server machines. If you choose not to configure a database server machine as a file server machine, then the kernel does not have to incorporate AFS modifications, but the local disk must include most of the files and directories found under */usr/afs* on a file server machine.

  The instructions in this section assume that the machine on which you are installing database server functionality is already a file server machine. Contact your AFS Product Support Representative to learn how to install database server functionality on a non-file server machine.

- During the installation of database server functionality, you must restart all of the database server machines to force the election of a new Ubik coordinator (sync site) for each database server process. AFS access is impossible throughout your cell during the election, which usually takes less than 5 minutes. You may want to schedule database server installation for times when activity is light. (Chapter 2 of the *AFS System Administrator's Guide* discusses Ubik's election procedures in some detail.)

- Updating the in-kernel list of database server machines on each of your cell's client machines is generally the most time- and labor-intensive part of installing a new database server machine, but it is arguably the most crucial for correct functioning in your cell. Incorrect knowledge of your cell's database server machines can prevent your users from authenticating, accessing files and issuing **kas**, **pts**, and **vos** commands.

  You update a client's in-kernel list either by changing *CellServDB* and rebooting, or issuing **fs newcell**; see chapter 15 of the *AFS System Administrator's Guide* for instructions.

  The point at which you need to update your clients' knowledge of database server machines depends on which of the database server machines has the lowest IP address:

  – If the new database server machine has a lower IP address than any the existing database server machines, then you should update every client before you restart the database server processes. If you do not, users may become unable to update (write to) any of the AFS databases. This is because the machine with the lowest IP address is elected Ubik coordinator (sync site) under normal circumstances, and database writes are possible only at the sync site's copy. Clients that you have not updated will not be able to contact the new sync site. (Be aware that if clients contact the new database server machine before it is actually in service, they will experience a timeout before contacting another database server machine. This is a minor, and temporary, problem compared to being unable to write to the database.)

– If the new database server machine does not have the lowest IP address of any database server machine, then it is better to update clients after restarting the database server processes. Clients will not start using the new database server machine until you update their in-kernel list, but that does not cause timeouts or update problems (because the new machine is not likely to become the sync site).

The following instructions indicate the appropriate place to update your clients in either case.

3.3.1. Procedure Overview

To install a database server machine, you will

1. install the **bos** suite of commands locally, as a precaution
2. add the new machine to */usr/afs/etc/CellServDB* on existing file server machines
3. update the *CellServDB* and/or in-kernel list on every client machine
4. start the Authentication Server
5. start the Protection Server
6. start the Volume Location Server
7. start the Backup Server
8. restart the database server processes on every database server machine
9. inform Transarc Corporation of the new database server machine

3.3.2. Instructions

Step 1: As a precaution, make sure that the **bos** command suite binaries are available on the local disk. Having a local copy of **bos** commands is necessary in case of an error during installation, since the following procedures make the file system inaccessible.

- If you are working on a file server machine, the **bos** command binary already resides in */usr/afs/bin*. No action is necessary.
- If you are working on a client workstation, the **bos** binary usually resides in */usr/afsws/bin*, which is normally a symbolic link into AFS. If so, copy the **bos** binary to */tmp*.

```
# cp  /usr/afsws/bin/bos  /tmp
```

Step 2: Add the new database server machine to the */usr/afs/etc/CellServDB* file on existing file server machines.

If using the United States edition of AFS, issue the following command to update *CellServDB* on your cell's system control machine. Its server portion of the Update Server distributes the updated file to all other file server machines.

If using the international edition of AFS, you should not be using the Update Server to distribute files from */usr/afs/etc*. Instead, update *CellServDB* on each file server machine individually.

In the **bos addhost** command, provide the full Internet-style machine name (including cell extension) for both *machine name to update* and *new db-server machine*.

If using the United States edition of AFS, substitute the system control machine's name for *machine name to update*.

If using the international edition of AFS, repeat the **bos addhost** command once for each server machine in your cell, substituting each one's name for *machine name to update* in turn.

```
# cd /usr/afs/bin

# bos addhost <machine name to update> <new db-server machine>
```

If using the United States edition, wait several minutes to allow the Update Server to distribute the new *CellServDB* to all other AFS file server machines in the cell. By default this takes a maximum of five minutes. If using the international edition, attempt to issue all of the **bos addhost** commands within five minutes.

Step 3: Verify that the new database server machine appears in *CellServDB* on all file server machines.

Note: Repeat this command on every file server machine.

bos listhosts <*machine name*>

Step 4: Add the new database server machine to your cell's central update source for the client machine version of *CellServDB*, if there is one. The standard location for the central file is */afs/**cellname**/common/etc/CellServDB*.

If you are willing to advertise your cell to other cells, add the new database server machine to the file that foreign cells can consult to learn about your database server machines. Transarc recommends */afs/**cellname**/service/etc/CellServDB.local* as the standard location.

Step 5: If this machine's IP address is lower than any existing database server machine's, then update every client machine's */usr/vice/etc/CellServDB* file and in-kernel list to include this machine. (If this machine's IP address is not the lowest, it is acceptable to wait until step 11.)

There are several ways to update *CellServDB* on client machines, as detailed in chapter 15 of the *AFS System Administrator's Guide*. You might, for instance, copy over the central update source (which you updated in step 4), with or without the **package** program. To update the machine's in-kernel list, you can either reboot the machine after changing *CellServDB*, or issue the **fs newcell** command.

Step 6: Start the Authentication Server, *kaserver*.

bos create <*machine name*> **kaserver simple /usr/afs/bin/kaserver**

Step 7: Start the Protection Server, *ptserver*.

bos create <*machine name*> **ptserver simple /usr/afs/bin/ptserver**

Step 8: Start the Volume Location (VL) Server, *vlserver*.

bos create <*machine name*> **vlserver simple /usr/afs/bin/vlserver**

Step 9: Start the Backup Server, *buserver*. The chapter entitled "Backing Up the System" in the *AFS System Administrator's Guide* details the other instructions you must perform before actually using the Backup System.

bos create <*machine name*> **buserver simple /usr/afs/bin/buserver**

Step 10: Restart the Authentication, Protection and Volume Location Servers on every database server machine in the cell, including the new server. This forces an election of a new Ubik coordinator site for each of the processes; the new machine votes in the election and is considered as a potential new coordinator.

A cell-wide service outage is possible during the election of a new Ubik coordinator for the VL Server, but should last less than five minutes. Such an outage is particularly likely if you are installing your cell's second database server machine. Messages tracing the progress of the election should appear on the console.

> **Note:** Repeat this command, substituting each of your cell's database server machines for *database server machine* in quick succession.
>
> # **bos restart** <*database server machine*> **kaserver ptserver vlserver buserver**

If an error occurs (perhaps because you accidentally omit one of the processes on one machine), you should restart all server processes on the database server machines, either by

- issuing **bos restart** with the **-bosserver** flag for each database server machine
- rebooting each database server machine, either using **bos exec** or at their consoles

Step 11: If you did not already update *CellServDB* on client machines in step 5, do so now.

There are several ways to update *CellServDB* on client machines, as detailed in chapter 15 of the *AFS System Administrator's Guide*. You might, for instance, copy over the central update source (which you updated in step 4), with or without the **package** program.

To have a client machine's Cache Manager recognize the new database machine right away, add it to the machine's in-kernel list. Either reboot the machine after updating *CellServDB*, or issue the **fs newcell** command.

Step 12: Inform your AFS Product Support Representative of the name and Internet address of the new database server machine.

Transarc maintains a national *CellServDB* file, */afs/transarc.com/service/etc/CellServDB.export*, available to all AFS sites. Your new database server will be added to this file.

If you do not want other AFS sites to know about this database server, specify that you do not want your *CellServDB* exported to other cells. Transarc will place your cell in a private *CellServDB*.

3.4. Removing Database Server Functionality

The process of removing database server machine functionality is nearly the reverse of installing it.

3.4.1. Procedure Overview

To remove database server functionality from a machine, you will

1. install the **bos** suite of commands locally, as a precaution
2. inform Transarc Corporation of the removal of the database server machine
3. update your cell's central update source for *CellServDB*, and the file you make available to foreign users
4. update *CellServDB* on every client machine
5. remove the machine from */usr/afs/etc/CellServDB* on the system control machine
6. stop the database server processes on the machine
7. remove the database server processes from the machine's *BosConfig*
8. restart the database server processes on the remaining database server machines

3.4.2. Instructions

Step 1: As a precaution, make sure that the **bos** command suite binaries are available on the local disk. Having a local copy of **bos** commands is necessary in case of an error during installation, since the following procedures make the file system inaccessible.

- If you are working on a file server machine, the **bos** command binary already resides in */usr/afs/bin*. No action is necessary.
- If you are working on a client workstation, the **bos** binary usually resides in */usr/afsws/bin*, which is normally a symbolic link into AFS. If so, copy the **bos** binary to */tmp*.

```
# cp /usr/afsws/bin/bos  /tmp
```

Step 2: Inform your AFS Product Support Representative which database server machine you are removing from service.

This step is particularly important if your cell is included in the national *CellServDB* that Transarc makes available to all AFS sites (*/afs/transarc.com/service/etc/CellServDB.export*). If other cells do not know about the change, and so do not remove the machine from *CellServDB* on their client machines, then their users may experience delays while requests to the non-existent database server machine time out.

Step 3: Remove the database server machine from your cell's central update source for the client machine version of *CellServDB*, if there is one. The standard location for the central file is */afs/**cellname**/common/etc/CellServDB*.

If you maintain a file that foreign cells can consult to learn about your database server machines, remove the database server machine from it. Transarc recommends */afs/**cellname**/service/etc/CellServDB.local* as the standard location.

Step 4: Remove the database server machine from the */usr/vice/etc/CellServDB* file on every client machine in your cell. There are several ways to update *CellServDB* on client machines, as detailed in chapter 15 of the *AFS System Administrator's Guide*. You might, for instance, copy over the central update source (which you updated in the previous step), with or without the **package** program.

You remove the database server machine from *CellServDB* before stopping the actual database server processes to avoid delays in database server access. If a process on a client machine sends a request to a non-existent database server machine (because it is still listed in *CellServDB*), then users on the machine experience a delay while the request times out and is forwarded to another database server machine.

Step 5: Remove the former database server machine from the */usr/afs/etc/CellServDB* file on file server machines.

If using the United States edition of AFS, using the following command to update *CellServDB* on your cell's system control machine. Its server portion of the Update Server distributes the updated file to all other file server machines.

If using the international edition of AFS, you should not be using the Update Server to distribute files from */usr/afs/etc*. Instead, update *CellServDB* on each file server machine individually.

In the **bos removehost** command, provide the full Internet-style machine name (including cell extension) for both *machine name to update* and *new db-server machine*.

If using the United States edition of AFS, substitute the system control machine's name for *machine name to update*.

If using the international edition of AFS, repeat the **bos removehost** command once for each server machine in your cell, substituting each one's name for *machine name to update* in turn.

cd /usr/afs/bin

bos removehost <*machine name to update*> <*former db-server machine*>

If using the United States edition, wait several minutes to allow the Update Server to distribute the new *CellServDB* to all other AFS file server machines in the cell. By default this takes a maximum of five minutes. If using the international edition, attempt to issue all of the **bos removehost** commands within five minutes.

Step 6: Verify that the former database server machine no longer appears in *CellServDB* on all file server machines.

Note: Repeat this command on every file server machine.

bos listhosts <*machine name*>

Step 7: Stop the database server processes on the machine. This command changes the process' status in */usr/afs/local/BosConfig* to `NotRun`, but does not remove them from the file.

bos stop <*former db-server machine*> **kaserver ptserver vlserver buserver**

Step 8: **(Optional.)** Remove the entries for the database server processes from *BosConfig*. One reason not to execute this command is if you plan to re-install the database server functionality on this machine soon.

bos delete <*former db-server machine*> **kaserver ptserver vlserver buserver**

Step 9: Restart the database server processes on every remaining database server machine in the cell. This forces the election of a Ubik coordinator for each process, ensuring that the remaining database server processes recognize that the machine is no longer a database server and exclude it from the election.

A cell-wide service outage is possible during the election of a new Ubik coordinator for the VL Server, but should last less than five minutes. Messages tracing the progress of the election should appear on the console.

Repeat on every remaining database server machine:

bos restart <*database server machine*> **kaserver ptserver vlserver buserver**

If an error occurs (perhaps because you accidentally omit one of the processes on one machine), you should restart all server processes on the database server machines, either by

- issuing **bos restart** with the **-bosserver** flag for each database server machine
- rebooting each database server machine, either using **bos exec** or at their consoles

4. Installing Additional Client Machines

This chapter describes how to install any AFS client machine after the first AFS machine (which should already have been installed following the instructions in chapter 2).

Some parts of client installation differ depending on whether the new client is of the same AFS system type (uses the same AFS binaries) as a previously installed client machine, or is a new system type. Determine if the client you are installing is the same system type as an existing client, and follow the appropriate procedures.

Summary of Procedures

1. Load files that must reside on the local disk.
2. Incorporate AFS into the machine's kernel, either dynamically or by replacing */vmunix* or equivalent.
3. Define the machine's cell membership.
4. Define cache location and size.
5. Create the */usr/vice/etc/CellServDB* file to determine which cells the client can contact.
6. Create the */afs* directory and start the Cache Manager.
7. Set up volumes (necessary only for clients of a new system type).
8. Load client binaries into AFS (necessary only for clients of a new system type).
9. Create a link from the local */usr/afsws* to the AFS directory housing the binaries.
10. Replace the standard **login** binary with a version that both authenticates with AFS and logs into the local UNIX file system.

4.1. Loading Client Files onto the Local Disk

The first step in installing a client is to load AFS client binaries onto the local disk—namely, the **afsd** binary and files needed to incorporate AFS using a dynamic kernel loader (**dkload** or equivalent). These files must reside on the local disk of every client machine.

It is assumed that you have previously installed at least one machine as an AFS client (such as the first machine in your cell). How you load files on this machine depends on whether it is the same system type as an existing client, or a new type.

If the machine being installed is the same system type as an existing client, the files you need to load onto the local disk should already reside in the AFS directory */afs/**cellname**/**sysname**/usr/afsws*. In this case, you copy the files over the new client using a network transfer program. See section 4.1.1. (Use of this directory name assumes you followed the naming recommendations in sections 2.27 and 4.6 when installing previous machines. If you substituted different directory names, use them throughout this section.)

If the machine being installed is a new system type, the necessary client files do not reside in AFS. You must load them from the AFS Binary Distribution, following the instructions appropriate for your machine configuration, as discussed in section 1.4.2).

There are two ways to approach loading the client binaries for a new system type:

- load the files onto the local disk. See section 4.1.2 and then proceed through this chapter in order.
- first create and mount volumes to house the binaries for the new system type and load the appropriate files into the volumes, working on an existing client machine (of any type). This effectively transforms this system type into an existing one. To do this, skip now to section 4.6.2 and execute the instructions there. When finished, return to this section and follow the instructions for installing a machine of an existing system type.

4.1.1. Loading Files for a Machine of an Existing System Type

The following instructions assume that the necessary client files for this system type already reside in */afs/**cellname**/**sysname**/usr/afsws* (as recommended in sections 2.27 and 4.6). They copy the **afsd** and dynamic kernel loader files found in the *root.client/usr/vice/etc* subdirectory into */usr/vice/etc* on the local disk. These files must reside on the local disk of every client machine.

Step 1: Create */usr/vice/etc* on the local disk.

```
# mkdir /usr/vice

# mkdir /usr/vice/etc
```

Step 2: Using **ftp**, NFS, or another network file transfer program, access the files stored in

*/afs/**cellname**/**sysname**/usr/afsws/root.client/usr/vice/etc*

via an existing client machine (of any system type), and copy them into */usr/vice/etc* on the client machine being installed. Substitute the system type name of the machine being installed for ***sysname***.

4.1.2. Loading Files for a Machine of a New System Type

If you are installing an AFS client machine of a new system type, you must load the needed client files onto the local disk. Follow the instructions appropriate for your machine configuration.

4.1.2.1. Loading Files Using a Local Tape Drive

The fourth **tar** set on the AFS Binary Distribution tape contains the */usr/vice/etc* directory, consisting of **afsd** and the files needed to incorporate AFS using the dynamic kernel loader for this system type (*dkload* or equivalent). The following steps load it into */usr/vice/etc*.

Step 1: Create the */usr/vice/etc* directory.

```
# mkdir /usr/vice

# mkdir /usr/vice/etc
```

Step 2: Mount the Binary Distribution tape and load fourth **tar** set into */usr/vice/etc*.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/vice/etc

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 3

# tar xvf /dev/<device>
```

4.1.2.2. Loading Files From a Remote Machine

If the local machine does not have a tape drive, you must load the files found in */usr/afsws/root.client/usr/vice/etc* into the */usr/afsws* directory on a remote machine that has a tape drive. You then copy the appropriate files to the local machine's disk using **ftp**, NFS or another file transfer method.

Step 1: **Working on the remote machine,** create the */usr/afsws* directory.

```
# mkdir /usr/afsws
```

Step 2: Mount the AFS Binary Distribution tape on the remote machine and load the fifth **tar** set into */usr/afsws*.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/afsws

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 4

# tar xvf /dev/<device>
```

Step 3: **Working on the local machine,** remove any old version of */usr/vice* that may exist, and create the */usr/vice/etc* directory.

```
# rmdir /usr/vice

# mkdir /usr/vice

# mkdir /usr/vice/etc
```

Step 4: Using the appropriate (local or remote) file transfer program, copy the files from */usr/afsws/root.client/usr/vice/etc* to */usr/vice/etc* on the local machine.

4.2. Incorporating AFS Modifications into the Kernel

Every AFS client machine's kernel must incorporate the modifications that make up the Cache Manager. There are two ways to add AFS into the kernel: dynamic loading or kernel building. The table at the beginning of chapter 5 summarizes which method is available for each system type.

If dynamic loading is possible for your system type, it is the recommended method, as it is significantly quicker and easier than kernel building. A dynamic loader adds the AFS modifications to the memory version of the kernel created at each reboot, without altering the disk version (*/vmunix* or equivalent). See sections 4.2.1 through 4.2.6 for brief instructions for the different loaders, and pointers to more details if needed.

If you must (or choose to) build a kernel, see section 4.2.7 for existing system types or 4.2.8 for new system types. These sections provide brief instructions and pointers to more details.

4.2.1. Using Transarc's Dynamic Kernel Loader, dkload

The **dkload** program is the dynamic kernel loader provided by Transarc. It is available for the following operating systems:

- AOS
- HP-UX
- SunOS
- Ultrix

For operating systems not listed here, consult the table at the beginning of chapter 5 to learn what loading and building options are available.

For this machine to remain an AFS client, **dkload** must run each time it reboots. See step 3 below for details.

The files containing the AFS kernel modifications are delivered in standard and "stripped" versions. The standard versions are called *libafs.a* and *libafs.nonfs.a* (the latter is for machines that do not support NFS). The corresponding stripped versions are called *libafss.a* and *libafss.nonfs.a*; because they lack the symbol table found in the standard versions, they require somewhat less disk and core memory, but do not permit debugging. If you wish to use the stripped versions, substitute them for the standard versions in instructions below.

The following instructions are brief. More detailed instructions appear in section 5.13.

Step 1: Verify that

- there is at least one spare megabyte of space in */tmp* for temporary files created as **dkload** runs
- the following are in */bin* on the local disk (not as symbolic links): *as*, *ld*, and *nm*.
- the **/usr/vice/etc/dkload** directory on the local disk contains: *dkload* (the binary), *libafs.a*, *libafs.nonfs.a*, *libcommon.a*, and *kalloc.o*
- **afsd** has not run on this machine since the last reboot. If necessary, reboot without invoking **afsd**.

Step 2: Invoke **dkload**. On SunOS and Ultrix systems only, you must run **ranlib** first.

> **Note:** If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.a* for *libafs.a*. Either use the **mv** command to replace *libafs.a* with *libafs.nonfs.a* in the */usr/vice/etc/dkload* directory (before issuing these commands), or make the substitution on the command line. If you wish to use the "stripped" version of the AFS library file, substitute '*libafss*' for '*libafs*' in the same fashion.
>
> # **cd /usr/vice/etc/dkload**
>
> # **ranlib libafs.a** on SunOS and Ultrix systems only
>
> # **ranlib libcommon.a** on SunOS and Ultrix systems only
>
> # **./dkload libafs.a**

Step 3: If you wish **dkload** to run each time the machine reboots, invoke it in the machine's initialization file (*/etc/rc* or equivalent). An easy way to add the needed commands is to copy the contents of */usr/vice/etc/dkload/rc.dkload.client*; place them in the file after the instructions that mount the file systems but before **afsd** is invoked. The contents of this file appear in full in section 5.13.3. Remember to substitute *libafs.nonfs.a* for *libafs.a* if the machine's kernel does not support NFS. If you wish to use the "stripped" version of the AFS library file, substitute *libafss.a* for *libafs.a* or *libafss.nonfs.a* for *libafs.nonfs.a*.

4.2.2. Using the Kernel Extension Facility on AIX 3.2 Systems

The AIX kernel extension facility is a dynamic kernel loader provided by IBM Corporation for AIX 3.2. The **dkload** program is not available for this system type, nor is it possible to add AFS during a kernel build.

For this machine to remain an AFS client, the kernel extension facility must run each time it reboots. You can invoke the facility automatically by adding the contents of */usr/vice/etc/dkload/rc.dkload.client* to the machine's initialization files, as instructed in step 3 below.

The following instructions are brief. More detailed instructions appear in section 5.14.

To invoke the kernel extension facility manually:

Step 1: Verify that

- the */usr/vice/etc/dkload* directory on the local disk contains: *afs.ext*, *cfgafs*, *cfgexport*, *export.ext* and *export.ext.nonfs*.
- **afsd** has not run on this machine since the last reboot. If necessary, reboot without invoking **afsd**.
- NFS is already in the kernel (*nfs.ext* is already loaded), if you wish NFS to run on this machine. NFS must run for the machine to function as an NFS/AFS Translator machine.

Step 2: Invoke **cfgexport** and **cfgafs**.

Notes: If this machine will act as an NFS/AFS Translator machine, you must make a substitution in this step. For details, consult the section entitled "Setting Up an NFS/AFS Translator Machine" in the NFS/AFS Translator Supplement to the *AFS System Administrator's Guide*.

If NFS will not run on this machine, then you must substitute *export.ext.nonfs* for *export.ext* in this step. The machine cannot function as an NFS/AFS Translator in this case.

If the machine is to act as an NFS/AFS Translator, remember to make the necessary substitution. If the machine will not run NFS, substitute *export.ext.nonfs* for *export.ext*.

```
# cd /usr/vice/etc/dkload

# ./cfgexport -a export.ext

# ./cfgafs -a afs.ext
```

Step 3: IBM delivers several function-specific initialization files for AIX 3.2 systems, rather than the single file used on some other systems. If you want the kernel extension facility to run each time the machine reboots, verify that it is invoked in the appropriate place in these initialization files. An easy way to add the needed commands is to copy the contents of */usr/vice/etc/dkload/rc.dkload.client*, which appear in section 5.14.3.

The following list summarizes the order in which the indicated commands must appear in the initialization files for the machine to function properly. Note particularly that you should not invoke **nfsd** at the same place in the initialization files as the commands that load *nfs.ext*; **nfsd** must follow the call to **afsd**.

- command(s) mounting the file systems
- command(s) loading *nfs.ext*, if the machine will run NFS, which an NFS/AFS Translator machine must (but only for AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)
- the contents of *rc.dkload.client*, to invoke the kernel extension facility. If the machine will act as an NFS/AFS Translator machine, be sure to make the same substitution as you made in step 2.
- **afsd** (you will be instructed to add this command in section 4.5)
- **nfsd** (if the machine is to be an NFS server)

4.2.3. Using ml on IRIX Systems

The **ml** program is the dynamic kernel loader provided by Silicon Graphics, Inc. for IRIX systems. The **dkload** program is not available for these system types. It is possible to build AFS into the kernel for these system types; see section 4.2.7 or 4.2.8.

For this machine to remain an AFS client, **ml** must run each time the machine reboots. Step 4 below explains how to ensure this.

The following instructions are brief. More detailed instructions appear in section 5.15.

To invoke **ml** manually:

Step 1: Verify that

- the */usr/vice/etc/sgiload* directory on the local disk contains: *afs*, *afs.rc*, *afs.sm*, *libafs.R3000.o.dl*, *libafs.R3000.nonfs.o.dl*, *libafs.R4000.o.dl*, *libafs.R4000.nonfs.o.dl*.
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time it should not have, but reboot without invoking **afsd** if necessary.

Step 2: Remove the *.dl* extension from the AFS library file that is appropriate for this machine, selecting R3000 versus R4000 processor, and no NFS support versus NFS support.

If you do not know which processor your machine has, issue IRIX's **hinv** command and check the line in the output that begins "CPU."

Remove the *.dl* extension from one of the AFS library files by issuing the (one) appropriate command.

cd /usr/vice/etc/sgiload

For R3000 processor with NFS support:

cp libafs.R3000.o.dl libafs.R3000.o

For R3000 processor without NFS support:

cp libafs.R3000.nonfs.o.dl libafs.R3000.nonfs.o

For R4000 processor with NFS support:

cp libafs.R4000.o.dl libafs.R4000.o

For R4000 processor without NFS support:

cp libafs.R4000.nonfs.o.dl libafs.R4000.nonfs.o

Step 3: Issue the **ml** command.

Issue the following command, replacing *library file* with the name of the library file whose *.dl* extension who removed in step 2.

ml ld -v -j /usr/afs/bin/sgiload/<*library file*> **-p afs_ -a afs**

Step 4: Transarc provides an initialization script for IRIX systems as */usr/vice/etc/sgiload/afs.rc* that includes an **ml** instruction for loading AFS as the machine reboots. This script also includes automatic tests for whether the machine has the R3000 or R4000 processor and NFS support or no NFS support.

To ensure that *afs.rc* is invoked properly at each reboot, you must copy it from */usr/vice/etc/sgiload* to the initialization files directory (standardly, */etc/init.d*), make sure it is executable, and link it to the two locations where **ml** expects to find it. Then use the **chkconfig** command to create configuration files that tell **ml** that this machine will run AFS and serve as a client machine.

```
# cd /etc/init.d

# cp /usr/vice/etc/sgiload/afs.rc afs.rc

# chmod 555 afs.rc

# ln -s afs.rc /etc/rc0.d/K35afs.rc

# ln -s afs.rc /etc/rc2.d/S35afs.rc

# cd /etc/config

# chkconfig -f afsml on

# chkconfig -f afsclient on
```

4.2.4. Using kl_util on NeXT OS Systems

The **kl_util** program is the dynamic kernel loader provided by NeXT, Inc. for NeXT OS. The **dkload** program is not available for this system type, nor is it possible to add AFS during a kernel build.

For the client to remain an AFS machine, the AFS modifications must be added to the kernel each time the machine reboots. This happens automatically if you add the appropriate line to */etc/kern_loader.conf*, as instructed in step 4 below.

The following instructions for running **kl_util** manually are brief. More detailed instructions appear in section 5.16.

To invoke **kl_util** manually:

Step 1: Verify that

- *afs_reloc* exists in */usr/lib/kern_loader/AFS* on the local disk
- the **kl_util** binary is available on the local disk (standard location is */usr/etc*)
- **afsd** has not run on this machine since the last reboot. If necessary, reboot without invoking **afsd**.

Step 2: Copy the *afs_reloc* file provided under */usr/vice/etc/dkload* to the location where **kl_util** expects to find it (*/usr/lib/kern_loader/AFS/afs_reloc*).

Create the */usr/lib/kern_loader/AFS* directory and copy the *afs_reloc* file into it.

```
# mkdir  /usr/lib/kern_loader/AFS

# cp  /usr/vice/etc/dkload/afs_reloc  /usr/lib/kern_loader/AFS
```

Step 3: Invoke **kl_util**.

```
# /usr/etc/kl_util  -a  /usr/lib/kern_loader/AFS/afs_reloc
```

Step 4: If you wish **kl_util** to run each time the machine reboots, verify that the following line appears in */etc/kern_loader.conf* (any ordering with other lines is acceptable):

```
/usr/lib/kern_loader/AFS/afs_reloc
```

4.2.5. Using modload on Solaris Systems

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for Solaris systems. The **dkload** program is not available for these system types, nor it it possible to build AFS into the kernel.

For this machine to remain an AFS client, **modload** must run each time the machine reboots. Step 5 below explains how to ensure this.

The following instructions are brief. More detailed instructions appear in section 5.17.

To invoke **modload** manually:

Step 1: Verify that

- the **modload** binary is available on the local disk (standard location is */usr/sbin*)
- the */usr/vice/etc/modload* directory on the local disk contains *libafs.o* and *libafs.nonfs.o*
- **afsd** (the program that initializes the Cache Manager) has not run on this machine since the last reboot. Since you are installing this machine for the first time, it should not have, but reboot without invoking **afsd** if necessary.

Step 2: Create a copy of the appropriate AFS library file called */kernel/fs/afs*. You must use the command for *libafs.o* if the machine is to function as an NFS/AFS Translator machine.

```
# cd /usr/vice/etc/modload

# cp libafs.o  /kernel/fs/afs if the machine supports NFS

# cp libafs.nonfs.o  /kernel/fs/afs if the machine does not support NFS
```

Step 3: Create an entry for AFS in the */etc/name_to_sysnum* file, to allow the kernel to make AFS system calls.

In the file */etc/name_to_sysnum*, create an "afs" entry in slot 105 (the slot just before the "nfs" entry), so that the file looks like:

```
reexit          1
fork            2
 .              .
 .              .
 .              .
sigpending      99
setcontext      100
statvfs         103
fstatvfs        104
afs             105
nfs             106
```

Step 4: Invoke **modload**.

```
# /usr/sbin/modload  /kernel/fs/afs
```

If you wish to verify that AFS loaded correctly, use the **modinfo** command.

```
# /usr/sbin/modinfo | egrep afs
```

The following output indicates that AFS loaded successfully:

```
69 fc71f000 4bc15 105   1  afs (afs syscall interface)
69 fc71f000 4bc15  15   1  afs (afs file system)
```

Step 5: Transarc provides an initialization script for Solaris systems as */usr/vice/etc/modload/afs.rc* that includes an **modload** instruction for loading AFS as the machine reboots.

To ensure that *afs.rc* is invoked properly at each reboot, you must copy it from */usr/vice/etc/modload* to the initialization files directory (standardly, */etc/init.d*), make sure it is executable, and link it to the two locations where **modload** expects to find it.

```
# cd  /etc/init.d

# cp  /usr/vice/etc/modload/afs.rc  afs.rc

# chmod  555  afs.rc

# ln -s afs.rc  /etc/rc3.d/S14afs

# ln -s afs.rc  /etc/rc2.d/K66afs
```

4.2.6. Using modload on SunOS Systems

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for SunOS systems. The **dkload** program is also available for these system types, if you prefer to follow the instructions in section 4.2.1. It is also possible to build AFS into the kernel for these system types; see section 4.2.7 or 4.2.8.

For this machine to remain an AFS client, **modload** must run each time it reboots. You can invoke **modload** automatically by adding the contents of */usr/vice/etc/modload/rc.modload.client* to the machine's initialization file (*/etc/rc* or equivalent), as instructed in step 3 below.

The following instructions are brief. More detailed instructions appear in section 5.18.

To invoke **modload** manually:

Step 1: Verify that

- the */usr/vice/etc/modload* directory on the local disk contains *libafs.o*
- the **modload** binary is available on the local disk (standard location is */usr/etc*)
- **afsd** has not run on this machine since the last reboot. If necessary, reboot without invoking **afsd**.

Step 2: Invoke **modload**.

> **Note:** If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.o* for *libafs.o*. Either use the **mv** command to replace *libafs.o* with *libafs.nonfs.o* in the */usr/vice/etc/modload* directory (before issuing these commands), or make the substitution on the command line.
>
> # **cd /usr/vice/etc/modload**
>
> # **/usr/etc/modload ./libafs.o**

Step 3: If you wish **modload** to run each time the machine reboots, verify that it is invoked in the machine's initialization file (*/etc/rc* or equivalent). An easy way to add the needed commands is to copy the contents of */usr/vice/etc/modload/rc.modload.client*; place them in the file after the instructions that mount the file systems but before **afsd** is invoked. The contents of this file appear in full in section 5.18.3. If the machine's kernel does not support NFS, remember to substitute *libafs.nonfs.o* for *libafs.o*.

4.2.7. Installing a New Kernel on an Existing System Type

If this client machine is of an existing system type, you may already have built a kernel that incorporates AFS, using standard kernel-build techniques. If so, all you have to do is copy over the modified kernel from a remote machine of the same type. (If you have not previously built an AFS-modified kernel for this system type, follow the instructions in section 4.2.8.)

Step 1: Move the existing kernel to a save file.

```
# mv  /vmunix  /vmunix_save
```

Step 2: Using **ftp**, NFS or another network file transfer program, copy the AFS-modified kernel from the remote machine (where it may be stored under */usr/sys/**kernel_name**/vmunix*) into */vmunix* on this machine.

Step 3: Reboot the machine to start using the new kernel. This example instruction show the rebooting command appropriate for most system types.

```
# sync

# /etc/reboot
```

4.2.8. Building a New Kernel on a New System Type

This section explains how to install an AFS modified kernel after you have built it following the instructions in chapter 5.

Step 1: Follow the instructions in chapter 5 to build a kernel for this system type that includes AFS modifications.

Step 2: Move the existing kernel to a save file.

```
# mv  /vmunix  /vmunix_save
```

Step 3: If you built the kernel on a remote machine, use **ftp**, NFS or another network file transfer program to copy the kernel from the remote machine (where it may be stored under */usr/sys/**kernel_name**/vmunix*) into */vmunix* on this machine. If you built it locally, copy it from */usr/sys/**kernel_name**/vmunix* or equivalent to */vmunix*.

Step 4: Reboot the machine to start using the new kernel. This example instruction shows the rebooting command appropriate for most system types.

```
# sync

# /etc/reboot
```

4.3. Defining the Machine's Cell Membership and Creating *CellServDB*

Every client machine must have the */usr/vice/etc/ThisCell* file on its local disk to define which cell the machine belongs to. Among other functions, this file determines:

- the cell in which users authenticate when they log into this machine
- which cell's file server processes this machine's AFS command interpreters contact by default

Step 1: Place the cell name into *ThisCell*; surround it with quotes on the command line, but not with the angle brackets:

```
# echo "<cellname> " > /usr/vice/etc/ThisCell
```

Every client machine's */usr/vice/etc/CellServDB* file lists the database server machines in each cell that the local Cache Manager can contact. If a cell is not listed in this file, or its list of database server machines is wrong, then users working on this machine will be unable to access that cell's file tree. Your cell's client version of *CellServDB* should already exist; it was created during the installation of your cell's first machine (section 2.21).

Remember that the Cache Manager consults */usr/vice/etc/CellServDB* only at reboot, when it copies the information into the kernel. For the Cache Manager to perform properly, the *CellServDB* file must be accurate at all times. Refer to the *AFS System Administrator's Guide* for instructions on updating this file, with or without rebooting.

Step 1: Create */usr/vice/etc/CellServDB* on this machine.

If your cell maintains a central source copy of *CellServDB* (the standard name is */afs/**cellname**/common/etc/CellServDB*), use a network file transfer program such as *ftp* or NFS to copy the file to */usr/vice/etc/CellServDB* on this machine.

Otherwise, use a network file transfer program to copy the file over from an existing AFS client machine's */usr/vice/etc/CellServDB*.

4.4. Setting Up the Cache

Every AFS client must have a cache in which to store local copies of files brought over from file server machines. The Cache Manager can cache either on disk or in machine memory.

For both types of caching, **afsd** consults the */usr/vice/etc/cacheinfo* file as it initializes the Cache Manager and cache, to learn the defaults for cache size and where to mount AFS locally. For disk caches, it also consults the file to learn cache location. You must create this file for both types of caching.

The file has three fields:

- the first field defines where to mount AFS on the local disk. The standard choice is */afs*.
- the second field defines the local disk directory to use for caching, in the case of a disk cache. The standard choice is */usr/vice/cache*, but you could specify a different directory to take advantage of more space on other partitions. Something must appear in this field even if the machine uses memory caching.
- the third field defines cache size as a number of kilobyte (1024 byte) blocks. Make it as large as possible, but do not make the cache larger than 90% to 95% of the space available on the partition housing */usr/vice/cache* or in memory: the cache implementation itself requires a small amount of space. For AIX 3.2 systems using a disk cache, cache size cannot exceed 85% of the disk capacity reported by the **df** command. This difference between AIX 3.2 and other systems is because the output of AIX 3.2 **df** shows actual disk capacity and use, whereas BSD-based versions "hide" about 10% of disk capacity to allow for overage.

  Violating this restriction on cache size can cause errors or worsened performance.

  AFS versions 3.2 and 3.3 allow a cache as large as 1 gigabyte. Earlier versions of AFS impose a cache size limit of 700 megabytes.

  Disk caches smaller than 5 megabytes do not generally perform well, and you may find the performance of caches smaller than 10 megabytes unsatisfactory, particularly on system types that have large binary files. Deciding on a suitable upper limit is more difficult. The point at which enlarging the cache does not really improve performance depends on the number of users on the machine, the size of the files they are accessing, and other factors. A cache larger than 40 megabytes is probably unnecessary on a machine serving only a few users accessing files that are not huge. Machines serving multiple users may perform better with a cache of 60 to 70 megabytes.

  Memory caches smaller than 1 megabyte are nonfunctional, and most users find the performance of caches smaller than 5 megabytes to be unsatisfactory. Again, this depends on the number of users working on the machine, and the number of processes running. Machines running only a few processes may be able to use a smaller memory cache.

The **afsd** program also sets other cache configuration parameters as it initializes, and starts up several daemons that improve performance. The *AFS Command Reference Manual* page on **afsd** describes these parameters and daemons, and explains how to use **afsd**'s arguments to override the default

settings for parameters if desired. As discussed in section 4.5, AFS also provides initialization scripts that set certain **afsd** parameters appropriately for machines of different sizes.

4.4.1. Setting Up a Disk Cache

This section explains how to configure a disk cache.

Step 1: Create the cache directory. This example instruction shows the standard location, */usr/vice/cache*.

```
# mkdir /usr/vice/cache
```

Step 2: Create the *cacheinfo* file to define the boot-time defaults discussed above. This example instruction shows the standard mount location, */afs*, and the standard cache location, */usr/vice/cache*.

```
# echo "/afs:/usr/vice/cache:<#blocks>" > /usr/vice/etc/cacheinfo
```

For example, to devote 10000 one-kilobyte blocks to the cache directory on this machine, type:

```
# echo "/afs:/usr/vice/cache:10000" >/usr/vice/etc/cacheinfo
```

4.4.2. Setting Up a Memory Cache

This section explains how to configure a memory cache.

Step 1: Create the *cacheinfo* file to define the boot-time defaults discussed above. This example instruction shows the standard mount location, */afs*, and the standard cache location, */usr/vice/cache*. The location specified is irrelevant for a memory cache, but a value must be provided.

```
# echo "/afs:/usr/vice/cache:<#blocks>" > /usr/vice/etc/cacheinfo
```

For example, to devote 10000 kilobytes of memory to caching on this client machine, type:

```
# echo "/afs:/usr/vice/cache:10000" >/usr/vice/etc/cacheinfo
```

4.5. Creating */afs* and Starting the Cache Manager

As mentioned previously, the Cache Manager mounts AFS at the local */afs* directory. In this section you create that directory and then run **afsd** to initialize the Cache Manager.

You should also add **afsd** to the machine's initialization file (*/etc/rc* or equivalent), so that it runs automatically at each reboot. If **afsd** does not run at each reboot, the Cache Manager will not exist on this machine, and it will not function as an AFS client.

You can control several aspects of Cache Manager configuration using switches on the **afsd** command. Depending on this machine's cache size, its amount of RAM and the number of users who will work on it, you may be able to improve its performance as a client by overriding the default values of some switches.

AFS also provides a simpler alternative to setting **afsd**'s arguments individually, in the form of sets of parameter settings that are appropriate for client machines of three different sizes. For system types other than IRIX and Solaris, the settings are specified in three initialization scripts distributed in */usr/vice/etc/dkload*. The scripts are appropriate only for machines with a disk cache. Both the *AFS Command Reference Manual* and chapter 13 of the *AFS System Administrator's Guide* discuss these scripts in more detail. They are called:

- *rc.afsd.small*, which configures the Cache Manager appropriately for a "small" machine with a single user, about 8 megabytes of RAM and a 20-megabyte cache. It sets **-stat** to 300, **-dcache** to 100, **-daemons** to 2, and **-volumes** to 50.
- *rc.afsd.medium*, which configures the Cache Manager appropriately for a "medium" machine with two to six users, about 16 megabytes of RAM and a 40-megabyte cache. It sets **-stat** to 2000, **-dcache** to 800, **-daemons** to 3, and **-volumes** to 70.
- *rc.afsd.large*, which configures the Cache Manager appropriately for "large" machine with five to ten users, about 32 megabytes of RAM and a 100-megabyte cache. It sets **-stat** to 2800, **-dcache** to 2400, **-daemons** to 5, and **-volumes** to 128.

For IRIX and Solaris systems, the parameter settings are defined in the *afs.rc* initialization script, which you installed into */etc/init.d* as part of dynamic kernel loading (in section 4.2.3 for IRIX or section 4.2.5 for Solaris). The script defines `LARGE`, `MEDIUM` and `SMALL` values for the `OPTIONS` variable, which is then included on the `afsd` command line in the script. The *afs.rc* script is distributed with `OPTIONS` set to `$MEDIUM`, but you may change this as desired.

Step 1: Create the */afs* directory and verify that it is empty.

```
# mkdir /afs

# ls /afs
```

Step 2: Invoke **afsd**. For a memory cache, use the **-memcache** flag to indicate that the cache should be in memory rather than on disk.

With a disk cache, starting up **afsd** for the first time on a machine can take up to 10 minutes, because the Cache Manager has to create all of the structures needed for caching (V files). Starting up **afsd** at future reboots does not take nearly this long, since the structures already exist.

With a memory cache, the memory structures must be allocated at each reboot, but the process is equally quick each time. On system types other than IRIX and Solaris, you can substitute one of the configuration scripts (such as *rc.afsd.medium*) for **afsd** in the following command if desired (but you still must type the **-nosettime** and **-verbose** flags). Since those scripts are not available for IRIX or Solaris systems, you must type on the command line any additional configuration parameters you may wish to add.

For a disk cache:

Invoke **afsd**. On system types other than IRIX and Solaris, you can substitute one of the configuration scripts (such as *rc.afsd.medium*) for **afsd** in the following command if desired (but you still must type the **-verbose** flag). Since the three configuration scripts are not available for IRIX or Solaris, you must type on the command line any additional configuration parameters you wish to set.

```
# /usr/vice/etc/afsd -verbose &
```

For a memory cache:

Invoke **afsd** with the **-memcache** flag. You may specify values for additional parameters if desired.

```
# /usr/vice/etc/afsd -memcache -verbose &
```

Step 3: Invoke **afsd** in the machine's initialization file.

For a disk cache:

If the machine does not run IRIX or Solaris, add **afsd** to its initialization file (*/etc/rc* or equivalent), after the commands that invoke a dynamic kernel loader and the BOS Server, but before any NFS daemons (**nfsd**) are started. To invoke one of the configuration scripts described in the introduction to this section (such as *rc.afsd.medium*), substitute it for `afsd`:

```
/usr/vice/etc/afsd > /dev/console
```

If the machine runs IRIX or Solaris, verify that the `OPTIONS` variable in */etc/init.d/afs.rc* is set to the appropriate value; as distributed, it is `$MEDIUM`.

For a memory cache:

If the machine does not run IRIX or Solaris, add the following to */etc/rc* or equivalent. Add other configuration parameters if desired, but remember that the "*large*," "*medium*" and "*small*" scripts cannot be used with a memory cache.

```
/usr/vice/etc/afsd -memcache > /dev/console
```

If the machine runs IRIX or Solaris, verify that the `OPTIONS` variable in */etc/init.d/afs.rc* is not set to any of `$LARGE`, `$MEDIUM` or `$SMALL`; these values cannot be used with a memory cache.

For system types other than IRIX and Solaris, the following should now appear in the machine's initialization file in the indicated order:

- command(s) mounting the file systems
- command(s) loading *nfs.ext* (only for RS/6000s running NFS and AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)
- dynamic kernel loader command(s), unless AFS was built into the kernel
- **bosserver**
- **afsd**
- **nfsd** (if the machine is to be an NFS server)

4.6. Setting Up Volumes and Loading Binaries into AFS

In this section, you link */usr/afsws* on the local disk to the directory in AFS that houses AFS binaries for this system type. The standard name for the AFS directory is */afs/**cellname**/**sysname**/usr/afsws*.

If this machine is an existing system type, you can simply create a link from its */usr/afsws* directory to the appropriate AFS directory, which should already exist. Follow the instructions in section 4.6.1.

If this machine is a new system type (there are no file server or client machines of this type in your cell), you must first create and mount volumes to store its AFS binaries, and then create the link from */usr/afsws* to the new directory. Follow the instructions in section 4.6.2.

It is also recommended that you store UNIX system binaries (such as */bin*, */etc*, and */lib*) under */afs/**cellname**/**sysname***. Sections 2.27 and 2.28 provide guidelines; this section does not include explicit instructions.

4.6.1. Linking */usr/afsws* on an Existing System Type

If this client machine is an existing system type, then its client binaries should already reside in an AFS directory.

Step 1: Create a symbolic link from */usr/afsws* (a local disk directory) to */afs/**cellname**/@sys/usr/afsws*. You could also substitute the machine's Transarc system name for *@sys* (make the link to */afs/**cellname**/**sysname**/usr/afsws*). The advantage of using *@sys* is that it automatically adjusts in case you upgrade this machine to a different system type.

```
# ln -s /afs/<cellname>/@sys/usr/afsws /usr/afsws
```

You should include */usr/afsws/bin* and */usr/afsws/etc* in the PATH variable for each user account, so that users can issue commands from the AFS suites (such as **fs**).

4.6.2. Creating Binary Volumes for a New System Type

If this client machine is a new system type, you must create and mount volumes for its binaries before you can link the local */usr/afsws* to an AFS directory.

You must work on a machine that is already an AFS client or file server machine, because the commands used to create and mount volumes (from the **vos** and **fs** suites) are not yet available on this machine (it is a new system type and you have not yet extracted the needed command suites from the Binary Distribution).

The following procedure creates and mounts volumes to contain the AFS client binaries for this system type, loads the binaries from the Binary Distribution, and links the local */usr/afsws* to the

appropriate AFS directory. The remote machine you use should have a tape drive attached to simplify the extraction from tape in step 6.

Step 1: **Working on an existing AFS client or file server machine with a tape drive,** authenticate as the *admin* account created in section 2.10.

```
On a file server machine:

# /usr/afs/bin/klog admin
Password: <admin's password>

On a client machine:

# /usr/afsws/bin/klog admin
Password: <admin's password>
```

Step 2: **Working on the existing AFS machine,** create volumes to store AFS files and binaries for the new system type. The following example instructions create three volumes called "***sysname***", "***sysname***.usr" and "***sysname***.usr.afsws", enough to permit loading AFS client binaries into */afs/**cellname**/**sysname**/usr/afsws*. You may also wish to create volumes for UNIX and other system binaries, as outlined in the table in section 2.28.

```
# vos create <machine name> <partition name> <sysname>

# vos create <machine name> <partition name> <sysname>.usr

# vos create <machine name> <partition name> <sysname>.usr.afsws
```

Step 3: **Working on the existing AFS machine,** mount the volumes created in step 2 at the indicated place in the AFS file tree. Because `root.cell` is now replicated, you must make the mount points in its ReadWrite version, by preceding ***cellname*** with a period as shown. You then issue **vos release** to release new replicas of `root.cell`, and **fs checkvolumes** to force the local Cache Manager to access them.

```
# fs mkmount /afs/.<cellname>/<sysname> <sysname>

# fs mkmount /afs/.<cellname>/<sysname>/usr <sysname>.usr

# fs mkmount /afs/.<cellname>/<sysname>/usr/afsws <sysname>.usr.afsws

# vos release root.cell

# fs checkvolumes
```

Step 4: **Working on the existing AFS machine,** set the ACL on the mount points created in step 3 to grant the READ and LOOKUP rights to *system:anyuser*.

```
# cd  /afs/<cellname>/<sysname>

# fs  sa  -dir  .  ./usr  ./usr/afsws  -acl  system:anyuser  rl
```

Step 5: **Working on the existing AFS machine,** set the quota on */afs/**cellname**/**sysname**/usr/afsws* according to the following chart. The values include a safety margin.

Operating system	**Quota in kilobyte blocks**
AOS	25000
AIX	30000
HP-UX for 300 and 400 Series	25000
HP-UX for 700 and 800 Series	35000
IRIX	45000
NeXT OS	50000
Solaris	30000
SunOS	25000
Ultrix for Vaxen	25000
Ultrix for DECstations	45000

```
# fs setquota /afs/<cellname>/<sysname>/usr/afsws  <quota>
```

Step 6: **Working on the existing AFS machine,** mount the Binary Distribution tape and load the fifth **tar** set into */afs/**cellname**/**sysname**/usr/afsws*.

The appropriate subdirectories are created automatically, and have their ACL set to match that on */afs/**cellname**/**sysname**/usr/afsws* (which at this point grants all rights to *system:administrators* and READ and LOOKUP to *system:anyuser*).

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/afsws

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 4

# tar xvf /dev/<device>
```

Step 7: You may make AFS software available to users only in accordance with the terms of your AFS License agreement. To prevent access by unauthorized users, you should change the ACL on some of the subdirectories of */afs/**cellname**/**sysname**/usr/afsws*, granting the READ and LOOKUP rights to *system:authuser* instead of *system:anyuser*. This way, only users who are authenticated in your cell can access AFS binaries. The ACL on the *bin* subdirectory must continue to grant the READ and LOOKUP rights to *system:anyuser*, because unauthenticated users must be able to access the **klog** binary stored there.

To be sure that unauthorized users are not accessing AFS software, you should periodically check that the ACL on these directories is set properly.

To limit access to AFS binaries to users authenticated in your cell, issue the following commands. The ACL on the *bin* subdirectory must continue to grant the READ and LOOKUP rights to *system:anyuser*.

```
# cd /afs/<cellname>/<sysname>/usr/afsws

# fs sa -dir ./* -acl system:authuser rl system:anyuser none

# fs sa -dir bin -acl system:anyuser rl
```

Step 8: **Working on the new client machine**, create a symbolic link from */usr/afsws* (a local disk directory) to */afs/**cellname**/@sys/usr/afsws*. You could also substitute the machine's Transarc system name for *@sys* (make the link to */afs/**cellname**/**sysname**/usr/afsws*). The advantage of using *@sys* is that it automatically adjusts in case you upgrade this machine to a different system type.

```
# ln -s /afs/<cellname>/@sys/usr/afsws /usr/afsws
```

You should include */usr/afsws/bin* and */usr/afsws/etc* in the PATH variable for each user account, so that users can issue commands from the AFS suites (such as **fs**).

4.7. Replacing login

Transarc provides a version of **login** that both authenticates the issuer with AFS and logs him or her into the local UNIX file system. It is strongly recommended that you replace standard **login** with the AFS-authenticating version so that your cell's users automatically receive PAG-based tokens when they log in. If you do not replace the standard **login**, then your users must use the two-step login procedure (log into the local UNIX file system followed by **pagsh** and **klog** to authenticate with AFS). For more details, see the section titled ''Login and Authentication in AFS" in chapter 2 of the *AFS System Administrator's Guide*.

You can tell you are running AFS **login** if the following banner appears after you provide your password:

```
AFS 3.3 (R) login
```

To install AFS **login**, follow the instructions below that are appropriate for your system type:

- For AIX 3.2 systems, see section 4.7.1.
- For IRIX systems, see section 4.7.2.
- For NeXT OS systems, see section 4.7.3.
- For all other system types, see section 4.7.4.

4.7.1. Replacing login for AIX 3.2

Follow the instructions in this section to replace **login** on AIX 3.2 systems (system type *rs_aix32*).

For this system type, Transarc supplies both **login.noafs**, which is invoked when AFS is not running on the machine, and **login.afs**, which is invoked when AFS is running; these files are found in */usr/afsws/bin*, assuming you have followed the instructions in section 4.6. Note that standard AIX 3.2 **login** is normally installed as */usr/sbin/login*, with links to */etc/tsm*, */etc/getty* and */bin/login*. You will install the replacement AFS binaries into the */bin* directory.

Step 1: Replace the link to standard **login** in */bin* with **login.noafs**.

```
# mv  /bin/login  /bin/login.orig

# cp  /usr/afsws/bin/login.noafs  /bin/login
```

Step 2: Replace the links from */etc/getty* and */etc/tsm* to standard **login** with links to */bin/login*.

```
# mv  /etc/getty  /etc/getty.orig

# mv  /etc/tsm  /etc/tsm.orig

# ln -s  /bin/login  /etc/getty

# ln -s  /bin/login  /etc/tsm
```

Step 3: Install **login.afs** into */bin*, and create a symbolic link to */etc/afsok*.

```
# cp  /usr/afsws/bin/login.afs  /bin/login.afs

# ln -s  /bin/login.afs  /etc/afsok
```

4.7.2. Replacing login for IRIX Systems

Silicon Graphics, Inc. has modified the IRIX **login** program to operate in an AFS environment in the same manner as the AFS version. It is not necessary to replace standard **login** on IRIX systems. Transarc does not provide a modified **login** program for these systems.

4.7.3. Replacing the Login Window binary for NeXT OS

Follow the instructions in this section to replace the binary for the "Login Window" on NeXT OS systems (system type *next_mach30*) with a version that performs AFS **login**.

Note that on this system type the `AFS 3.3 (R) login` banner does not appear.

Step 1: Replace the standard **loginwindow** binary with the AFS version.

```
# cd  /usr/lib/NextStep/loginwindow.app

# mv  loginwindow  loginwindow.NextStep

# cp  /usr/afsws/bin/loginwindow  loginwindow
```

4.7.4. Replacing login for Other System Types

Follow the instructions in this section to replace **login** on other system types.

For these system types, the replacement AFS **login** binary resides in */usr/afsws/bin*, assuming you have followed the instructions in section 4.6. You install it as */bin/login*.

Step 1: Replace standard **login** with AFS **login**.

```
# mv  /bin/login  /bin/login.orig

# cp  /usr/afsws/bin/login  /bin/login
```

4.8. Altering File System Clean-Up Scripts

NeXT systems and many Sun systems are distributed with a crontab file that contains a command for removing unneeded files from the file system (it usually begins with the **find(1)** command). On NeXT systems, the standard location for this file is */usr/adm/daily*; on Sun systems, it is standardly */usr/spool/cron/crontabs/root*.

Once this machine is an AFS client, you must modify the pathname specification in this cron command to exclude */afs*. Otherwise, the command will traverse the entire portion of the AFS tree accessible from this machine, which includes every cell whose database server machines appear in the machine's kernel list (derived from */usr/vice/etc/CellServDB*). The traversal could take many hours.

Use care in altering the pathname specification, so that you do not accidentally exclude directories that you wish to be searched. The following may be suitable alterations, but are suggestions only—you must verify that they are appropriate for your system.

The first possible alteration requires that you list all file system directories to be searched.

On SunOS and NeXT OS systems, use:

```
find / /usr /<other partitions> -xdev remainder of existing command
```

On Solaris systems, use:

```
find / /usr /<other partitions> -mount remainder of existing command
```

The second possibility excludes any directories whose names begin with "*a*" or a non-alphabetic character.

```
find /[A-Zb-z]*  remainder of existing command
```

Note that you should not use the following, because it still searches under */afs*, looking for a subdirectory of type "4.2".

```
find / -fstype 4.2      /* do not use */
```

5. Incorporating AFS Into the Kernel

All AFS file server and client machines must run a kernel that includes AFS modifications. This chapter provides detailed information on adding AFS into the UNIX kernel of an AFS file server or client machine, either during a kernel build or using a dynamic loader. The sections on dynamic loading are intended as a reference for installers who are following the briefer instructions found in chapters 2, 3 and 4.

You can incorporate AFS into the kernel in two ways:

- by including AFS libraries when performing a standard kernel build. This method is appropriate if you are already building a UNIX kernel for other reasons.

  Sections 5.1 through 5.10 describe how to incorporate AFS during a kernel build.

- by using a "dynamic kernel loader" to incorporate AFS into the memory version of the kernel. This method does not require building a special kernel and is simpler. Transarc's dynamic kernel loader is called **dkload**. It is not available for all system types, because several manufacturers provide dynamic kernel loaders for their own operating systems.

  Sections 5.11 and following are a detailed reference for using dynamic kernel loaders.

The following table summarizes how you can use dynamic loading and/or kernel building for each supported operating system, and directs you to the section that describes the necessary procedure.

Operating system	Dynamic loading	Building
AIX 2.2.1	Not possible	See section 5.3
AIX 3.2	Use AIX kernel extension facility, provided by IBM (see section 5.14)	Not possible
AOS	Use **dkload** (see section 5.13)	See section 5.4
HP-UX for HP 9000 Series 300, 400 and 700 machines	Use **dkload** (see section 5.13)	See section 5.5
HP-UX for HP 9000 Series 800 machines	Use **dkload** (see section 5.13)	See section 5.6
IRIX	Use **ml**, provided by Silicon Graphics (see section 5.15)	See section 5.7
NeXT OS	Use **kl_util**, provided by NeXT (see section 5.16)	Not possible
Solaris	Use **modload**, provided by Sun (see section 5.17)	Not possible
SunOS	Use **dkload** (see section 5.13) or **modload**, provided by Sun (see section 5.18)	See section 5.8
Ultrix	Use **dkload** (see section 5.13)	Possible only with Ultrix source license (see section 5.9 for DECstations, section 5.10 for VAX systems)

5.1. Overview: Building a New Kernel

For the following system types, you can incorporate AFS into the kernel using normal kernel-build techniques. The Transarc system name for each type appears in parentheses.

- AIX 2.2.1 (*rt_aix221*). See section 5.3.
- AOS (*rt_aos4*). See section 5.4.
- HP-UX for HP 9000 Series 300, 400 and 700 machines (*hp300_ux90, hp700_ux90*). See section 5.5.
- HP-UX for HP 9000 Series 800 machines (*hp800_ux90*). See section 5.6.
- IRIX (*sgi_ir50, sgi_ir51*). See section 5.7.
- SunOS (*sun3_411, sun3x_411, sun4_411, sun4c_411, sun4m_412*). See section 5.8.
- Ultrix (*pmax_ul43, pmax_ul43a, vax_ul43*), if you have an Ultrix source license. See section 5.9 for *pmax_ul43* and *pmax_ul43a*, or 5.10 for *vax_ul43*.

For many of system types, you can also use a dynamic kernel loader to add AFS modifications into the memory version of the kernel. The table at the beginning of this chapter summarizes the loading and building options for each system type.

This section explains only how to build a kernel with AFS modifications in it. Procedures for installing the modified kernel on individual machines appear in chapters 2, 3 and 4.

5.1.1. Assumptions

Since compiling a kernel is a complicated and system-specific process, the following assumptions are made:

- the kernel system distribution directory is */usr/sys*. Adjust the pathname in the following procedures if your system distribution directory is different.
- the person building the kernel has a basic knowledge of kernel building. This section explains how to incorporate AFS into the kernel, but does not explain the kernel-build process.
- the person building the kernel is familiar with the different system types. The instructions for building a kernel are system-specific. Follow the instructions for your system type only.

5.2. Loading the Files Needed for a Kernel Build

Before beginning to build a kernel, you must load build-related files from the Binary Distribution tape to the local disk of the machine where you are performing the build. You can load files from the tape into the UNIX file system on either

- the local machine (where you are performing the build), if it has a tape drive attached. See section 5.2.1.
- a remote machine with a tape drive attached. You then copy them to the local machine's disk using **ftp**, NFS, or another network transfer method. See section 5.2.2.

5.2.1. Loading Files From a Local Tape Drive

The second **tar** set on the Binary Distribution tape contains the */usr/afs/sys* directory, which houses the library files needed to build AFS into the kernel: *libafs.a* and *libafs.nonfs.a* (which you should use if the kernel does not support NFS).

Follow the instructions to create the */usr/afs/sys* directory and load the appropriate files from the tape into it.

Step 1: Create the */usr/afs/sys* directory.

```
# mkdir /usr/afs

# mkdir /usr/afs/sys
```

Step 2: Mount the Binary Distribution tape and load the second **tar** set into */usr/afs/sys*.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/afs/sys

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 1

# tar xvf /dev/<device>
```

5.2.2. Loading Files From a Remote Machine

If the local machine does not have a tape drive, you must transfer the needed files onto the local disk from a remote machine. This section assumes you have already loaded the fifth **tar** set from the Binary Distribution tape to the directory */usr/afsws* on the remote machine. The */usr/afsws/root.client/bin* directory contains the needed library files.

Step 1: Create */usr/afs/sys* on the local disk (of the machine where you are building the kernel).

```
# mkdir /usr/afs

# mkdir /usr/afs/sys
```

Step 2: Using **ftp**, NFS, or another network file transfer method, copy the files from the remote machine's */usr/afsws/root.client/bin* directory to the */usr/afs/sys* directory on the local machine.

Step 3: Verify that */usr/afs/sys* contains *libafs.a* and *libafs.nonfs.a*.

```
# ls /usr/afs/sys
```

5.3. Building Kernels on AIX 2.2.1 Systems

This section explains how to add AFS into the kernel on IBM RT/PC systems running AIX 2.2.1 (system type *rt_aix221*).

IMPORTANT NOTE: With this system type, you must also include NFS when building the AFS kernel. AIX 2.2.1 imports the *xdr* set of modules (used by both NFS and AFS) only when NFS is built into the kernel. Consult your AIX 2.2.1 documentation for details.

Step 1: Add the following line to the include header file */usr/include/sys/vmount.h*, which contains definitions of all supported file systems, so that AFS is included.

Add the first line shown below to */usr/include/sys/vmount.h*. Change the number on the second line (the definition of `MNT_NUM`) accordingly.

```
#define MNT_AFS 3          /* Or higher, if slot 3 is already used */
#define MNT_NUM 4          /* Or higher, if you adjusted MNT_AFS */
```

Step 2: If you followed the loading instructions in section 5.2, the AFS kernel modules reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/bin*. Copy the appropriate AFS kernel module from */usr/afs/sys* to */usr/sys/libafs*: *libafs.a* if you will use the kernel on machines running NFS version 1.0, and *libafs.nfs11.a* if you will use it on machines running NFS version 1.1 or higher.

If the kernel is for machines running NFS version 1.0:

cp /usr/afs/sys/libafs.a /usr/sys/libafs

If the kernel is for machines running NFS version 1.1 or higher:

cp /usr/afs/sys/libafs.nfs11.a /usr/sys/libafs

Step 3: Edit the master *Makefile* in */usr/sys* to include the *libafs* module.

In the master *Makefile* in */usr/sys*, modify the line beginning "`LIBVFS =`" to read:

```
LIBVFS= libafs libnfs
```

Step 4: Change the number of callouts in */etc/master* to 100.

Change the `callouts` entry in */etc/master* to read:

```
        callouts = 100
```

Step 5: Issue the following command to extract the */usr/sys/sysconf* and */usr/sys/sysent.c* files from the archive format in which they exist in all AIX 2.2.1 binary object distributions. This command extracts the files using **ar(1)**.

```
# make sysconfig1
```

Step 6: Add an entry for AFS to */usr/sys/sysconf*, if it does not already exist.

Add the following line to */usr/sys/sysconf*, if it does not already appear there.

Note: If you set `MNT_AFS` to a value other than `3` in step 1, substitute that number here too.

```
define_gfs afs_vfsops afs_gn_vnodeops 3 "afs" afs_init GFS_REMOTE
```

Step 7: Alter */usr/sys/sysent.c* to include AFS system calls. In this step and the next, you alter the first of the three sections into which this file is divided, the "declarations" section. Then, in the following two steps, you alter the "sysaud[ ] array" and "sysent[ ] array" sections of */usr/sys/sysent.c*.

Make the following two alterations to the declarations section of */usr/sys/sysent.c*.

First, replace the `setgroups` call with `afs_xsetgroups`:

```
int     afs_xsetgroups();    /* was setgroups */
```

Second, add a declaration for `afs_syscall`:

```
int     afs_syscall();
```

The order of calls in the declaration section is not important, but you may wish to place the `afs_syscall` declaration after the `ifndef` section that follows `setdomainname`, so that when you are finished that part of the file looks like:

```
        .
        .
int     setdomainname();
#ifndef notdef
int    afs_xnfs_svc();
int    nfs_getfh();
int    exportfs();
int    async_daemon();
#endif
int     afs_syscall();
int     statfs();
        .
        .
```

Step 8: Now define several additional system calls in the declarations section of */usr/sys/sysent.c*.

Define the indicated calls in the declarations section of */usr/sys/sysent.c*. Although ordering of system calls does not matter in the declarations section, it makes sense to put calls in the same positions as they will appear in the sysaud[ ] and sysent[ ] arrays. The declaration section generally includes predefined empty slots for system calls, so you can replace the lines from `/*DECLARE_224*/` through `/*DECLARE_232*/` with the following:

```
int     afs_pioctl();   /* 224 */
int     afs_setpag();   /* 225 */
int     iopen();        /* 226 */
int     icreate();      /* 227 */
int     oiread();       /* 228 */
int     iwrite();       /* 229 */
int     iinc();         /* 230 */
int     idec();         /* 231 */
int     afs_call();     /* 232 */
```

Step 9: In the "sysaud[ ] array" section of */usr/sys/sysent.c*, add `afs_syscall` and replace `setgroups` with `afs_xsetgroups`.

In slot 38 of the sysaud[ ] array in */usr/sys/sysent.c*, replace

```
"nosys",          /* 38 = old version of uvmount */
```

with `afs_syscall`:

```
"afs_syscall",     /* 38 = afs_syscall*/
```

In slot 65, replace the existing definition of `setgroups` with:

```
"afs_xsetgroups",  /* 65 = setgroups */
```

Step 10: Add AFS system calls to the "sysent[ ] array" section of */usr/sys/sysent.c*.

In slot 38 of the sysent[ ] array in */usr/sys/sysent.c*, replace

```
0, 0, 0, nosys,          /* 38 = old version of uvmount */
```

with `afs_syscall`:

```
7, 0, 0, afs_syscall,    /* 38 = afs_syscall */
```

In slot 65, replace the existing definition of `setgroups` with:

```
2, 0, 0, afs_xsetgroups, /* 65 = setgroups */
```

Then insert the following in slots 224 through 232, replacing the current contents, which should be `0, 0, 0, nosys`.

```
    4, 0, 0, afs_pioctl,      /* 224 */
    0, 0, 0, afs_setpag,      /* 225 */
    3, 0, 0, iopen,           /* 226 */
    6, 0, 0, icreate,         /* 227 */
    6, 0, 0, oiread,          /* 228 */
    6, 0, 0, iwrite,          /* 229 */
    3, 0, 0, iinc,            /* 230 */
    3, 0, 0, idec,            /* 231 */
    6, 0, 0, afs_call,        /* 232 */
```

Step 11: Issue the following command to return the modified */usr/sys/sysconf* and */usr/sys/sysent.c* files to the archive. It runs **awk(1)** against the *sysconf* file, then compiles both it and *sysent.c*. It checks both files into the archive, along with the binaries that result, if the compilation is successful.

```
# make sysconfig2
```

Step 12: Link the new kernel. The *unix.std* that results includes AFS modifications and is ready for use.

```
# make
```

If you are installing your cell's first machine, return now to section 2.3.8.

If you are installing an additional file server machine (of a new system type), return to step 4 in section 3.2.2.

If you are installing an additional client machine, return to section 4.2.8.

5.4. Building Kernels on AOS Systems

This section explains how to add AFS into the kernel on IBM RT/PC systems running AOS 4.3 (system type *rt_aos4*).

IMPORTANT NOTE: IBM distributes AOS 4.3 with a version of AFS that predates the AFS 3.0 release of April 1990. The names of some of the AFS-specific system calls in the *init_sysent.c* file distributed with AOS 4.3 differ from the names in AFS 3. In particular, you must rename the IBM distribution's *afs_afscall* call to its AFS 3 name, *afs_call*.

Step 1: Declare AFS as a supported file system in */usr/sys/h/mount.h* (which is also accessible as */usr/include/sys/mount.h*), by making the indicated changes.

Note to object-distribution licensees: Some early object releases of AOS 4.3 do not include source code for */usr/sys/sys/vfs.c*. This file refers to `MOUNT_MAXTYPE`, the size of the virtual file system switch defined in */usr/sys/h/mount.h*. If you do not have a source license, then increasing the value of `MOUNT_MAXTYPE` in *mount.h* has no effect on the value in *vfs.c*. Therefore, you must instead reuse one of the existing slots in *mount.h*'s "vfssw" table, preferably slot 2 (PCFS). You could also reuse NFS's slot. Do not reuse slot 0 (UFS), because that is the basic UNIX file system.

If you do not make this change, `MOUNT_AFS` will be defined in slot 3 by default, as defined in */usr/afsws/include/afs/param.h*; this will not work, because slot 3 does not exist in the standard distribution.

If you have a source license, add the first line shown to */usr/sys/h/mount.h*, defining a value one higher than the previous `MOUNT_MAXTYPE`. In the second line, add one to the value of `MOUNT_MAXTYPE`.

```
#define MOUNT_AFS 3       /* Add 1 to the previous MOUNT_MAXTYPE */
#define MOUNT_MAXTYPE 3   /* Assuming it was previously 2 */
```

If you do not have a source license, replace the `MOUNT_PC` definition in */usr/sys/h/mount.h* with the line shown. Verify that `MOUNT_MAXTYPE` is set to `2`.

```
#define MOUNT_AFS 2       /* Replace the existing MOUNT_PC */
```

Step 2: Add one of the following lines to the kernel configuration file housed in */usr/sys/conf*. Place the line in the section of the configuration file that defines options (such as `options INET`). Which line to use depends on whether the kernel supports NFS or not.

Add a line to */usr/sys/conf/**config_name*** to define the AFS library file to use. The choice depends on whether the kernel support NFS or not.

If the kernel supports NFS, , use the following, which refers to *libafs.a*:

```
options AFS        # AFS client/server
```

If the kernel does not support NFS, use the following, which refers to *libafs.nonfs.a*:

```
options AFS_NONFS  # AFS using libafs.nonfs.a
                   # (no NFS support)
```

Step 3: Remove any reference to AFS that might appear in */usr/sys/conf/files* from the IBM distribution.

If */usr/sys/conf/files* refers to AFS, remove the reference.

Step 4: Add the AFS module, *libafs.o*, to the list of object modules in the prototype Makefile, */usr/sys/conf/makefile.ca*.

In */usr/sys/conf/makefile.ca*, replace

```
OTHEROBJS=xFPexception.o kFP0.o
```

with

```
OTHEROBJS=../afs/libafs.o xFPexception.o kFP0.o
```

Step 5: Add entries for AFS to */usr/sys/sys/vfs_conf.c*, which contains a table listing all supported file systems.

To */usr/sys/sys/vfs_conf.c*, add the following lines, which declare AFS among the other remote file system entries:

```
#ifdef AFS
       extern struct vfsops afs_vfsops;
#endif AFS
```

Add the following lines to insert the AFS file system entry into the table. The entry slot number you assign to `MOUNT_AFS` should match the one you used in */usr/sys/h/mount.h*, in step 1.

```
#ifdef  AFS
        &afs_vfsops,            /* 3 = MOUNT_AFS */
#else
        (struct vfsops *)0,
#endif AFS
```

Step 6: Substitute the following AFS-specific system calls for their standard UNIX counterparts in */usr/sys/sys/init_sysent.c*, which defines all supported system calls:

- *afs_xioctl* replaces *ioctl* (probably in slot 54)
- *afs_xsetgroups* replaces *setgroups* (probably in slot 80)
- *afs_xflock* replaces *flock* (probably in slot 131)

In */usr/sys/sys/init_sysent.c*, make the following substitutions:
Replace

```
        3, ioctl,               /* 54 = ioctl */
```

with

```
#ifdef AFS
        3, afs_xioctl,          /* 54 = afs_xioctl */
#else
        3, ioctl,               /* 54 = ioctl */
#endif AFS
```

Replace

```
        2, setgroups,           /* 80 = setgroups */
```

with

```
#ifdef AFS
        2, afs_xsetgroups,      /* 80 = afs_xsetgroups */
#else
        2, setgroups,           /* 80 = setgroups */
#endif AFS
```

Replace

```
        2, flock,               /* 131 = flock */
```

with

```
#ifdef AFS
        2, afs_xflock,          /* 131 = afs_xflock */
#else
        2, flock,               /* 131 = flock */
#endif AFS
```

For systems that will run the NFS/AFS Translator, also substitute *afs_xnfs_svc* for *nfs_svc* (probably in slot 169).

If the system will run the NFS/AFS translator, replace

```
        1, nfs_svc,             /* 169 = nfs_svc*/
```

with

```
#if     defined(AFS) && !defined(AFS_NONFS)
        1, afs_xnfs_svc,        /* 169 = afs_xnfs_svc */
#else
        1, nfs_svc,             /* 169 = nfs_svc*/
#endif  AFS
```

Step 7: Declare additional AFS-specific system calls in */usr/sys/sys/init_sysent.c*, using the single-entry system call method.

Add the following declarations to */usr/sys/sys/init_sysent.c*:

```
#ifdef  AFS
int     afs_syscall(), afs_xioctl(), afs_xnfs_svc();
int     afs_xflock(), afs_xsetgroups();
#endif AFS
```

Now declare `afs_syscall` in slot 31. (In most AOS 4.3 systems, slot 31 is not used, so it is the default choice for `afs_syscall`. If you have AFS source and wish to declare `afs_syscall` in a different slot, you must also redefine the slot number in */usr/afsws/include/afs/param.h*.)

```
#ifdef AFS
       7, afs_syscall,       /* 31 = afs entry call */
#else
       0, nosys,             /* 31 = was stty */
#endif AFS
```

Step 8: If you followed the loading instructions in section 5.2, the AFS kernel modules reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/bin*. Copy the appropriate AFS kernel module from */usr/afs/sys* to */usr/sys/sys*: either *libafs.a* if the kernel you are building supports NFS, or *libafs.nonfs.a* if it does not support NFS.

For kernels that support NFS:
cp /usr/afs/sys/libafs.a /usr/sys/sys/libafs.o

For kernels that do not support NFS:
cp /usr/afs/sys/libafs.nonfs.a /usr/sys/sys/libafs.o

Step 9: Run **ranlib** on */usr/sys/sys/libafs.o* to restore its table of contents, which was destroyed by **cp** in the previous step.

ranlib /usr/sys/sys/libafs.o

Step 10: Working in the */usr/sys/conf* directory, configure the kernel.

cd /usr/sys/conf

/etc/config <*config_name*>

Step 11: Working in the configuration directory, compile the kernel. The resulting kernel is suitable for (and must run on) both AFS clients and file server machines of this system type. It is created as */usr/sys/**config_name**/vmunix*.

```
# cd /usr/sys/<config_name>

# make depend

# make vmunix
```

If you are installing your cell's first machine, return now to section 2.3.8.

If you are installing an additional file server machine (of a new system type), return to step 4 in section 3.2.2.

If you are installing an additional client machine, return to section 4.2.8.

5.5. Building Kernels on HP 9000 Series 300, 400, and 700 Systems Running HP-UX 9.0

This section explains how to add AFS into the kernel on HP 9000 Series 300, 400 or 700 systems running HP-UX 9.0 (system types *hp300_ux90* and *hp700_ux90*).

Step 1: Copy the AFS kernel modules *libafs.a* and *libafs.nonfs.a* from the distribution directory to */etc/conf*. If you followed the loading instructions in section 5.2, the AFS kernel modules reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/bin*.

Copy the AFS library files to */etc/conf*.

cd /etc/conf

cp /usr/afs/sys/libafs.a .

cp /usr/afs/sys/libafs.nonfs.a .

Step 2: Add entries for AFS to */etc/conf/conf.c*.

To */etc/conf/conf.c*, add the following AFS declarations in the indicated places.

Add an "extern" declaration for `afscs_link()` right after the one for `nfsc_link()`, so that the result looks like the following:

```
        .
        .
        .
extern nipc_line();
extern nfsc_link();
extern afsc_link();
extern cdfs_link();
        .
        .
        .
```

Add an `afsc_link` entry to the `int (*driver_link[]) ()=` declaration, so that the result looks like the following:

```
sdisk_link,
    .
    .
    .
nipc_link,
nfsc_link,
afsc_link,
ite_link,
    .
    .
    .
```

Step 3: Edit */etc/conf/config.mk*, adding an AFS entry to the list of libraries defined as `LIBS`. The entry should refer to *libafs.a* if the kernel supports NFS, or to *libafs.nonfs.a* if it does not support NFS.

Edit */etc/conf/config.mk*, adding an AFS entry to the list of libraries defined as `LIBS`:

- if the kernel supports NFS, add an entry for `$(ROOT)/libafs.a`
- if the kernel does not support NFS, add an entry for `$(ROOT)/libafs.nonfs.a`

The file should look something like:

```
LIBS= \
        $(ROOT)/libuipc.a \
                  .
                  .
                  .
        $(ROOT)/libnfs.a \
        $(ROOT)/libafs.a \
        $(ROOT)/libcdfs.a \
                  .
                  .
```

Step 4: Working in */etc/conf*, compile the kernel. The resulting kernel is suitable for (and must run on) both AFS clients and file server machines of this system type. It is created it as */etc/conf/hp-ux*.

If the working directory is not still */etc/conf*, change directories.

cd /etc/conf

Compile the kernel.

make -f config.mk

If you are installing your cell's first machine, return now to section 2.3.8.

If you are installing an additional file server machine (of a new system type), return to step 4 in section 3.2.2.

If you are installing an additional client machine, return to section 4.2.8.

5.6. Building Kernels on HP 9000 Series 800 Systems Running HP-UX 9.0

This section explains how to add AFS into the kernel on HP 9000 Series 800 systems running HP-UX 9.0 (system type *hp800_ux90*).

Step 1: Copy the AFS kernel modules *libafs.a* and *libafs.nonfs.a* from the distribution directory to */etc/conf*. If you followed the loading instructions in section 5.2, the AFS kernel modules reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/bin*.

Copy the AFS library files to */etc/conf/lib*.

cd /etc/conf/lib

cp /usr/afs/sys/libafs.a .

cp /usr/afs/sys/libafs.nonfs.a .

Step 2: Add entries for the AFS library to */etc/master*, in both the list of file systems and in the "loadorder" section.

In */etc/master*, add an entry for AFS to the list of file systems. If the kernel supports NFS, add the AFS entry just after its entry, so that the file looks like

```
     .          .            .
     .          .            .
filesystem     nfs     lib libnfs.a;
filesystem     afs     lib libafs.a;
     .          .            .
     .          .            .
```

If the kernel does not support NFS, substitute `libafs.nonfs.a` for `libafs.a` and place the AFS entry after the entry for UFS.

Also add `libafs.a` (or `libafs.nonfs.a`) to the list of libraries in the "loadorder" section (the list appears on a single line in the file and appears here on multiple lines for space reasons only).

```
libsixq.a libpvc.a libsixtr.a libpsi.a libotsam.a \
libxtisys.a libufs.a libnfs.a libafs.a
```

Step 3: Make a copy of */etc/conf/gen/S800* as */etc/conf/gen/SAFS*, and add an entry for AFS to the copy.

```
# cd  /etc/conf/gen
# cp S800  SAFS
```

Now edit *SAFS* to include an entry for AFS right after the one for NFS, so that the file looks like

```
       .
       .
include switch;    /*include modules*/
include disc1;
include nfs;
include afs;
include lan1;
include lan3;
       .
       .
```

Step 4: Working in */etc/conf/gen* (you changed directory there in the previous step), compile the kernel. The resulting kernel, created as */etc/conf/SAFS*, is suitable for (and must run on) both AFS clients and file server machines of this system type.

Compile the kernel.

```
# /etc/uxgen  SAFS
```

5.7. Building Kernels on IRIX Systems

This section explains how to add AFS into the kernel on Silicon Graphics, Inc. Indigo machines running IRIX (system types *sgi_ir50, sgi_ir51*).

Step 1: Create */var/sysgen/system/afs.sm* as a symbolic link to the AFS initialization file. If you followed the loading instructions in section 5.2, the AFS kernel libraries and initialization file reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/usr/vice/etc/sgiload*.

Create */var/sysgen/system/afs.sm* as a symbolic link to the AFS initialization file, by substituting one of the following for *afs_library* in the **ln -s** command:

- *libafs.R3000.a* for R3000 processor with NFS support
- *libafs.R3000.nonfs.a* for R3000 processor without NFS support
- *libafs.R4000.a* for R4000 processor with NFS support
- *libafs.R4000.nonfs.a* for R4000 processor without NFS support

cd /var/sysgen/system

ln -s /usr/afs/sys/<*afs_library*> **afs.sm**

Step 2: Edit */var/sysgen/system/irix.sm* to allow AFS to reside in the kernel.

Edit the indicated section of */var/sysgen/system/irix.sm* to allow AFS to reside in the kernel. Normally this file is provided with the first set of `CCOPTS` and `LDOPTS` lines commented out with asterisks and the second set in force. Reverse this so that when you are finished the lines look like:

```
*   to load a kernel that can be co-resident with symmon
*   (for breakpoint debugging), replace CCOPTS and LDOPTS
*   with the following.  You must also INCLUDE prf and idbg.
*
CCOPTS: -xansi   -DECLIPSE -DIP12 -DECLIPSE -cckr   \
        -D_K32U32 -D_KERNEL -DSTATIC=static ....
LDOPTS: -non_shared -coff -N -e start -G 8 -coff   \
        -non_shared -T 80056000
*
*CCOPTS: -xansi  -DECLIPSE -DIP12 -DECLIPSE -cckr   \
         -D_K32U32  -D_KERNEL -DSTATIC=static ....
*LDOPTS: -non_shared -coff -N -e start -G 8 -coff   \
         -non_shared -T 80002000
```

Also verify that the file includes the line

```
INCLUDE: idbg
```

If necessary, add this line and remove the `EXCLUDE: idbg` line.

Step 3: Compile the kernel. The resulting kernel is suitable for (and must run on) both AFS clients and file server machines of this system type. It is created as */unix.install*.

Note: The existing */unix* is preserved when */unix.install* is created. However, the next time the machine is rebooted, */unix.install* will be rewritten to */unix*. If you wish to keep the original */unix* kernel, copy it to a save file.

```
# autoconfig
```

5.8. Building Kernels on SunOS Systems

This section explains how to add AFS into the kernel on Sun3, Sun4 and Sun SPARCstation machines running SunOS (system types *sun3_411*, *sun3x_411*, *sun4_411*, *sun4c_411*, *sun4m_412*).

Step 1: Add one of the following lines to the kernel configuration file housed in */usr/sys/**cpu**/conf*, where ***cpu*** is one of *sun3*, *sun3x*, *sun4*, *sun4c* or *sun4m*. Place the line in the section of the configuration file that defines options (such as `options INET`). Which line to use depends on whether the kernel supports NFS or not.

To */usr/sys/**cpu**/conf/**config_name***, add the appropriate line.

If the kernel supports NFS, use the following, which refers to *libafs.a*:

```
options AFS        # AFS client/server
```

If the kernel does not support NFS, use the following, which refers to *libafs.nonfs.a*:

```
options AFS_NONFS  # AFS using libafs.nonfs.a
                   # (no NFS support)
```

Step 2: Add the indicated line to */usr/sys/conf.common/files.cmn*, which lists all of the system-independent files constituting the new kernel.

Place the following line at the end of */usr/sys/conf.common/files.cmn*:

```
       afs/libafs.o     optional AFS
```

Step 3: Add entries for AFS to */usr/sys/os/vfs_conf.c*, which contains a table listing all supported file systems.

To */usr/sys/os/vfs_conf.c*, add the following lines, which declare AFS among the other remote file system entries:

```
#ifdef AFS
       extern struct vfsops afs_vfsops;
#endif AFS
```

Add the following lines to insert the AFS file system entry into the table. (You may place the entry anywhere in the table, because unlike some other operating system types, SunOS 4.1.1, 4.1.2 and 4.1.3 look up file system entries by name.)

```
#ifdef  AFS
        "afs", &afs_vfsops,     /* AFS */
#else
        (char *)0, (struct vfsops *)0,
#endif AFS
```

Step 4: Substitute the following AFS-specific system calls for their standard UNIX counterparts in */usr/sys/os/init_sysent.c*, which defines all supported system calls:

- *afs_xioctl* replaces *ioctl* (probably in slot 54)
- *afs_xsetgroups* replaces *setgroups* (probably in slot 80)
- *afs_xflock* replaces *flock* (probably in slot 131)

In */usr/sys/os/init_sysent.c*, make the following substitutions:
Replace

```
        3, ioctl,               /* 54 = ioctl */
```

with

```
#ifdef AFS
        3, afs_xioctl,          /* 54 = afs_xioctl */
#else
        3, ioctl,               /* 54 = ioctl */
#endif AFS
```

Replace

```
        2, setgroups,           /* 80 = setgroups */
```

with

```
#ifdef AFS
        2, afs_xsetgroups,      /* 80 = afs_xsetgroups */
#else
        2, setgroups,           /* 80 = setgroups */
#endif AFS
```

Replace

```
        2, flock,               /* 131 = flock */
```

with

```
#ifdef AFS
        2, afs_xflock,          /* 131 = afs_xflock */
#else
        2, flock,               /* 131 = flock */
#endif AFS
```

For systems that will run the NFS/AFS Translator, also substitute *afs_xnfs_svc* for *nfs_svc* (probably in slot 155).

If the system will run the NFS/AFS translator, replace

```
        1, nfs_svc,             /* 155 = nfs_svc*/
```

with

```
#if     defined(AFS) && !defined(AFS_NONFS)
        1, afs_xnfs_svc,        /* 155 = afs_xnfs_svc */
#else
        1, nfs_svc,             /* 155 = nfs_svc*/
#endif  AFS
```

Step 5: Declare additional AFS-specific system calls in */usr/sys/os/init_sysent.c*, using the single-entry system call method.

Add the following declarations to */usr/sys/os/init_sysent.c*:

```
#ifdef  AFS
int     afs_syscall(), afs_xioctl(), afs_xflock();
int     afs_xsetgroups(), afs_xnfs_svc();
#endif AFS
```

Now declare `afs_syscall` in slot 31. (In most SunOS 4.1.1, 4.1.2 and 4.1.3 systems, slot 31 is not used, so it is the default choice for `afs_syscall`. If you have AFS source and wish to declare `afs_syscall` in a different slot, you must also redefine the slot number in */usr/afsws/include/afs/param.h*.)

```
#ifdef AFS
       7, afs_syscall,       /* 31 = afs entry call */
#else
       0, nosys,             /*  31 = was stty */
#endif AFS
```

Step 6: If you followed the loading instructions in section 5.2, the AFS kernel modules reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/bin*. Copy the appropriate AFS kernel module from */usr/afs/sys* to the kernel binary object directory (*/usr/sys/**cpu**/OBJ*, where ***cpu*** is one of *sun3*, *sun3x*, *sun4*, *sun4c* or *sun4m*). Copy either *libafs.a* if the kernel you are building supports NFS, or *libafs.nonfs.a* if it does not support NFS.

For kernels that support NFS:

cp /usr/afs/sys/libafs.a /usr/sys/<*cpu*>**/OBJ/libafs.o**

For kernels that do not support NFS:

cp /usr/afs/sys/libafs.nonfs.a /usr/sys/<*cpu*>**/OBJ/libafs.o**

Step 7: Run **ranlib** on */usr/sys/**cpu**/OBJ/libafs.o* to restore its table of contents, which was destroyed by **cp** in the previous step.

ranlib /usr/sys/<*cpu*>**/OBJ/libafs.o**

Step 8: Working in the configuration directory */usr/sys/**cpu**/conf*, configure the kernel.

cd /usr/sys/<*cpu*>**/conf**

/etc/config <*config_name*>

Step 9: Working in the configuration kernel directory, */usr/sys/**cpu/config_name***, compile the kernel. The resulting kernel is suitable for (and must run on) both AFS clients and file server machines of this system type. It is created as */usr/sys/**cpu/config_name**/vmunix*.

```
# cd /usr/sys/<cpu>/<config_name>

# make vmunix
```

If you are installing your cell's first machine, return now to section 2.3.8.

If you are installing an additional file server machine (of a new system type), return to step 4 in section 3.2.2.

If you are installing an additional client machine, return to section 4.2.8.

5.9. Building Kernels on DECstations Running Ultrix

This section explains how to add AFS into the kernel on DECstations running Ultrix (system types *pmax_ul43* and *pmax_ul43a*).

Note: You must have an Ultrix source license to perform the following steps.

Step 1: Declare AFS as a supported file system in */usr/sys/h/fs_types.h* (which is also accessible as */usr/include/sys/fs_types.h*), by making the indicated changes. If you do not make these changes, `GT_AFS` will be defined in a default slot, possibly with undesirable results.

Add the following line to */usr/sys/h/fs_types.h*:

```
#define GT_AFS    0x0c     /* slot 12, or the next free slot */
```

Also add `"afs"` to the list of fs-type names. Put it in the twelfth position (or the one that matches the `GT_AFS` declaration), padding with `"unknown"` if necessary. The end result should look like:

```
char *gt_names[GT_NUMTYPES]={
"unknown", "ufs", "ods", "brfs", "msdos", "nfs", "dfs",
"sysv", "unknown", "unknown", "unknown", "afs", 0
}
```

Step 2: Add one of the following lines to the kernel configuration file housed in */usr/sys/conf/mips*. Place the line in the section of the configuration file that defines options (such as `options INET`). Which line to use depends on whether the kernel supports NFS or not. For both types of kernel, add the indicated line to the list of pseudo-devices.

To */usr/sys/conf/mips/**config_name***, add the appropriate line.

If the kernel supports NFS, use the following, which refers to *libafs.a*:

```
options AFS         # AFS client/server
```

If the kernel does not support NFS, use the following, which refers to *libafs.nonfs.a*:

```
options AFS_NONFS  # AFS using libafs.nonfs.a
                   #(no NFS support)
```

Also add the following to the list of pseudo-devices, for both types of kernel:

```
pseudo-device   AFS
```

Step 3: Add the indicated line to */usr/sys/conf/mips/files*, which lists all of the system-independent files constituting the new kernel. Also make the indicated changes to the lines associated with the *xdr* module files. These changes are necessary because these modules are normally brought in with NFS (they are usually marked "`optional nfs`"), but also need to be brought in with AFS.

Add the following line at the end of */usr/sys/conf/mips/files*:

```
afs/libafs.c        optional AFS Binary
```

Modify the *xdr* module lines as shown:

```
rpc/xdr.c               optional nfs or AFS Binary
rpc/xdr_array.c         optional nfs or AFS Binary
rpc/xdr_mbuf.c          optional nfs or AFS Binary
rpc/xdr_mem.c           optional nfs or AFS Binary
rpc/xdr_reference.c     optional nfs or AFS Binary
```

Step 4: Add entries for AFS to */usr/sys/data/gfs_data.c*, which contains a table listing all supported file systems.

To */usr/sys/data/gfs_data.c*, add the following lines, which declare AFS among the other remote file system entries:

```
#ifdef AFS
       extern struct mount *afs_gmount();
#endif AFS
```

Add the following lines to insert the AFS file system entry into the table. The entry slot number you assign to `GT_AFS` should match the one you used in */usr/sys/h/fs_types.h*, in step 1:

```
#ifdef AFS
       MOUNTFS(GT_AFS) = afs_gmount;
#endif AFS
```

Step 5: Substitute the following AFS-specific system calls for their standard UNIX counterparts in */usr/sys/sys/init_sysent.c*, which defines all supported system calls:

- *afs_xioctl* replaces *ioctl* (probably in slot 54)
- *afs_xsetgroups* replaces *setgroups* (probably in slot 80)
- *afs_xflock* replaces *flock* (probably in slot 131)

In */usr/sys/sys/init_sysent.c*, make the following substitutions:

Replace

```
        3, ioctl,               /* 54 = ioctl */
```

with

```
#ifdef AFS
        3, afs_xioctl,          /* 54 = afs_xioctl */
#else
        3, ioctl,               /* 54 = ioctl */
#endif AFS
```

Replace

```
        2, setgroups,           /* 80 = setgroups */
```

with

```
#ifdef AFS
        2, afs_xsetgroups,      /* 80 = afs_xsetgroups */
#else
        2, setgroups,           /* 80 = setgroups */
#endif AFS
```

Replace

```
        2, flock,               /* 131 = flock */
```

with

```
#ifdef AFS
        2, afs_xflock,          /* 131 = afs_xflock */
#else
        2, flock,               /* 131 = flock */
#endif AFS
```

Step 6: Declare additional AFS-specific system calls in */usr/sys/sys/init_sysent.c*, using the single-entry system call method.

Add the following declarations to */usr/sys/sys/init_sysent.c*:

```
#ifdef AFS
int    afs_syscall(), afs_xioctl(), afs_xnfs_svc();
int    afs_xflock(), afs_xsetgroups();
#endif AFS
```

Now declare `afs_syscall` in slot 31. (In most DECstation systems, slot 31 is not used, so it is the default choice for `afs_syscall`. If you have AFS source and wish to declare `afs_syscall` in a different slot, you must also redefine the slot number in */usr/afsws/include/afs/param.h*.)

```
#ifdef AFS
       7, afs_syscall,          /* 31 = afs entry call */
#else
       0, nosys,                /*  31 = was entry */
#endif AFS
```

Step 7: If you followed the loading instructions in section 5.2, the AFS kernel modules reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/bin*. Copy the appropriate AFS kernel module from */usr/afs/sys* to */usr/sys/MIPS/BINARY*: either *libafs.a* if the kernel you are building supports NFS, or *libafs.nonfs.a* if it does not support NFS.

Note: For Ultrix systems, the kernel library files found in */usr/afsws/root.client/usr/vice/etc/dkload* are not appropriate for kernel building. You must use the modules found in */usr/afs/sys*.

For kernels that support NFS:

cp /usr/afs/sys/libafs.a /usr/sys/MIPS/BINARY/libafs.o

For kernels that do not support NFS:

cp /usr/afs/sys/libafs.nonfs.a /usr/sys/MIPS/BINARY/libafs.o

Step 8: Run **ranlib** on */usr/sys/MIPS/BINARY/libafs.o*.

```
# ranlib /usr/sys/MIPS/BINARY/libafs.o
```

Step 9: Working in the configuration directory */usr/sys/conf/mips*, configure the kernel.

```
# cd /usr/sys/conf/mips

# /etc/config <config_name>
```

Step 10: Working in the configuration kernel directory, */usr/sys/MIPS/**config_name***, compile the kernel. The resulting kernel is suitable for (and must run on) both AFS clients and file server machines of this system type. It is created as */usr/sys/MIPS/**config_name**/vmunix*.

```
# cd /usr/sys/MIPS/<config_name>

# make depend

# make vmunix
```

If you are installing your cell's first machine, return now to section 2.3.8.

If you are installing an additional file server machine (of a new system type), return to step 4 in section 3.2.2.

If you are installing an additional client machine, return to section 4.2.8.

5.10. Building Kernels on VAXstations Running Ultrix

This section explains how to add AFS into the kernel on VAXstations running Ultrix (system type *vax_ul43*).

Note: You must have an Ultrix source license to perform the following steps.

Step 1: Declare AFS as a supported file system in */usr/sys/h/fs_types.h* (which is also accessible as */usr/include/sys/fs_types.h*), by making the indicated changes. If you do not make these changes, `GT_AFS` will be defined in a default slot, possibly with undesirable results.

Add the following line to */usr/sys/h/fs_types.h*:

```
#define GT_AFS    0x0c     /* slot 12, or the next free slot */
```

Also add `"afs"` to the list of fs-type names. Put it in the twelfth position (or the one that matches the `GT_AFS` declaration), padding with `"unknown"` if necessary. The end result should look like:

```
char *gt_names[GT_NUMTYPES]={
"unknown", "ufs", "ods", "brfs", "msdos", "nfs", "dfs",
"sysv", "unknown", "unknown", "unknown", "afs", 0
}
```

Step 2: Add one of the following lines to the kernel configuration file housed in */usr/sys/conf*. Place the line in the section of the configuration file that defines options (such as `options INET`). Which line to use depends on whether the kernel supports NFS or not. For both types of kernel, add the indicated line to the list of pseudo-devices.

To */usr/sys/conf/**config_name***, add the appropriate line.

If the kernel supports NFS, use the following, which refers to *libafs.a*:

```
options AFS         # AFS client/server
```

If the kernel does not support NFS, use the following, which refers to *libafs.nonfs.a*:

```
options AFS_NONFS  # AFS using libafs.nonfs.a
                   # (no NFS support)
```

Also add the following to the list of pseudo-devices, for both types of kernel:

```
pseudo-device   AFS
```

Step 3: Add the indicated line to */usr/sys/conf/files*, which lists all of the system-independent files constituting the new kernel. Also make the indicated changes to the lines associated with the *xdr* module files. These changes are necessary because these modules are normally brought in with NFS (they are usually marked "`optional nfs`"), but also need to be brought in with AFS.

Place the following line at the end of */usr/sys/conf/files*:

```
afs/libafs.c     optional AFS Binary
```

Modify the *xdr* module lines as shown:

```
rpc/xdr.c               optional nfs or AFS Binary
rpc/xdr_array.c         optional nfs or AFS Binary
rpc/xdr_mbuf.c          optional nfs or AFS Binary
rpc/xdr_mem.c           optional nfs or AFS Binary
rpc/xdr_reference.c     optional nfs or AFS Binary
```

Step 4: Add entries for AFS to */usr/sys/data/gfs_data.c*, which contains a table listing all supported file systems.

To */usr/sys/data/gfs_data.c*, add the following lines, which declare AFS among the other remote file system entries:

```
#ifdef AFS
       extern struct mount *afs_gmount();
#endif AFS
```

Add the following lines to insert the AFS file system entry into the table. The entry slot number you assign to `GT_AFS` should match the one you used in */usr/sys/h/fs_types.h*, in step 1:

```
#ifdef AFS
       MOUNTFS(GT_AFS) = afs_gmount;
#endif AFS
```

Step 5: Substitute the following AFS-specific system calls for their standard UNIX counterparts in */usr/sys/sys/init_sysent.c*, which defines all supported system calls:

- *afs_xioctl* replaces *ioctl* (probably in slot 54)
- *afs_xsetgroups* replaces *setgroups* (probably in slot 80)
- *afs_xflock* replaces *flock* (probably in slot 131)

In */usr/sys/sys/init_sysent.c*, make the following substitutions:

Replace

```
        3, ioctl,                       /* 54 = ioctl */
```

with

```
#ifdef AFS
        3, afs_xioctl,                  /* 54 = afs_xioctl */
#else
        3, ioctl,                       /* 54 = ioctl */
#endif AFS
```

Replace

```
        2, setgroups,                   /* 80 = setgroups */
```

with

```
#ifdef AFS
        2, afs_xsetgroups,              /* 80 = afs_xsetgroups */
#else
        2, setgroups,                   /* 80 = setgroups */
#endif AFS
```

Replace

```
        2, flock,                       /* 131 = flock */
```

with

```
#ifdef AFS
        2, afs_xflock,                  /* 131 = afs_xflock */
#else
        2, flock,                       /* 131 = flock */
#endif AFS
```

Step 6: Declare additional AFS-specific system calls in */usr/sys/sys/init_sysent.c*, using the single-entry system call method.

Add the following declarations to */usr/sys/sys/init_sysent.c*:

```
#ifdef  AFS
int     afs_syscall(), afs_xioctl(), afs_xnfs_svc();
int     afs_xflock(), afs_xsetgroups();
#endif AFS
```

Now declare `afs_syscall` in slot 31. (In most Vax systems, slot 31 is not used, so it is the default choice for `afs_syscall`. If you have AFS source and wish to declare `afs_syscall` in a different slot, you must also redefine the slot number in */usr/afsws/include/afs/param.h*.)

```
#ifdef AFS
       7, afs_syscall,           /* 31 = afs entry call */
#else
       0, nosys,                 /* 31 = was stty */
#endif AFS
```

Step 7: If you followed the loading instructions in section 5.2, the AFS kernel modules reside in */usr/afs/sys*; they are also accessible in */usr/afsws/root.client/bin*. Copy the appropriate AFS kernel module from */usr/afs/sys* to */usr/sys/VAX*: either *libafs.a* if the kernel you are building supports NFS, or *libafs.nonfs.a* if it does not support NFS.

Note: For Ultrix systems, the kernel library files found in */usr/afsws/root.client/usr/vice/etc/dkload* are not appropriate for kernel building. You must use the modules found in */usr/afs/sys*.

For kernels that support NFS:

cp /usr/afs/sys/libafs.a /usr/sys/VAX/libafs.o

For kernels that do not support NFS:

cp /usr/afs/sys/libafs.nonfs.a /usr/sys/VAX/libafs.o

Step 8: Run **ranlib** on */usr/sys/VAX/libafs.o* to restore its table of contents, which was destroyed by **cp** in the previous step.

ranlib /usr/sys/VAX/libafs.o

Step 9: Working in the configuration directory */usr/sys/conf*, configure the kernel.

```
# cd /usr/sys/conf

# /etc/config <config_name>
```

Step 10: Working in the configuration directory, compile the kernel. The resulting kernel is suitable for (and must run on) both AFS clients and file server machines of this system type. It is created as */usr/sys/**config_name**/vmunix*.

```
# cd /usr/sys/<config_name>

# make depend

# make vmunix
```

If you are installing your cell's first machine, return now to section 2.3.8.

If you are installing an additional file server machine (of a new system type), return to step 4 in section 3.2.2.

If you are installing an additional client machine, return to section 4.2.8.

5.11. Overview: Using Dynamic Kernel Loaders

This section explains how to use dynamic kernel loader programs to incorporate necessary AFS modifications into the memory version of a machine's kernel, eliminating the need to build a new kernel. The disk version of the kernel (*/vmunix* or equivalent) is not affected.

AFS must be dynamically loaded each time a machine boots for the machine to run AFS. You may wish to include loader initialization commands in each machine's initialization file (*/etc/rc* or equivalent). Instructions for altering the initialization file appear in the section about each type of kernel loader.

If you prefer to incorporate AFS by building a new kernel, first check the table at the beginning of the chapter to verify that building is possible, then follow the appropriate instructions in sections 5.3 through 5.10.

5.12. Loading Dynamic Kernel Loader Files

The first **tar** file on the Binary Distribution tape contains the files needed for running a dynamic kernel loader. Before running the program, you must load the files onto the local disk, in:

- */usr/afs/bin/dkload* on a file server machine
- */usr/vice/etc/dkload* on a client machine

There are two loading methods. The first requires that the local machine have a tape drive; see section 5.12.1. The second allows you to load the files onto a remote machine with a tape drive, and transfer them over to the local machine; see section 5.12.2.

5.12.1. Loading Dynamic Kernel Loader Files Using a Local Tape Drive

If the local machine has a tape drive, you can load the first **tar** set directly onto the local disk.

Step 1: Create the */usr/afs/bin* or */usr/vice/etc* directory as indicated.

On a file server machine:

```
# mkdir /usr/afs
# mkdir /usr/afs/bin
```

On a client machine:

```
# mkdir /usr/vice
# mkdir /usr/vice/etc
```

Step 2: Mount the AFS Binary Distribution tape and extract the first set of **tar** files into */usr/afs/bin* or */usr/vice/etc*. On Solaris and SunOS systems, the *modload* directory is created automatically; on IRIX systems, the *sgiload* directory is created automatically; on all systems except Solaris and IRIX, the *dkload* directory is created automatically.

On a file server machine:

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/afs/bin

# mt -f /dev/<device> rewind

# tar xvf /dev/<device>
```

On a client machine:

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd /usr/vice/etc

# mt -f /dev/<device> rewind

# tar xvf /dev/<device>
```

5.12.2. Loading Dynamic Kernel Loader Files From a Remote Machine

If the machine you are installing (the "local" machine) does not have a tape drive, you must transfer the needed files onto the local disk from an existing */usr/afsws* directory.

This section assumes you have already loaded the fifth **tar** set from the Binary Distribution tape to the */usr/afsws* directory. The the files necessary for running a dynamic kernel loader are in subdirectories of */usr/afsws/root.client/usr/vice/etc*:

- *modload* contains the files necessary for running **modload** on Solaris and SunOS systems.
- *sgiload* contains the files necessary for running **ml** on IRIX systems.
- *dkload* contains the files necessary for running other dynamic kernel loaders.

Step 1: Create the directory for housing dynamic kernel loader files on the local disk (under */usr/afs/bin* or */usr/vice/etc* as indicated).

On a file server machine:

```
# mkdir /usr/afs

# mkdir /usr/afs/bin

# mkdir /usr/afs/bin/modload on Solaris, and SunOS systems using modload

# mkdir /usr/afs/bin/sgiload on IRIX systems

# mkdir /usr/afs/bin/dkload on other system types
```

On a client machine:

```
# mkdir /usr/vice

# mkdir /usr/vice/etc

# mkdir /usr/vice/etc/modload on Solaris, and SunOS systems using modload

# mkdir /usr/vice/etc/sgiload on IRIX systems

# mkdir /usr/vice/etc/dkload on other system types
```

Step 2: Using **ftp**, NFS, or another network file transfer method, copy the files from the remote machine's */usr/afsws/root.client/usr/vice/etc/dkload* (or *modload*) directory to the local directory created in step 1.

5.13. Using dkload

The **dkload** program is a dynamic kernel loader provided by Transarc. It is available for the following operating systems:

- AOS
- HP-UX
- SunOS
- Ultrix

For other system types:

- for AIX 2.2.1, you must build a kernel to incorporate AFS, as described in section 5.3
- for AIX 3.2, use the AIX kernel extension facility as described in section 5.14
- for IRIX, use **ml** as described in section 5.15
- for NeXT OS, use **kl_util** as described in section 5.16
- for Solaris systems, use **modload**, as described in section 5.17
- for SunOS systems, you can also use **modload**, as described in section 5.18

5.13.1. Requirements for Running dkload

Before you use **dkload**, verify the following:

- the file server or client machine has a least one megabyte of free disk space in */tmp* for temporary files created as **dkload** runs
- the following files are on the local disk:
  - */bin/as*
  - */bin/ld*
  - */bin/nm*

- the *dkload* directory exists in the appropriate place on the local disk (under */usr/afs/bin* on file server machines or */usr/vice/etc* on client machines) and contains the following files:

  – *dkload* - dynamic kernel loader executable

  – *libafs.a* - AFS kernel library module

  – *libafs.nonfs.a* - AFS kernel library module for machines that do not run NFS

  – *libcommon.a* - library file used to link in *libafs.a* or *libafs.nonfs.a*

  – *kalloc.o* - memory allocator module

  – *rc.dkload.client* and *rc.dkload.server* - contain the commands to add to the machine's initialization file (*/etc/rc* or equivalent) to invoke **dkload** automatically at reboot

  If the directory or files do not exist, follow the instructions in section 5.12.

The files containing the AFS kernel modifications are delivered in standard and "stripped" versions. The standard versions are called *libafs.a* and *libafs.nonfs.a*, and the corresponding stripped versions are called *libafss.a* and *libafss.nonfs.a*. Because the stripped versions lack the symbol table found in the standard versions, they require somewhat less disk and core memory, but do not permit debugging. If you wish to use the stripped versions, substitute them for the standard versions in instructions below.

5.13.2. Running dkload

As previously explained, dynamic kernel loaders like **dkload** add a library of AFS modifications to the memory version of the kernel without changing the disk version (*/vmunix* or equivalent). Thus **dkload** must run each time the machine reboots, if the machine is to remain an AFS machine. You can automate dynamic loading by invoking **dkload** in the machine's initialization file (*/etc/rc* or equivalent). Section 5.13.3 explains how.

To run **dkload** manually:

Step 1: Verify that the local machine meets the requirements listed in section 5.13.1.

Note: The **dkload** program must run before the **afsd** program, which starts the Cache Manager. The Cache Manager cannot be operative when you run **dkload**.

Step 2: Invoke **dkload**. On SunOS and Ultrix systems only, you must run **ranlib** first.

On a file server machine:

> **Note:** If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.a* for *libafs.a*. Either use the **mv** command to replace *libafs.a* with *libafs.nonfs.a* in the */usr/afs/bin/dkload* directory (before issuing these commands), or make the substitution on the command line. If you wish to use the "stripped" version of the AFS library file, substitute '*libafss*' for '*libafs*' in the same fashion.
>
> # **cd /usr/afs/bin/dkload**
>
> # **ranlib libafs.a** on SunOS and Ultrix systems only
>
> # **ranlib libcommon.a** on SunOS and Ultrix systems only
>
> # **./dkload libafs.a**

On a client machine:

> **Note:** If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.a* for *libafs.a*. Either use the **mv** command to replace *libafs.a* with *libafs.nonfs.a* in the */usr/afs/bin/dkload* directory (before issuing these commands), or make the substitution on the command line. If you wish to use the "stripped" version of the AFS library file, substitute '*libafss*' for '*libafs*' in the same fashion.
>
> # **cd /usr/vice/etc/dkload**
>
> # **ranlib libafs.a** on SunOS and Ultrix systems only
>
> # **ranlib libcommon.a** on SunOS and Ultrix systems only
>
> # **./dkload libafs.a**

Step 3: If the machine's initialization file (*/etc/rc* or equivalent) does not already invoke **dkload**, insert the instructions found in section 5.13.3.

Step 4: If you are installing the first AFS machine, return now to section 2.3.2.

If you are installing an additional file server machine of an existing system type, return now to step 3 in section 3.1. If you are installing an additional file server machine of a new system type, return now to step 4 in section 3.2.

If you are installing an additional client machine, return now to section 4.2.1.

5.13.3. Altering the Initialization File to Invoke dkload

The section details the instructions you should place in the machine's start-up file (*/etc/rc* or equivalent) to invoke **dkload** automatically at each reboot.

For a file server machine, you can copy in the instructions found in */usr/afs/bin/dkload/rc.dkload.server*. If you wish to type them in, they are shown below. Place them in the file after the instructions that mount the file systems but before **afsd** is invoked.

Note: If the machine's kernel does not support NFS, substitute *libafs.nonfs.a* for *libafs.a*. If you wish to use the "stripped" version of the AFS library file, substitute *libafss.a* for *libafs.a* or *libafss.nonfs.a* for *libafs.nonfs.a*.

```
if [ -d /usr/afs/bin/dkload ]; then
  echo 'Invoking dynamic kernel loader...' > /dev/console
  cd /usr/afs/bin/dkload
  if [ -f libafs.a ]; then
      echo 'Loading AFS' > /dev/console
      ./dkload -quiet libafs.a 1> /dev/console 2>&1
      if [ $? != 0 ]; then
           echo 'Error loading AFS' > /dev/console
      fi
  else
      echo 'Cannot load AFS--no AFS kernel library' > /dev/console
  fi
  cd /
fi
```

For a client machine, you can copy in the instructions found in */usr/vice/etc/dkload/rc.dkload.client*. If you wish to type them in, they are shown below. Place them in the file after the instructions that mount the file systems but before **afsd** is invoked.

Note: If the machine's kernel does not support NFS, substitute *libafs.nonfs.a* for *libafs.a*. If you wish to use the "stripped" version of the AFS library file, substitute *libafss.a* for *libafs.a* or *libafss.nonfs.a* for *libafs.nonfs.a*.

```
if [ -d /usr/vice/etc/dkload ]; then
  echo 'Invoking dynamic kernel loader...' > /dev/console
  cd /usr/vice/etc/dkload
  if [ -f libafs.a ]; then
      echo 'Loading AFS' > /dev/console
      ./dkload -quiet libafs.a 1> /dev/console 2>&1
      if [ $? != 0 ]; then
           echo 'Error loading AFS' > /dev/console
      fi
  else
      echo 'Cannot load AFS--no AFS kernel library' > /dev/console
  fi
  cd /
fi
```

5.14. Using the Kernel Extension Facility on AIX 3.2 Systems

The AIX kernel extension facility is a dynamic kernel loader provided by IBM Corporation for AIX 3.2 systems. The **dkload** program is not available for these systems, nor is it possible to incorporate AFS during a kernel build.

5.14.1. Requirements for Running the Kernel Extension Facility

Before you use the kernel extension facility, verify that the *dkload* directory exists in the appropriate place on the local disk (under */usr/afs/bin* on file server machines or */usr/vice/etc* on client machines) and contains the following files:

- *afs.ext*
- *cfgafs*
- *cfgexport*
- *export.ext*
- *export.ext.nonfs*
- *rc.dkload.client* and *rc.dkload.server* - contain the commands to add to the machine's startup file (*/etc/rc* or equivalent) to invoke the kernel extension facility automatically at reboot

If the directory or files do not exist, follow the instructions in section 5.12.

Also verify that NFS is already in the kernel (*nfs.ext* is already loaded), if you wish NFS to run on this machine. NFS must run for the machine to function as an NFS/AFS Translator machine.

5.14.2. Running the Kernel Extension Facility

As previously explained, dynamic kernel loaders like the AIX kernel extension facility add a library of AFS modifications to the memory version of the kernel without changing the disk version (*/vmunix* or equivalent). Thus the kernel extension facility must run each time the machine reboots, if the machine is to remain an AFS machine. You can automate dynamic loading by invoking the kernel extension facility in the machine's initialization files. Section 5.14.3 explains how.

To run the kernel extension facility manually:

Step 1: Verify that the local machine's local disk contains the files listed in section 5.14.1.

Important Notes: The kernel extension facility must run before the **afsd** program, which starts the Cache Manager. The Cache Manager cannot be operative when you run the facility.

NFS should already be loaded into the kernel, if you wish it to run on this machine. NFS must run for the machine to function as an NFS/AFS Translator machine.

Step 2: Invoke **cfgexport** and **cfgafs**.

Notes: If this machine will act as an NFS/AFS Translator machine, you must make a substitution in this step. For details, consult the section entitled "Setting Up an NFS/AFS Translator Machine" in the NFS/AFS Translator Supplement to the *AFS System Administrator's Guide*.

If NFS will not run on this machine, then you must substitute *export.ext.nonfs* for *export.ext* in this step. The machine cannot function as an NFS/AFS Translator in this case.

On a file server machine:

If the machine is to act as an NFS/AFS Translator, remember to make the necessary substitution. If the machine will not run NFS, substitute *export.ext.nonfs* for *export.ext*.

```
# cd /usr/afs/bin/dkload

# ./cfgexport -a export.ext

# ./cfgafs -a afs.ext
```

On a client machine:

If the machine is to act as an NFS/AFS Translator, remember to make the necessary substitution. If the machine will not run NFS, substitute *export.ext.nonfs* for *export.ext*.

```
# cd /usr/vice/etc/dkload

# ./cfgexport -a export.ext

# ./cfgafs -a afs.ext
```

Step 3: If necessary, alter the machine's initialization files to invoke the kernel extension facility automatically, as detailed in section 5.14.3.

Step 4: If you are installing the first AFS machine, return now to section 2.3.3.

If you are installing an additional file server machine of an existing system type, return now to step 3 in section 3.1. If you are installing an additional file server machine of a new system type, return now to step 4 in section 3.2.

If you are installing an additional client machine, return now to section 4.2.2.

5.14.3. Altering the Initialization File to Invoke the Kernel Extension Facility

IBM delivers several function-specific initialization files for AIX 3.2 systems, rather than the single file used on some other systems. The section details the instructions you should place in one of the machine's initialization files to invoke the kernel extension facility automatically at each reboot.

The following list summarizes the order in which the indicated commands must appear in the initialization files for the machine to function properly (you will add some of the commands in later sections). Note particularly that you should not invoke **nfsd** at the same place in the initialization files as the commands that load *nfs.ext*; **nfsd** must follow the call to **afsd**.

- command(s) mounting the file systems
- command(s) loading *nfs.ext*, if the machine will run NFS, which an NFS/AFS Translator machine must (but only for AIX version 3.2.2 or lower; with AIX version 3.2.3 and higher, NFS is already loaded into the kernel)
- the commands shown below to invoke the kernel extension facility. If the machine will act as an NFS/AFS Translator machine, be sure to make the same substitution as you made when running the **cfgafs** command in section 5.14.2. The section entitled "Setting Up an NFS/AFS Translator Machine" in the NFS/AFS Translator Supplement to the *AFS System Administrator's Guide* explains the substitution.
- **bosserver** (if this is a file server machine)
- **afsd** (if this is a client machine)
- **nfsd** (if the machine is to be an NFS server)

For a file server machine, you can copy in the instructions found in */usr/afs/bin/dkload/rc.dkload.server*. If NFS will not run on the machine, substitute *export.ext.nonfs* for *export.ext*. If you wish to type in the instructions, they are:

```
echo 'Loading AFS ...'>/dev/console
/usr/afs/bin/dkload/cfgexport -a /usr/afs/bin/dkload/export.ext
/usr/afs/bin/dkload/cfgafs  -a /usr/afs/bin/dkload/afs.ext
```

For a client machine, you can copy in the instructions found in */usr/vice/etc/dkload/rc.dkload.client*. If NFS will not run on the machine, substitute *export.ext.nonfs* for *export.ext*. If you wish to type in the instructions, they are:

```
echo 'Loading AFS ...'>/dev/console
/usr/vice/etc/dkload/cfgexport -a /usr/vice/etc/dkload/export.ext
/usr/vice/etc/dkload/cfgafs  -a /usr/vice/etc/dkload/afs.ext
```

5.15. Using ml on IRIX Systems

The **ml** program is the dynamic kernel loader provided by Silicon Graphics, Inc. for IRIX systems. The **dkload** program is not available for these system types.

5.15.1. Requirements for Running ml

Before you use **ml**, you must verify that:

- the *sgiload* directory exists in the appropriate place on the local disk (under */usr/afs/bin* on file server machines or */usr/vice/etc* on client machines), and contains
  - *afs*
  - *afs.rc*
  - *afs.sm*
  - *libafs.R3000.o.dl*
  - *libafs.R3000.nonfs.o.dl*
  - *libafs.R4000.o.dl*
  - *libafs.R4000.nonfs.o.dl*
- If the directory or files do not exist, follow the instructions in section 5.12.

Further, before running **ml** for the first time on a given machine, you must perform the following step.

Step 1: Remove the *.dl* extension from one of the AFS library files by issuing the (one) appropriate command.

```
# cd  /usr/afs/bin/sgiload on a file server machine

# cd  /usr/vice/etc/sgiload on a client machine

For R3000 processor with NFS support:

# cp  libafs.R3000.o.dl  libafs.R3000.o

For R3000 processor without NFS support:

# cp  libafs.R3000.nonfs.o.dl  libafs.R3000.nonfs.o

For R4000 processor with NFS support:

# cp  libafs.R4000.o.dl  libafs.R4000.o

For R4000 processor without NFS support:

# cp libafs.R4000.nonfs.o.dl libafs.R4000.nonfs.o
```

5.15.2. Running ml

As previously explained, dynamic kernel loaders like **ml** add a library of AFS modifications to the memory version of the kernel without changing the disk version (*/unix* or equivalent). Thus **ml** must run each time the machine reboots, if the machine is to remain an AFS machine. Section 5.15.3 explains how to automate dynamic loading for IRIX systems.

To run **ml** manually:

Step 1: Verify that the machine meets the requirements listed in section 5.15.1.

Important Note: The **ml** program must run before the **afsd** program, which starts the Cache Manager. The Cache Manger cannot be operative when you run **ml**.

Step 2: Run **ml**.

Issue the following command, replacing *library file* with the name of the library file whose *.dl* extension you removed in step 1 of section 5.15.1.

ml ld -v -j /usr/afs/bin/sgiload/<*library file*> **-p afs_ -a afs**

Step 3: Follow the instructions in section 5.15.3 to automate dynamic kernel loading.

Step 4: If you are installing the first AFS machine, return now to section 2.3.6.

If you are installing an additional file server machine of an existing system type, return now to step 3 in section 3.1. If you are installing an additional file server machine of a new system type, return now to step 4 in section 3.2.

If you are installing an additional client machine, return now to section 4.2.5.

5.15.3. Installing the AFS Initialization Script on IRIX Systems

Transarc provides an initialization script for IRIX systems that includes an **ml** instruction for loading AFS as the machine reboots. The script is distributed as */usr/afs/bin/sgiload/afs.rc* for file server machines and as */usr/vice/etc/sgiload/afs.rc* for client machines.

To ensure that *afs.rc* is invoked properly at each reboot, you must copy it from its distribution directory to the initialization files directory (standardly, */etc/init.d*), make sure it is executable, and link it to the two locations where **ml** expects to find it. Then use the **chkconfig** command to create configuration files that tell **ml** that this machine will run AFS, and whether it will serve as a file server machine or client machine.

Step 1: Create a copy of the *afs.rc* script in the */etc/init.d* directory.

```
# cd  /etc/init.d
# cp  /usr/afs/bin/sgiload/afs.rc  afs.rc for file server machines
# cp  /usr/vice/etc/sgiload/afs.rc  afs.rc for client machines
# chmod  555  afs.rc
# ln -s afs.rc  /etc/rc0.d/K35afs.rc
# ln -s afs.rc  /etc/rc2.d/S35afs.rc
```

Step 2: Create configuration files to tell **ml** which function the machine will serve.

On a file server machine:

```
# cd  /etc/config
# chkconfig  -f  afsml  on
# chkconfig  -f  afsserver  on
```

On a client machine:

```
# cd  /etc/config
# chkconfig  -f  afsml  on
# chkconfig  -f  afsclient  on
```

5.16. Using kl_util on NeXT OS Systems

The **kl_util** program is the dynamic kernel loader provided by NeXT, Inc. for NeXT OS systems. The **dkload** program is not available for this system type, nor is it possible to incorporate AFS during a kernel build.

5.16.1. Requirements for Running kl_util

Before you use the kernel extension facility, verify that the **kl_util** binary exists in the appropriate place on the local disk. You must also copy the *afs_reloc* file provided in the AFS Binary Distribution to the place where **kl_util** expects to find it, */usr/lib/kern_loader/AFS/afs_reloc*.

Step 1: Verify that *afs_reloc* exists in */usr/afs/bin/dkload* (on a file server machine) or */usr/vice/etc/dkload* (on a client machine). If it does not, follow the instructions in section 5.12 to load it from the Binary Distribution into the appropriate directory.

Step 2: Create the */usr/lib/kern_loader/AFS* directory and copy *afs_reloc* file into it. (This instruction assumes that you store the **kl_util**-related files provided by NeXT in the standard location, */usr/lib/kern_loader*.)

On a file server machine:

```
# mkdir  /usr/lib/kern_loader/AFS

# cp  /usr/afs/bin/dkload/afs_reloc  /usr/lib/kern_loader/AFS
```

On a client machine:

```
# mkdir  /usr/lib/kern_loader/AFS

# cp  /usr/vice/etc/dkload/afs_reloc  /usr/lib/kern_loader/AFS
```

5.16.2. Running kl_util

As previously explained, dynamic kernel loaders like **kl_util** add a library of AFS modifications to the memory version of the kernel without changing the disk version (*/vmunix* or equivalent). Thus **kl_util** must run each time the machine reboots, if the machine is to remain an AFS machine.

To run **kl_util** manually:

Step 1: Verify that the local machine meets the requirements listed in section 5.16.1.

Important Note: The **kl_util** program must run before the **afsd** program, which starts the Cache Manager. The Cache Manager cannot be operative when you run **kl_util**.

Step 2: Run **kl_util**.

```
# /usr/etc/kl_util  -a  /usr/lib/kern_loader/AFS/afs_reloc
```

Step 3: Messages in the console window should report that AFS was loaded. You can also check explicitly using one of two methods:

- Run **kl_util** with the **-s** flag to check on the status of the kernel loads. AFS should be mentioned.

  ```
  # /usr/etc/kl_util  -s
  ```

- Inspect */usr/adm/messages* for a mention of loading AFS.

  ```
  # tail /usr/adm/messages
  ```

Step 4: If you wish **kl_util** to run each time the machine reboots, place the following line in */etc/kern_loader.conf*:

```
/usr/lib/kern_loader/AFS/afs_reloc
```

Step 5: If you are installing the first AFS machine, return now to section 2.3.5.

If you are installing an additional file server machine of an existing system type, return now to step 3 in section 3.1. If you are installing an additional file server machine of a new system type, return now to step 4 in section 3.2.

If you are installing an additional client machine, return now to section 4.2.4.

5.17. Using modload on Solaris Systems

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for Solaris. The **dkload** program is not available for Solaris systems, nor is it possible to incorporate AFS during a kernel build.

5.17.1. Requirements for Running modload on Solaris Systems

Before you use **modload**, you must verify that the following are true:

- the **modload** binary resides on the local disk (the standard location is */usr/sbin*).
- the *modload* directory exists in the appropriate place on the local disk (under */usr/afs/bin* on file server machines or */usr/vice/etc* on client machines), and contains *libafs.o* and *libafs.nonfs.o*. If the directory or files do not exist, follow the instructions in section 5.12.

Further, before running **modload** for the first time, you must perform the following steps.

Step 1: Create the file */kernel/fs/afs* as a copy of either *libafs.o* (if the machine runs NFS) or *libafs.nonfs.o* (if the machine does not run NFS). You must use *libafs.o* if the machine is to function as an NFS/AFS Translator machine.

cd /usr/afs/bin/modload on a file server machine

cd /usr/vice/etc/modload on a client machine

cp libafs.o /kernel/fs/afs if the machine supports NFS

cp libafs.nonfs.o /kernel/fs/afs if the machine does not support NFS

Step 2: Create an entry for AFS in the */etc/name_to_sysnum* file, to allow the kernel to make AFS system calls.

In the file */etc/name_to_sysnum*, create an "afs" entry in slot 105 (the slot just before the "nfs" entry), so that the file looks like:

```
reexit          1
fork            2
 .              .
 .              .
 .              .
sigpending      99
setcontext      100
statvfs         103
fstatvfs        104
afs             105
nfs             106
```

5.17.2. Running modload on Solaris Systems

As previously explained, dynamic kernel loaders like **modload** add a library of AFS modifications to the memory version of the kernel without changing the disk version (*/vmunix* or equivalent). Thus **modload** must run each time the machine reboots, if the machine is to remain an AFS machine. Section 5.17.3 explains how to automate dynamic loading for Solaris systems.

To run **modload** manually:

Step 1: Verify that the machine meets the requirements listed in section 5.17.1.

Important Note: The **modload** program must run before the **afsd** program, which starts the Cache Manager. The Cache Manger cannot be operative when you run **modload**.

Step 2: Run **modload**.

```
# /usr/sbin/modload  /kernel/fs/afs
```

If you wish to verify that AFS loaded correctly, use the **modinfo** command.

```
# /usr/sbin/modinfo | egrep afs
```

The following output indicates that AFS loaded successfully:

```
69 fc71f000 4bc15 105   1  afs (afs syscall interface)
69 fc71f000 4bc15  15   1  afs (afs file system)
```

Step 3: Follow the instructions in section 5.17.3 to automate dynamic kernel loading.

Step 4: If you are installing the first AFS machine, return now to section 2.3.6.

If you are installing an additional file server machine of an existing system type, return now to step 3 in section 3.1. If you are installing an additional file server machine of a new system type, return now to step 4 in section 3.2.

If you are installing an additional client machine, return now to section 4.2.5.

5.17.3. Installing the AFS Initialization Script on Solaris Systems

Transarc provides an initialization script for Solaris systems that includes a **modload** instruction for loading AFS as the machine reboots. The script is distributed as */usr/afs/bin/modload/afs.rc* for file server machines and as */usr/vice/etc/modload/afs.rc* for client machines.

To ensure that *afs.rc* is invoked properly at each reboot, you must copy it from its distribution directory to the initialization files directory (standardly, */etc/init.d*), make sure it is executable, and link it to the two locations where **modload** expects to find it.

Step 1: Create a copy of the *afs.rc* script in the */etc/init.d* directory.

```
# cd  /etc/init.d

# cp  /usr/afs/bin/afs.rc  afs.rc for file server machines

# cp  /usr/afs/bin/afs.rc  afs.rc for client machines

# chmod  555  afs.rc

# ln -s afs.rc  /etc/rc3.d/S14afs

# ln -s afs.rc  /etc/rc2.d/K66afs
```

5.18. Using modload on SunOS Systems

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for SunOS systems.

The **dkload** program is also available for these system types; to use it instead of **modload**, follow the instructions in section 5.13.

5.18.1. Requirements for Running modload on SunOS Systems

Before you use **modload**, verify that:

- the **modload** binary resides on the local disk (the standard location is */usr/etc*).
- the *modload* directory exists in the appropriate place on the local disk(under */usr/afs/bin* on file server machines or */usr/vice/etc* on client machines), and contains *libafs.o* and *libafs.nonfs.o*. If the directory or files do not exist, follow the instructions in section 5.12.

5.18.2. Running modload on SunOS Systems

As previously explained, dynamic kernel loaders like **modload** add a library of AFS modifications to the memory version of the kernel without changing the disk version (*/vmunix* or equivalent). Thus **modload** must run each time the machine reboots, if the machine is to remain an AFS machine. You can automate dynamic loading by invoking **modload** in the machine's initialization file (*/etc/rc* or equivalent). Section 5.18.3 explains how.

To run **modload** manually:

Step 1: Verify that the local machine meets the requirements listed in section 5.18.1.

Important Note: The **modload** program must run before the **afsd** program, which starts the Cache Manager. The Cache Manger cannot be operative when you run **modload**.

Step 2: Run **modload**.

On a file server machine:

Note: If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.o* for *libafs.o*. Either use the **mv** command to replace *libafs.o* with *libafs.nonfs.o* in the */usr/afs/bin/modload* directory (before issuing these commands), or make the substitution on the command line.

```
# cd /usr/afs/bin/modload

# /usr/etc/modload  ./libafs.o
```

On a client machine:

Note: If the machine's kernel does not support NFS, you must substitute *libafs.nonfs.o* for *libafs.o*. Either use the **mv** command to replace *libafs.o* with *libafs.nonfs.o* in the */usr/vice/etc/modload* directory (before issuing these commands), or make the substitution on the command line.

```
# cd /usr/vice/etc/modload

# /usr/etc/modload  ./libafs.o
```

Step 3: If the machine's initialization file (*/etc/rc* or equivalent) does not already invoke **modload**, insert the instructions found in section 5.18.3.

Step 4: If you are installing the first AFS machine, return now to section 2.3.7.

If you are installing an additional file server machine of an existing system type, return now to step 3 in section 3.1. If you are installing an additional file server machine of a new system type, return now to step 4 in section 3.2.

If you are installing an additional client machine, return now to section 4.2.6.

5.18.3. Altering the Initialization File to Invoke modload on SunOS Systems

For a file server machine, you can copy in the instructions found in */usr/afs/bin/modload/rc.modload.server*. If you wish to type them in, they are shown below. Place them in the file after the instructions that mount the file systems but before **afsd** is invoked.

Note: If the machine's kernel does not support NFS, substitute *libafs.nonfs.o* for *libafs.o*.

```
if [ -d /usr/afs/bin/modload ]; then
    echo 'Invoking modload...' > /dev/console
    cd /usr/afs/bin/modload
    if [ -f libafs.o ]; then
        echo 'Loading AFS' > /dev/console
        /usr/etc/modload libafs.o 1> /dev/console 2>&1
        if [ $? != 0 ]; then
            echo 'Error loading AFS' > /dev/console
        fi
    else
      echo 'Cannot load AFS--no AFS kernel library' > /dev/console
    fi
    cd /
fi
```

For a client machine, you can copy in the instructions found in */usr/vice/etc/modload/rc.modload.client*. If you wish to type them in, they are shown below. Place them in the file after the instructions that mount the file systems but before **afsd** is invoked.

Note: If the machine's kernel does not support NFS, substitute *libafs.nonfs.o* for *libafs.o*.

```
if [ -d /usr/vice/etc/modload ]; then
    echo 'Invoking modload...' > /dev/console
    cd /usr/vice/etc/modload
    if [ -f libafs.o ]; then
        echo 'Loading AFS' > /dev/console
        /usr/etc/modload libafs.o 1> /dev/console 2>&1
        if [ $? != 0 ]; then
            echo 'Error loading AFS' > /dev/console
        fi
    else
      echo 'Cannot load AFS--no AFS kernel library' > /dev/console
    fi
    cd /
fi
```

6. Installing Sample Package Files

The **package** program aids in the administration of client workstations by verifying the contents of the local disk against a configuration file. In the case of disagreements with the configuration file, **package** makes needed updates to the local disk by copying over files and directories from AFS (as long as the configuration file contains no errors and all of the master copies exist in AFS). It also deletes any files or directories that are not a part of the configuration file. If the kernel on the local disk is updated, **package** forces a reboot of the workstation so the most up-to-date software will run.

This chapter describes how to modify sample **package** files for your environment. However, certain system binaries must reside in AFS before **package** is functional, and the sample **package** files must be modified. Refer to chapter 14 of the *AFS System Administrator's Guide* for detailed information about **package**.

6.1. Loading the Files

The files you need to use **package**, along with sample files, are in the sixth **tar** file on the AFS Binary Distribution tape. See the following instructions to load them onto the local disk using a local or remote tape drive.

In this section you will load the **package** files into an AFS volume you create to house them. The **package** program will access these files in AFS rather than on the local disk, which is possible because **package** runs after **afsd** at each reboot.

6.1.1. Loading the Files into AFS Using a Local Tape Drive

If the local machine has a tape drive, follow these instructions to mount the Binary Distribution tape and unload the **package** files into an AFS volume.

Step 1: Create and mount a volume to house the **package** files. A recommended name is `wsadmin` or ***cellname*`.wsadmin`**. The ACL on the volume's root directory by default gives all access rights to *system:administrators*.

```
# vos create <machine name> <partition name> wsadmin

# cd /afs/.<cellname>

# fs mkmount wsadmin wsadmin

# vos release root.cell

# fs checkvolumes
```

Step 2: Mount the AFS Binary distribution tape and load the files from the sixth **tar** set into */afs/**cellname**/wsadmin*.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# cd wsadmin

# mt -f /dev/<device> rewind

# mt -f /dev/<device> fsf 5

# tar xvf /dev/<device>
```

6.1.2. Loading Files into AFS from */usr/wsadmin*

If the local machine does not have a tape drive, you must load the **package** files into */usr/wsadmin* on a remote machine and from there into the AFS volume.

Step 1: Create a */usr/wsadmin* directory on the remote machine, and load the **package** files into it.

> **Note:** If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.
>
> For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.
>
> # **mkdir /usr/wsadmin**
>
> # **cd /usr/wsadmin**
>
> # **mt -f /dev/**<*device*> **rewind**
>
> # **mt -f /dev/**<*device*> **fsf 5**
>
> # **tar xvf /dev/**<*device*>

Step 2: Create and mount a volume to house the **package** files. A recommended name is `wsadmin` or ***cellname***`.wsadmin`. The ACL on the volume's root directory by default gives all access rights to *system:administrators*.

> # **vos create** <*machine name*> <*partition name*> **wsadmin**
>
> # **cd /afs/.**<*cellname*>
>
> # **fs mkmount wsadmin wsadmin**
>
> # **vos release root.cell**
>
> # **fs checkvolumes**

Step 3: Copy the files from */usr/wsadmin* to */afs/**cellname**/wsadmin* using **cp**, **ftp**, NFS, or another network file transfer utility.

6.2. Using the Sample Package Files

The */afs/**cellname**/wsadmin/etc* directory contains **package** configuration files that were built from the prototype files in *src* and the libraries in *lib*. You never need to directly edit any of the files in this directory. These files are modified by running **make**.

The */afs/**cellname**/wsadmin/lib* directory contains the package library files referred to by the prototype files. Files specific to a machine type are called ***sysname**.**type***.

The */afs/**cellname**/wsadmin/src* directory contains the prototype files used to build configurations. These prototype files include library files from the libraries in */afs/**cellname**/wsadmin/lib*.

The *minimal.** files are the minimum needed to keep the most recent version of AFS on the system. These are built from */afs/**cellname**/src/minimal.proto*, which is built from (among other files) *lib/rt_aix221.AFS* or *lib/43.AFS*. You should use the minimal files, and build upon them. The *staff.** files are examples of more detailed **package** files.

Follow these steps to change the minimal files for your system:

Step 1: Edit */afs/**cellname**/wsadmin/lib/base.generic* to contain your cellname.

Step 2: Edit */afs/**cellname**/wsadmin/lib/*.readonly* to contain your cellname and point to your compilation directories. For example, in the Transarc tree there are three different destinations: */usr/afs.rel*, */usr/afs.test*, and */usr/afs.dev*. You will probably want to set all of these directories to */usr/afsws*.

Step 3: Verify that */etc/package* exists on each machine's local disk. If it does not, install a copy from AFS and add execute permission.

```
# cp  /afs/<cellname>/<sysname>/usr/afsws/etc/package  /etc/package

# chmod +x /etc/package
```

Step 4: Create the package configuration file on each machine.

```
# echo  "/afs/<cellname>/wsadmin/etc/minimal"  >  /.package
```

Step 5: Add commands to the machine's initialization file (*/etc/rc* or equivalent) to invoke **package**.

Add the following lines to the initialization file (such as */etc/rc*) after the lines that invoke **afsd**.

```
if [ -f /etc/package ]; then
        if [ -f /.package ]; then
                /etc/package -v -c ‘cat /.package‘ >
                    /dev/console
           else
                /etc/package -v >/dev/console
        fi
case ? in
0)
        echo "Package completed successfully" >
                                        /dev/console 2>&1
        date >/dev/console 2>&1
        ;;
4)
        echo "Rebooting to restart system" >
                                       /dev/console 2>&1
        echo >/fastboot
        /etc/reboot
        ;;
*)
        echo "Update failed, continuing anyway" >
                                       /dev/console 2>&1
        ;;
esac
fi
```

Step 6: Build the new configuration files so your changes are incorporated into the **package** files in */afs/**cellname**/wsadmin/etc*, after verifying that the current path environment includes needed definitions.

Before running the following commands, verify that */usr/afsws/bin* and */usr/afsws/etc* are included in the current path environment definition. Add them if necessary; errors will result if these directories are not defined.

cd /afs/<*cellname*>**/wsadmin/src**

make system wsadmin=/afs/<*cellname*>**/wsadmin**

7. Building from Source

This chapter describes how to build AFS from source.

Note for sites using the NFS/AFS Translator: 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.

7.1. Loading the Source

Working on the AFS machine where you will build the source, follow these steps to load the AFS source tree from the AFS Source Distribution tape.

Step 1: Create and mount the volume for housing the AFS source tree.

```
# vos create <machine name> <partition name> src.afs

# cd /afs/.<cellname>

# mkdir afs

# fs mkmount afs/src src.afs

# vos release root.cell

# fs checkvolumes
```

Step 2: Load the source tree to the local disk.

Note: If the tape drive is attached to an AIX system, use **tctl** instead of **mt**. If the system is an RS/6000, further verify that block size is set to 0 (meaning variable block size); if necessary, use **SMIT** to set block size to 0 before reading the tape.

For <*device*>, substitute the name of the tape device for your system that does not rewind after each operation.

```
# mt -f /dev/<device> rewind

# cd /afs/<cellname>/afs/src

# tar xvf /dev/<device>
```

7.2. Setting Up the Tree Structure and Tools Using washtool

Transarc provides the **washtool** program for managing a hierarchy of software development projects. The program builds project trees which allow program editing, compilation, and installation. Using **washtool**, follow these steps to set up the object and destination trees.

Note: The **washtool** program assumes that you use RCS. If you do not, or do not want to use **washtool**, see section 7.4.

Step 1: Create an *rcs* directory under */afs/**cellname**/afs* (at the same level as the *src* directory). Then create directories under it for the AFS packages. Transarc distributes source directories that already have links in the source package files from RCS to the RCS package files.

```
# mkdir /afs/<cellname>/afs/rcs

# cd /afs/<cellname>/afs/src

# foreach dir (*)
? if (-d $dir) then
? echo $dir
? mkdir ../rcs/$dir
? mkdir ../rcs/$dir/RCS
? endif
? end
```

Step 2: Create a directory under */afs/**cellname**/afs* for each system type for which you will build AFS. In order for AFS's *@sys* mechanism to work, you must use Transarc's recommended system names, as listed in section 1.3.

Note: Repeat the second command for each system type.

```
# cd /afs/<cellname>/afs

# mkdir <sysname>
```

For example, to create directories for DECStations running Ultrix 4.3 and for IBM RS/6000s running AIX 3.2, issue the following commands.

```
# mkdir pmax_ul43
# mkdir rs_aix32
```

Step 3: Define two symbolic links in */afs/**cellname**/afs*:

- *dest*, pointing to *@sys/dest*
- *obj*, pointing to *@sys/obj*

```
# cd /afs/<cellname>/afs
# ln -s @sys/obj obj
# ln -s @sys/dest dest
```

Step 4: **(Optional.)** The build procedure normally creates a destination directory for each system type under */afs/**cellname**/afs/**sysname**/dest*, and installs its results there. If you want to locate a system type's destination directory elsewhere, then create a link from the *dest* directory to that alternate directory.

Note: Perform this step only if you want to locate a system type's *dest* directory somewhere other than */afs/**cellname**/afs/**sysname**/dest*. Repeat it for each desired system type.

```
# cd /afs/<cellname>/afs/<sysname>
# ln -s <full path of alternate directory> dest
```

For example, if you wanted the destination directory for DECStations running Ultrix 4.1 to be */afs/**cellname**/pmax_ul43/usr/afsws*, you would issue the following.

```
# cd /afs/<cellname>/afs/pmax_ul43
# ln -s /afs/<cellname>/pmax_ul43/usr/afsws dest
```

Step 5: Create a link called *PARENT* in the top level directory (*/afs/**cellname**/afs*).

```
# cd /afs/<cellname>/afs
# ln -s . PARENT
```

Step 6: Create a link in */afs/**cellname**/afs* that points to the global *Makefile* in the *src* directory.

```
# cd /afs/<cellname>/afs

# ln -s src/Makefile Makefile
```

The following is an example of what your */afs/**cellname**/afs* might look like after you have completed this step. It shows */afs/transarc.com/afs.test*.

```
lrwxr-xr-x hines  12 Sep 19 11:02 Makefile->src/Makefile
lrwxr-xr-x hines   1 Sep 19 11:03 PARENT -> .
drwxrwxrwx root 2048 Sep 28 18:57 rs_aix32
lrwxr-xr-x hines   9 Sep 19 11:25 dest -> @sys/dest
lrwxr-xr-x hines   8 Sep 19 11:25 obj -> @sys/obj
drwxrwxrwx root 2048 Sep 28 19:00 pmax_ul43
drwxrwxrwx root 4096 Sep 18 21:01 rcs
drwxrwxrwx root 2048 Sep 28 19:04 rt_aos4
drwxrwxrwx root 2048 Sep 20 14:58 src
drwxrwxrwx root 2048 Sep 28 16:55 sun3_411
drwxrwxrwx root 2048 Sep 28 19:05 vax_ul43
```

Step 7: Verify that */usr/afsws/bin/co* and */usr/afsws/bin/ci* exist (they are distributed as part of the AFS Binary Distribution). They are programs that check the times listed in the *.rcstimes* files in each package directory, making sure that the build procedure checks out a new copy of the source files when necessary.

Step 8: Verify that the **washtool** and **rcsupdate** binaries are available. The *Makefile* looks for them in */afs/**cellname**/afs/dest/bin* by default, but you can specify a different location by setting the `WASHTOOL` variable on the **make** command line (see section 7.3 for an example).

7.3. Building the System

This section explains how to build the system. It assumes you have followed the instructions in section 7.2.

Step 1: Working in the */afs/**cellname**/afs* directory, issue the **make finale** command to build the system.

> # **cd /afs/**<*cellname*>**/afs**
>
> If the **washtool** binary is in the expected directory (*/afs/**cellname**/afs/dest/bin*), then issue:
>
> # **make finale**
>
> If the **washtool** binary is in a different directory, then issue the following, specifying the full path of the alternate directory:
>
> # **make finale WASHTOOL=**<*alternate washtool directory*>

For example, if **washtool** is in */usr/afsws/bin*, then you would issue:

make finale WASHTOOL=/usr/afsws/bin/washtool

7.4. Building Without Washtool

This section explains how to build AFS without using **washtool**. It assumes that you have followed the instructions in section 7.1, but not in section 7.2. Use these instructions if you do not use RCS for the AFS source files.

Step 1: Create an object directory for each system type you will build.

> **Note:** Repeat these commands for each system type.
>
> # **mkdir /afs/**<*cellname*>**/afs/**<*sysname*>
>
> # **mkdir /afs/**<*cellname*>**/afs/**<*sysname*>**/obj**

Step 2: Copy the source files from */afs/**cellname**/afs/src* to each system type's *obj* directory.

> **Note:** Repeat these commands for each system type you will build.
>
> # **cd /afs/**<*cellname*>**/afs/**<*sysname*>**/obj**
>
> # **up -v /afs/**<*cellname*>**/afs/src /afs/**<*cellname*>**/afs/**<*sysname*>**/obj**

Step 3: Create a destination directory for each system type.

> **Note:** Repeat this command for each system type.
>
> # **mkdir /afs/**<*cellname*>/<*sysname*>**/dest**

Step 4: For each AFS package found under */afs/**cellname**/afs/**sysname**/obj*, create a link from its *DEST* subdirectory to the destination tree. Repeat for each system type.

> **Note:** Repeat this command for each system type ***sysname*** and each AFS package ***AFS-pkg***. Type it on a single line in each case.
>
> # **ln -s /afs/**<*cellname*>/<*sysname*>**/dest**
> **/afs/**<*cellname*>**/afs/**<*sysname*>**/obj/**<*AFS-pkg*>**/DEST**

Step 5: Working in the */afs/**cellname**/afs* directory, run the **make** program for each system type.

Note: Repeat the second command for each system type, substituting its Transarc system name for ***sysname***.

cd /afs/<*cellname*>**/afs**

make SYS_NAME=<*sysname*>

Table of Contents

1. Installation Overview 1-1

1.1. Overview 1-2
1.1.1. Initial Tasks 1-2
1.1.2. As-needed Tasks 1-2
1.2. Requirements 1-4
1.2.1. Login Identity 1-4
1.2.2. Recommended Reading 1-4
1.2.3. General Requirements 1-5
1.2.4. File Server Machine Requirements 1-5
1.2.5. Client Machine Requirements 1-6
1.3. Supported System Types 1-7
1.3.1. About Upgrading to a New Operating System 1-8
1.4. The AFS Binary Distribution 1-9
1.4.1. The Contents of the Binary Distribution Tape 1-9
1.4.2. Unloading Files from the Binary Distribution Tape 1-10
1.5. How to Continue 1-12

2. Installing the First AFS Machine 2-1

Assumptions 2-1
Configuration Decisions 2-1
How to Use This Chapter 2-2
2.1. Overview: Installing File Server Functionality 2-3
2.2. Choosing the First AFS Machine 2-4
2.3. Incorporating AFS Modifications into the Kernel 2-5
2.3.1. Loading Dynamic Kernel Loader Files to the Local Disk 2-7
2.3.2. Using Transarc's Dynamic Kernel Loader, dkload 2-9
2.3.3. Using the Kernel Extension Facility on AIX 3.2 Systems 2-11
2.3.4. Using ml on IRIX Systems 2-13
2.3.5. Using kl_util on NeXT OS Systems 2-15
2.3.6. Using modload on Solaris Systems 2-16
2.3.7. Using modload on SunOS Systems 2-18
2.3.8. Installing a Previously-Built AFS-Modified Kernel 2-19
2.4. Loading the File Server Process Binaries 2-20
2.4.1. Loading Files From a Local Tape Drive 2-20
2.4.2. Loading Files From a Remote Machine 2-21
2.5. Setting Up AFS Partitions 2-22
2.6. Replacing fsck 2-25
2.6.1. Replacing fsck on AIX 3.2 Systems 2-25
2.6.2. Replacing fsck on DECstations 2-26
2.6.3. Replacing fsck on IRIX Systems 2-26
2.6.4. Replacing fsck on NeXT OS Systems 2-26
2.6.5. Replacing fsck on Solaris Systems 2-27
2.6.6. Replacing fsck on SunOS Systems 2-27
2.6.7. Replacing fsck on Other System Types 2-27
2.7. Starting the BOS Server 2-28
2.8. Defining the Cell Name and the Machine's Cell Membership 2-30
2.9. Starting the Authentication Server 2-31

2.9.1. A Note on Kerberos 2-31
2.9.2. Instructions for Installing the Authentication Server 2-31
2.10. Initializing Security Mechanisms 2-32
2.11. Starting the Protection Server 2-35
2.12. Starting the Volume Location Server 2-36
2.13. Starting the Backup Server 2-37
2.14. Starting the File Server, Volume Server, and Salvager 2-38
2.15. Starting the Server Portion of the Update Server 2-40
2.16. Starting the Controller for NTPD 2-42
2.17. Completing the Installation of Server Functionality 2-43
2.18. Overview: Installing Client Functionality 2-45
2.19. Loading Client Files on the First AFS Machine 2-46
2.19.1. Loading Client Files Using a Local Tape Drive 2-46
2.19.2. Loading Client Files from a Remote Machine 2-47
2.20. Defining the Client Machine's Cell Membership 2-49
2.21. Creating the Client Version of CellServDB 2-50
2.22. Setting Up the Cache 2-52
2.22.1. Setting Up a Disk Cache 2-53
2.22.2. Setting Up a Memory Cache 2-53
2.23. Creating */afs* and Starting the Cache Manager 2-54
2.24. Overview: Completing the Installation of the First AFS Machine 2-58
2.25. Setting Up the Top Levels of the AFS Tree 2-59
2.26. Turning On Authorization Checking 2-62
2.27. Setting up Volumes to House AFS Binaries 2-63
2.27.1. Loading AFS Binaries into a Volume and Creating a Link to the Local Disk 2-65
2.28. Storing System Binaries in AFS 2-67
2.28.1. Setting the ACL on System Binary Volumes 2-67
2.28.2. Volume and Directory Naming Scheme 2-67
2.29. Enabling Access to Transarc and Other Cells 2-69
2.30. Enabling Access to New Cells in the Future 2-70
2.31. Improving Your Cell's Security 2-73
2.31.1. Controlling root Access 2-73
2.31.2. Controlling System Administrator Access 2-73
2.31.3. Protecting Sensitive AFS Directories 2-73
2.32. Replacing login 2-75
2.32.1. Replacing login for AIX 3.2 2-75
2.32.2. Replacing login for IRIX Systems 2-76
2.32.3. Replacing the Login Window binary for NeXT OS Systems 2-76
2.32.4. Replacing login for Other System Types 2-77
2.33. Altering File System Clean-Up Scripts 2-78
2.34. Removing Client Functionality 2-79

3. Installing Additional Servers 3-1

Assumptions 3-1
3.1. Installing a File Server Machine of an Existing System Type 3-2
3.1.1. Procedure Overview 3-2
3.1.2. Instructions 3-3
3.2. Installing a File Server Machine of a New System Type 3-17
3.2.1. Procedure Overview 3-17
3.2.2. Instructions 3-18
3.3. Installing Database Server Functionality on an Existing File Server Machine 3-33
3.3.1. Procedure Overview 3-34

3.3.2. Instructions 3-35
3.4. Removing Database Server Functionality 3-38
3.4.1. Procedure Overview 3-38
3.4.2. Instructions 3-38

4. Installing Additional Client Machines 4-1

Summary of Procedures 4-1
4.1. Loading Client Files onto the Local Disk 4-2
4.1.1. Loading Files for a Machine of an Existing System Type 4-3
4.1.2. Loading Files for a Machine of a New System Type 4-4
4.2. Incorporating AFS Modifications into the Kernel 4-6
4.2.1. Using Transarc's Dynamic Kernel Loader, dkload 4-7
4.2.2. Using the Kernel Extension Facility on AIX 3.2 Systems 4-9
4.2.3. Using ml on IRIX Systems 4-11
4.2.4. Using kl_util on NeXT OS Systems 4-13
4.2.5. Using modload on Solaris Systems 4-14
4.2.6. Using modload on SunOS Systems 4-16
4.2.7. Installing a New Kernel on an Existing System Type 4-17
4.2.8. Building a New Kernel on a New System Type 4-18
4.3. Defining the Machine's Cell Membership and Creating *CellServDB* 4-19
4.4. Setting Up the Cache 4-20
4.4.1. Setting Up a Disk Cache 4-21
4.4.2. Setting Up a Memory Cache 4-21
4.5. Creating */afs* and Starting the Cache Manager 4-22
4.6. Setting Up Volumes and Loading Binaries into AFS 4-25
4.6.1. Linking */usr/afsws* on an Existing System Type 4-25
4.6.2. Creating Binary Volumes for a New System Type 4-25
4.7. Replacing login 4-29
4.7.1. Replacing login for AIX 3.2 4-29
4.7.2. Replacing login for IRIX Systems 4-30
4.7.3. Replacing the Login Window binary for NeXT OS 4-30
4.7.4. Replacing login for Other System Types 4-31
4.8. Altering File System Clean-Up Scripts 4-32

5. Incorporating AFS Into the Kernel 5-1

5.1. Overview: Building a New Kernel 5-3
5.1.1. Assumptions 5-3
5.2. Loading the Files Needed for a Kernel Build 5-4
5.2.1. Loading Files From a Local Tape Drive 5-4
5.2.2. Loading Files From a Remote Machine 5-5
5.3. Building Kernels on AIX 2.2.1 Systems 5-6
5.4. Building Kernels on AOS Systems 5-10
5.5. Building Kernels on HP 9000 Series 300, 400, and 700 Systems Running HP-UX 9.0 5-15
5.6. Building Kernels on HP 9000 Series 800 Systems Running HP-UX 9.0 5-17
5.7. Building Kernels on IRIX Systems 5-19
5.8. Building Kernels on SunOS Systems 5-21
5.9. Building Kernels on DECstations Running Ultrix 5-25
5.10. Building Kernels on VAXstations Running Ultrix 5-30
5.11. Overview: Using Dynamic Kernel Loaders 5-35
5.12. Loading Dynamic Kernel Loader Files 5-36
5.12.1. Loading Dynamic Kernel Loader Files Using a Local Tape Drive 5-36

5.12.2. Loading Dynamic Kernel Loader Files From a Remote Machine 5-38
5.13. Using dkload 5-39
5.13.1. Requirements for Running dkload 5-39
5.13.2. Running dkload 5-40
5.13.3. Altering the Initialization File to Invoke dkload 5-42
5.14. Using the Kernel Extension Facility on AIX 3.2 Systems 5-43
5.14.1. Requirements for Running the Kernel Extension Facility 5-43
5.14.2. Running the Kernel Extension Facility 5-44
5.14.3. Altering the Initialization File to Invoke the Kernel Extension Facility 5-46
5.15. Using ml on IRIX Systems 5-47
5.15.1. Requirements for Running ml 5-47
5.15.2. Running ml 5-48
5.15.3. Installing the AFS Initialization Script on IRIX Systems 5-49
5.16. Using kl_util on NeXT OS Systems 5-50
5.16.1. Requirements for Running kl_util 5-50
5.16.2. Running kl_util 5-51
5.17. Using modload on Solaris Systems 5-52
5.17.1. Requirements for Running modload on Solaris Systems 5-52
5.17.2. Running modload on Solaris Systems 5-53
5.17.3. Installing the AFS Initialization Script on Solaris Systems 5-54
5.18. Using modload on SunOS Systems 5-55
5.18.1. Requirements for Running modload on SunOS Systems 5-55
5.18.2. Running modload on SunOS Systems 5-56
5.18.3. Altering the Initialization File to Invoke modload on SunOS Systems 5-57

6. Installing Sample Package Files 6-1

6.1. Loading the Files 6-2
6.1.1. Loading the Files into AFS Using a Local Tape Drive 6-3
6.1.2. Loading Files into AFS from */usr/wsadmin* 6-4
6.2. Using the Sample Package Files 6-5

7. Building from Source 7-1

7.1. Loading the Source 7-2
7.2. Setting Up the Tree Structure and Tools Using washtool 7-3
7.3. Building the System 7-6
7.4. Building Without Washtool 7-7