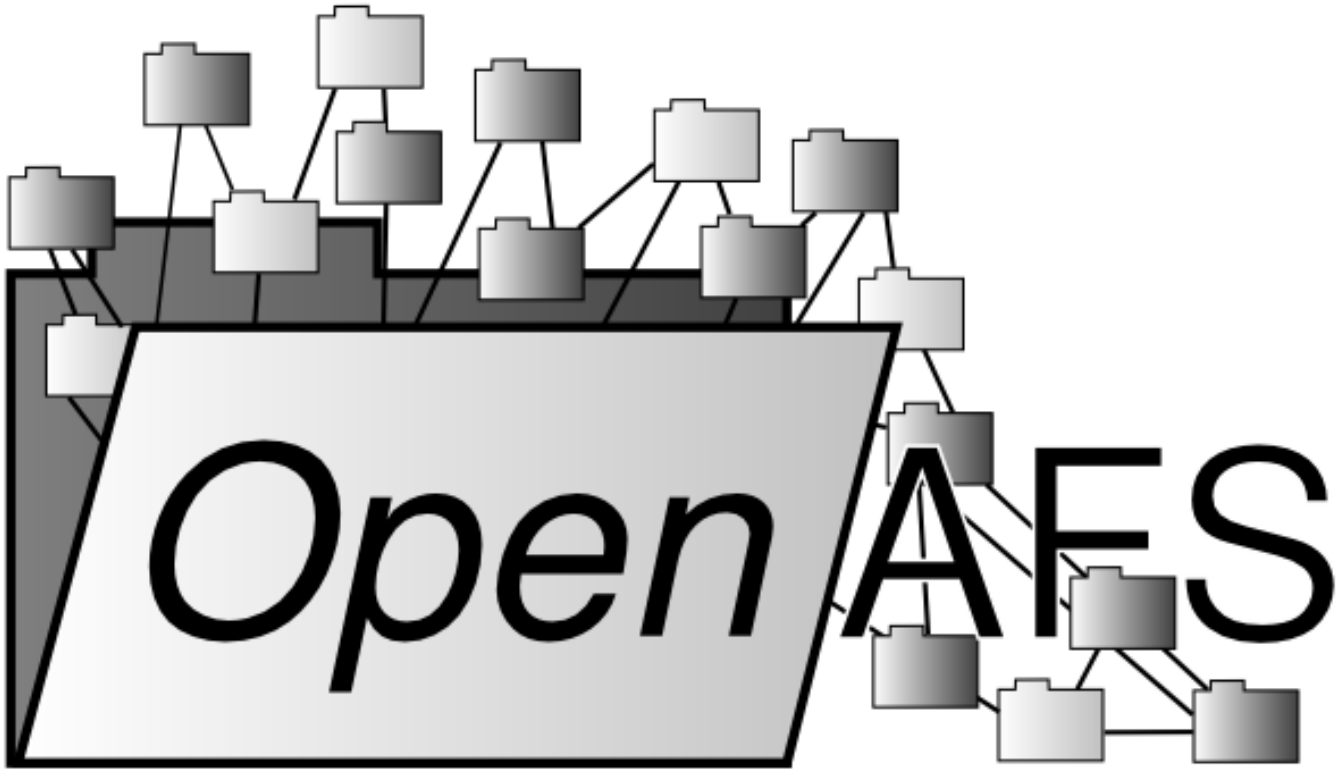

Quick Start Guide for UNIX

OpenAFS Quick Start Guide for UNIX

COLLABORATORS			
	TITLE : OpenAFS Quick Start Guide for UNIX		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 22, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1.8.9-1-debian			

Contents

Abstract

This document describes the initial setup of an OpenAFS cell and an OpenAFS client. It is currently being updated for OpenAFS 1.6.10 and is still dated and incorrect in many details. This edition applies to OpenAFS for UNIX, Version 1.4.10, and to all subsequent releases and modifications until otherwise indicated in new editions.

About This Guide

This section describes the purpose, organization, and conventions of this document.

Audience and Purpose

This guide explains how to install and configure OpenAFS server and client machines. It assumes that the reader is familiar with UNIX® system administration, but not AFS.

The instructions explain how to issue AFS® commands in the context of specific tasks, but do not describe a command's function or arguments in detail. Refer to the *OpenAFS Administration Reference* as necessary.

Organization of the Document

See The Procedures Described in this Guide.

How to Use This Document

See The Procedures Described in this Guide and How to Continue.

Related Documents

The OpenAFS documentation set also includes the following documents. Large portions of them are as released to the community by IBM, and are not directly applicable to current releases of OpenAFS. This document uses the legacy "Transarc" paths (`/usr/afs`, `/usr/vice`, etc.), which do not correspond to the normal file system hierarchy on many modern machines; the paths may need to be substituted according to the local software installation.

OpenAFS Administration Guide This guide describes the concepts and procedures that a system administrator must know to manage an AFS cell. It assumes familiarity with UNIX, but requires no previous knowledge of AFS.

The first chapters of the *OpenAFS Administration Guide* present basic concepts and guidelines. Understanding them is crucial to successful administration of an AFS cell. The remaining chapters in the guide provide step-by-step instructions for specific administrative tasks, along with discussions of the concepts important to that particular task.

OpenAFS Administration Reference This reference manual details the syntax and effect of each AFS command. It is intended for the experienced AFS administrator, programmer, or user.

The *OpenAFS Administration Reference* lists AFS files and commands in alphabetical order. The reference page for each command specifies its syntax, including the acceptable aliases and abbreviations. It then describes the command's function, arguments, and output if any. Examples and a list of related commands are provided, as are warnings where appropriate.

This manual complements the *OpenAFS Administration Guide*: it does not include procedural information, but describes commands in more detail than the *OpenAFS Administration Guide*.

OpenAFS User Guide This guide presents the basic concepts and procedures necessary for using AFS effectively. It assumes that the reader has some experience with UNIX, but does not require familiarity with networking or AFS.

The guide explains how to perform basic functions, including authenticating, changing a password, protecting AFS data, creating groups, and troubleshooting. It provides illustrative examples for each function and describes some of the differences between the UNIX file system and AFS.

OpenAFS Release Notes This document provides information specific to each release of AFS, such as a list of new features and commands, a list of requirements and limitations, and instructions for upgrading server and client machines.

Typographical Conventions

This document uses the following typographical conventions:

- Command and option names appear in **bold type** in syntax definitions, examples, and running text. Names of directories, files, machines, partitions, volumes, and users also appear in **bold type**.
- Variable information appears in *italic type*. This includes user-supplied information on command lines and the parts of prompts that differ depending on who issues the command. New terms also appear in *italic type*.
- Examples of screen output and file contents appear in `monospace type`.

In addition, the following symbols appear in command syntax definitions, both in the documentation and in AFS online help statements. When issuing a command, do not type these symbols.

- Square brackets **[ ]** surround optional items.
- Angle brackets **< >** surround user-supplied values in AFS commands.
- A superscripted plus sign **+** follows an argument that accepts more than one value.
- The percent sign `%` represents the regular command shell prompt. Some operating systems possibly use a different character for this prompt.
- The number sign `#` represents the command shell prompt for the local superuser **root**. Some operating systems possibly use a different character for this prompt.
- The pipe symbol **|** in a command syntax statement separates mutually exclusive values for an argument.

For additional information on AFS commands, including a description of command string components, acceptable abbreviations and aliases, and how to get online help for commands, see the appendix to the *OpenAFS Administration Guide*.

Chapter 1

Installation Overview

This chapter describes the type of instructions provided in this guide and the hardware and software requirements for installing AFS®.

Before beginning the installation of your cell's first machine, read this chapter and the material from the *OpenAFS Administration Guide* listed in Recommended Reading List. It is also best to read the entirety of certain sections of this document, in particular Installing the First AFS Machine, before beginning the installation, so that you understand the overall scope of the installation procedure. Similarly, before installing additional server or client machines it is best to read through Installing Additional Server Machines and Installing Additional Client Machines.

If you are already running a version of AFS, consult the upgrade instructions in the *OpenAFS Release Notes* before proceeding with the installation.

If you are working with an existing cell that uses **kaserver** or external Kerberos v4 for authentication, please see the notes in kaserver and legacy Kerberos 5 authentication and the rest of Appendix B for how the installation steps will differ from those described in the rest of this guide. Do not use the **kaserver** for new deployments of AFS; it uses extremely insecure cryptography.

1.1 The Procedures Described in this Guide

This guide describes two types of installation procedures: initial procedures (such as installing the first AFS machine or incorporating AFS into the kernel) and as-needed procedures (such as installing additional server machines or client machines).

1.1.1 Required Initial Procedures

You must perform the following basic procedures to start using AFS.

1.1.1.1 Incorporating AFS Into the Kernel

You must incorporate AFS modifications into the kernel of every client machine. Depending on the operating system, you either use a program for dynamic kernel loading, build a new static kernel, or can choose between the two. For your convenience, the instructions for incorporating AFS into the kernel appear in full in every chapter where you need to use them.

1.1.1.2 Installing the First AFS Machine

You install the first AFS machine in your cell to function as both an AFS server and client machine. You can disable the client functionality after completing the installation, if you wish.

The first server machine in a cell performs several functions:

- It acts as the first *database server machine*, running the server processes that maintain the AFS administrative databases

- It may act as the *system control machine*, distributing certain configuration files to the other server machines in the cell
- It may act as the *binary distribution machine* for its system type, distributing AFS binaries to other server machines of its system type

The latter two functions are performed by the Update Server, which is considered to be deprecated and may be removed in a future release.

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

1.1.2 As-needed Procedures

1.1.2.1 Upgrading the Operating System

Upgrading the operating system requires you to take several steps to protect data and AFS-modified binaries from being lost or overwritten. For guidelines, see About Upgrading the Operating System.

1.1.2.2 Installing Additional File Server Machines

See Installing an Additional File Server Machine.

1.1.2.3 Configuring or Decommissioning Database Server Machines

See Installing Database Server Functionality and Removing Database Server Functionality.

1.1.2.4 Installing Additional AFS Client Machines

See Installing Additional Client Machines.

1.1.2.5 Building AFS from Source Code

See Appendix A, Building AFS from Source Code

1.1.2.6 Configuring Legacy Components

See Appendix B, Configuring Legacy Components

1.2 Recommended Reading List

To develop the best understanding of the overall scope of an installation procedure, read through the entire chapter or section that describes it before performing any actions.

In addition, familiarity with some basic AFS concepts can make the installation more efficient, because you understand better the purpose of the steps. The following is a prioritized list of material to read before installing the first AFS machine. At minimum, read the first chapter of the *OpenAFS Administration Guide*. Then continue your reading in the indicated order, as extensively as you can. It is more important at this point to read the conceptual material in each section than the instructions.

Selected Topics in the *OpenAFS Administration Guide*

- The chapter titled *An Overview of AFS Administration*

- Selected sections in the *Administering Server Machines* chapter: *Local Disk Files on a Server Machine*, *The Four Roles for a Server Machine*, *Maintaining the Server CellServDB File*
- Selected sections in the *Monitoring and Controlling Server Processes* chapter: *Controlling and Checking Process Status*
- Selected sections in the *Managing Server Encryption Keys* chapter: *About Server Encryption Keys*
- Selected sections in the *Managing Volumes* chapter: *About Volumes*, *Creating Read/write Volumes*, *Clones and Cloning*, *Mounting Volumes*
- Selected sections in the *Administering Client Machines and the Cache Manager* chapter: *Overview of Cache Manager Customization*, *Configuration and Cache-related Files on the Local Disk*, *Determining the Cache Type, Size, and Location*
- Selected sections in the *Managing Access Control Lists* chapter: *Protecting Data in AFS*

More Selected Topics in the *OpenAFS Administration Guide*

- Selected sections in the *Managing Volumes* chapter: *Creating and Releasing Read-only Volumes (Replication)*, *Creating Backup Volumes*
- Selected sections in the *Administering the Protection Database* chapter: *About the Protection Database*
- Selected sections in the *Administering User Accounts* chapter: *The Components of an AFS User Account*
- Selected sections in the *Managing Administrative Privilege* chapter: *An Overview of Administrative Privilege*

1.3 Requirements

You must comply with the following requirements to install AFS successfully.

1.3.1 Login Identity

Log into the machine you are installing as the local superuser **root**. When instructed, also authenticate to AFS using Kerberos as the administrative user **admin**.

1.3.2 General Requirements

- You must have a Kerberos 5 realm running for your site, and the ability to create new principals within that realm. If you are working with an existing cell using the deprecated *kaserver* or Kerberos v4 authentication, please see kaserver and legacy Kerberos 4 authentication for modifications to the following instructions.
- You must have a NTP, or similar, timeservice running. Each AFS machine should derive its system time from this timeservice. If you are working with an existing cell, and wish to use AFS's internal time service, please see Appendix B for modifications to the following instructions.
- You must have an OpenAFS Binary Distribution for each system type you are installing, or have built a binary from the supplied source code. Unless otherwise noted, the Binary Distribution includes software for both client and server machines.
- All AFS machines that belong to a cell must be able to access each other via the network.
- The machine must be running the standard, vendor-supplied version of the operating system supported by the current version of AFS. The operating system must already be installed on the machine's root partition.
- You must be familiar with the current operating system and disk configuration of the machine you are installing.
- All hardware and non-AFS software on the machine must be functioning normally.
- No critical processes can be running on the machine you are installing, because you may need to reboot it during the installation.

1.3.3 File Server Machine Requirements

- Cell configuration is simplest if the first machine you install has the lowest IP address of any database server machine you currently plan to install. If you later configure a machine with a lower IP address as a database server machine, you must update the **/usr/vice/etc/CellServDB** file on all of your cell's client machines before the installation. For further discussion, see Installing Database Server Functionality.
- The partition mounted on the **/usr** directory must have a sufficient amount of space to hold the AFS binaries that will be used; a few hundred MB should be more than sufficient.

  More significant amounts of space on the partition are required by the administrative databases stored in the **/usr/afs/db** directory and the server process log files stored in the **/usr/afs/logs** directory. The exact requirement depends on many factors, such as the size of your cell and how often you truncate the log files.
- There should be at least one partition (or logical volume, if the operating system and AFS support them) dedicated exclusively to storing AFS volumes. Special configuration is required to use non-dedicated partitions as the backing store for AFS file data. The total number and size of server partitions on all file server machines in the cell determines how much space is available for AFS files.

1.3.4 Client Machine Requirements

- The partition mounted on the **/usr** directory must have a sufficient amount of disk space to store the AFS binaries that will be used; a few hundred MB should be more than sufficient.
- On a client machine that uses a disk cache, there must be enough free space on the cache partition (by convention, mounted on the **/usr/vice/cache** directory) to accommodate the cache. The minimum recommended cache size is 50 MB, but larger caches generally perform better. It is recommended to have a dedicated partition for this cache, as the client does not degrade gracefully when the partition containing the cache is filled by other processes.
- On a client machine that uses a memory cache, there must be at least 50 MB of machine memory to devote to caching, but again more memory generally leads to better performance. For further discussion, see the sections in Installing Additional Client Machines about configuring the cache.

1.4 Supported System Types

The *OpenAFS Release Notes* for each AFS release list the supported system types. Support for subsequent revisions of an operating system often becomes available between AFS releases. The OpenAFS mailing lists can provide information regarding this interim support

It is the goal of OpenAFS to support AFS on a wide range of popular system types. Furthermore, each time an operating system vendor releases a new general availability version of a supported operating system, it is a goal to support AFS on it within a short time. Support can be delayed a bit longer if it is necessary to generate completely new binaries.

It is not always possible to support AFS on every intermediate version of an operating system or for certain processor types. In some cases, platform limitations make certain AFS functionality (such as file server or NFS/AFS translator functionality) unavailable on one or more platforms. For a list of limitations, see the *OpenAFS Release Notes* or ask on the OpenAFS mailing lists.

1.5 About Upgrading the Operating System

On most modern systems, using Kerberos 5 for authentication and the namei fileserver backend, no particular precautions need to be taken across operating system upgrades. Legacy confiruations involving kaserver authentication or inode fileserver backends will need to undertake the following precautions.

These actions include, but are not necessarily limited to, the following.

- On platforms running the inode fileserver, unmount the AFS server partitions (mounted at **/vicep***xx* directories) 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, it is prudent to comment out commands in the machine's initialization file that remount the server partitions, to prevent them from being remounted until you can replace the standard **fsck** program with the AFS-modified version. The instructions in this guide for installing AFS server machines explain how to replace the **fsck** program. If you are unsure if your platform uses the inode fileserver, it is worth following this advice for all platforms.
- Protect the AFS-modified versions of commands and configuration files from being overwritten by vendor-supplied versions. These include **vfsck** (the AFS version of **fsck**), and configuration files such as the one for the Pluggable Authentication Module (PAM). After you have successfully installed the operating system, remember to move the AFS-modified commands and files back to the locations where they are accessed during normal functioning.

1.6 The OpenAFS Binary Distribution

Binary Distributions for supported systems may be downloaded from the OpenAFS website. The distributions are in the native packaging format for the system in question, and should generally be installed using your system's package management tools.

For those distributions provided as tar files, or those built from source, the instructions in this guide specify how to copy out both binaries and configuration files

1.7 How to Continue

If you are installing the first AFS machine in your cell, proceed to Installing the First AFS Machine.

If you are installing an additional file server machine, or configuring or decommissioning a database server machine, proceed to Installing Additional Server Machines.

If you are installing an additional client machine, proceed to Installing Additional Client Machines.

Chapter 2

Installing the First AFS Machine

This chapter describes how to install the first AFS machine in your cell, configuring it as both a file server machine and a client machine. After completing all procedures in this chapter, you can remove the client functionality if you wish, as described in Removing Client Functionality.

To install additional file server machines after completing this chapter, see Installing Additional Server Machines.

To install additional client machines after completing this chapter, see Installing Additional Client Machines.

2.1 Requirements and Configuration Decisions

The instructions in this chapter assume that you meet the following requirements.

- You are logged onto the machine's console as the local superuser **root**
- A standard version of one of the operating systems supported by the current version of AFS is running on the machine
- You have either installed the provided OpenAFS packages for your system, have access to a binary distribution tarball, or have successfully built OpenAFS from source
- You have a Kerberos v5 realm running for your site. If you are working with an existing cell which uses legacy **kaserver** or Kerberos v4 for authentication, please see kaserver and Legacy Kerberos v4 Authentication for the modifications required to this installation procedure.
- You have NTP or a similar time service deployed to ensure rough clock syncronistation between your clients and servers.

You must make the following configuration decisions while installing the first AFS machine. To speed the installation itself, it is best to make the decisions before beginning. See the chapter in the *OpenAFS Administration Guide* about issues in cell administration and configuration for detailed guidelines.

- Select the first AFS machine
- Select the cell name
- Decide which partitions or logical volumes to configure as AFS server partitions, and choose the directory names on which to mount them
- Decide how big to make the client cache
- Decide how to configure the top levels of your cell's AFS filespace

This chapter is divided into three large sections corresponding to the three parts of installing the first AFS machine. Perform all of the steps in the order they appear. Each functional section begins with a summary of the procedures to perform. The sections are as follows:

- Installing server functionality (begins in Overview: Installing Server Functionality)
- Installing client functionality (begins in Overview: Installing Client Functionality)
- Configuring your cell's filespace, establishing further security mechanisms, and enabling access to foreign cells (begins in Overview: Completing the Installation of the First AFS Machine)

2.2 Overview: Installing Server Functionality

In the first phase of installing your cell's first AFS machine, you install file server and database server functionality by performing the following procedures:

1. Choose which machine to install as the first AFS machine
2. Create AFS-related directories on the local disk
3. Incorporate AFS modifications into the machine's kernel
4. Configure partitions or logical volumes for storing AFS volumes
5. On some system types (very rare), install and configure an AFS-modified version of the **fsck** program
6. If the machine is to remain a client machine, incorporate AFS into its authentication system
7. Start the Basic OverSeer (BOS) Server
8. Define the cell name and the machine's cell membership
9. Start the database server processes: Backup Server, Protection Server, and Volume Location (VL) Server
10. Configure initial security mechanisms
11. Start the **fs** process, which incorporates three component processes: the File Server, Volume Server, and Salvager
12. Optionally, start the server portion of the Update Server

2.3 Choosing the First AFS Machine

The first AFS machine you install must have sufficient disk space to store AFS volumes. When you later install additional file server machines in your cell, you can distribute these volumes among the different machines as you see fit.

These instructions configure the first AFS machine as a *database server machine*, and optionally as the *binary distribution machine* for its system type and the cell's *system control machine*. For a description of these roles, see the *OpenAFS Administration Guide*.

Installation of additional machines is simplest if the first machine has the lowest IP address of any database server machine you currently plan to install. If you later install database server functionality on a machine with a lower IP address, you must first update the **/usr/vice/etc/CellServDB** file on all of your cell's client machines. For more details, see Installing Database Server Functionality.

2.4 Creating AFS Directories

If you are installing from packages (such as Debian .deb or Fedora/SuSe .rpm files), you should now install all of the available OpenAFS packages for your system type. Typically, these will include packages for client and server functionality, and a seperate package containing a suitable kernel module for your running kernel. Consult the package lists on the OpenAFS website to determine the packages appropriate for your system. The preparer of such packages may have included some helper scripts to partially automate the creation of a new cell; such scripts can supersede much of the procedures described in the rest of this document.

If you are installing from a tarfile, or from a locally compiled source tree you should create the **/usr/afs** and **/usr/vice/etc** directories on the local disk, to house server and client files respectively. Subsequent instructions copy files from the distribution tarfile into them.

```
# mkdir /usr/afs
# mkdir /usr/vice
# mkdir /usr/vice/etc
```

2.5 Performing Platform-Specific Procedures

Several of the initial procedures for installing a file server machine differ for each system type. For convenience, the following sections group them together for each system type:

- Incorporate AFS modifications into the kernel.

  The kernel on every AFS client machine and, on some systems, the AFS fileservers, must incorporate AFS extensions. On machines that use a dynamic kernel module loader, it is conventional to alter the machine's initialization script to load the AFS extensions at each reboot. The preparer of OS-format binary packages may have included an init script which automates the loading of the needed kernel module, eliminating a need to manually configure this step.

- Configure server partitions or logical volumes to house AFS volumes.

  Every AFS file server machine should have at least one partition or logical volume dedicated to storing AFS volumes (for convenience, the documentation hereafter refers to partitions only). Each server partition is mounted at a directory named **/vicep***xx*, where *xx* is one or two lowercase letters. By convention, the first 26 partitions are mounted on the directories called **/vicepa** through **/vicepz**, the 27th one is mounted on the **/vicepaa** directory, and so on through **/vicepaz** and **/vicepba**, continuing up to the index corresponding to the maximum number of server partitions supported in the current version of AFS (which is specified in the *OpenAFS Release Notes*).

  The **/vicep***xx* directories must reside in the file server machine's root directory, not in one of its subdirectories (for example, **/usr/vicepa** is not an acceptable directory location). The **fileserver** will refuse to mount any **/vicep***xx* folders that are not separate partitions without additional configuration.

  

  Warning
  The separate partition requirement may be overridden by creating a file named /**vicep*xx*/AlwaysAttach**; however, mixed-use partitions, whether cache or fileserver, have the risk that a non-AFS use will fill the partition and not leave enough free space for AFS. Even though it is allowed, be wary of configuring a mixed-use partition without understanding the ramifications of doing so with the workload on your filesystem.

  You can also add or remove server partitions on an existing file server machine. For instructions, see the chapter in the *OpenAFS Administration Guide* about maintaining server machines.

  Note
  Not all file system types supported by an operating system are necessarily supported as AFS server partitions. For possible restrictions, see the *OpenAFS Release Notes*.

- On (rare) system types using the **inode** storage format, install and configure a modified **fsck** program which recognizes the structures that the File Server uses to organize volume data on AFS server partitions. The **fsck** program provided with the operating system does not understand the AFS data structures, and so removes them to the **lost+found** directory.
- If the machine is to remain an AFS client machine, modify the machine's authentication system so that users obtain an AFS token as they log into the local file system. Using AFS is simpler and more convenient for your users if you make the modifications on all client machines. Otherwise, users must perform a two or three step login procedure (login to the local system, then obtain Kerberos credentials, and then issue the **aklog** command). For further discussion of AFS authentication, see the chapter in the *OpenAFS Administration Guide* about cell configuration and administration issues.

To continue, proceed to the appropriate section:

- Getting Started on Linux Systems
- Getting Started on Solaris Systems

2.6 Getting Started on Linux Systems

Since this guide was originally written, the procedure for starting OpenAFS has diverged significantly between different Linux distributions. The instructions that follow are appropriate for both the Fedora and RedHat Enterprise Linux packages distributed by OpenAFS. Additional instructions are provided for those building from source.

Begin by running the AFS client startup scripts, which call the **modprobe** program to dynamically load the AFS modifications into the kernel. Then create partitions for storing AFS volumes. You do not need to replace the Linux **fsck** program. If the machine is to remain an AFS client machine, incorporate AFS into the machine's Pluggable Authentication Module (PAM) scheme.

2.6.1 Loading AFS into the Linux Kernel

The **modprobe** program is the dynamic kernel loader for Linux. Linux does not support incorporation of AFS modifications during a kernel build.

For AFS to function correctly, the **modprobe** program must run each time the machine reboots, so your distribution's AFS initialization script invokes it automatically. The script also includes commands that select the appropriate AFS library file automatically. In this section you run the script.

In later sections you verify that the script correctly initializes all AFS components, then activate a configuration variable, which results in the script being incorporated into the Linux startup and shutdown sequence.

The procedure for starting up OpenAFS depends upon your distribution

2.6.1.1 Fedora and RedHat Enterprise Linux

OpenAFS provides RPMS for all current Fedora and RedHat Enterprise Linux (RHEL) releases prior to EL7 on the OpenAFS web site and the OpenAFS yum repository.

1. Browse to http://dl.openafs.org/dl/openafs/*VERSION*, where VERSION is the latest stable release of OpenAFS for Unix. Download the openafs-repository-*VERSION*.noarch.rpm file for Fedora systems or the openafs-repository-rhel-*VERSION*.noarch.rp file for RedHat-based systems.
2. Install the downloaded RPM file using the following command:

```
# rpm -U openafs-repository*.rpm
```

3. Install the RPM set for your operating system using the yum command as follows:

```
# yum -y install openafs-client openafs-server openafs-krb5 kmod-openafs
```

Alternatively, you may use dynamically-compiled kernel modules if you have the kernel headers, a compiler, and the dkms package from *EPEL* installed.

To use dynamically-compiled kernel modules instead of statically compiled modules, use the following command instead of the kmod-openafs as shown above:

```
# yum install openafs-client openafs-server openafs-krb5 dkms-openafs
```

2.6.1.2 Debian and Ubuntu Linux

OpenAFS is available as binary packages from the Debian linux distribution and its derivatives such as Ubuntu.

1. Install the client and server packages using the following command:

```
# apt-get install openafs-client openafs-modules-dkms openafs-krb5 \
  openafs-fileserver openafs-dbserver
```

You will be prompted by debconf to select your cell name and the size of your local cache.

The Debian package also includes helper scripts `afs-newcell` and `afs-rootvol`, which can automate much of the remainder of this document.

2.6.1.3 Systems built from source

If you are running a system where you have built the system from source yourself, you need to install the relevant components by hand:

1. Unpack the distribution tarball. The examples below assume that you have extracted and built OpenAFS in the **/tmp/afsdist** directory. If you pick a different location, substitute this in all of the following examples. Once you have compiled the distribution, change to the source directory as indicated.

```
# cd /tmp/afsdist
```

2. Copy the AFS kernel library files to the local **/usr/vice/etc/modload** directory. The filenames for the libraries have the format **libafs-***version***.o**, where *version* indicates the kernel build level. The string **.mp** in the *version* indicates that the file is appropriate for machines running a multiprocessor kernel.

```
# mkdir -p /usr/vice/etc/modload
# cp -rp src/libafs/*.ko  /usr/vice/etc/modload
```

3. Copy the AFS initialization script to the local directory for initialization files (by convention, **/etc/rc.d/init.d** on Linux machines). Note the removal of the **.rc** extension as you copy the script.

```
# cp -p src/afsd/afs.rc.linux /etc/rc.d/init.d/afs
```

2.6.2 Configuring Server Partitions on Linux Systems

Every AFS file server machine must have at least one partition or logical volume dedicated to storing AFS volumes. Each server partition is mounted at a directory named **/vicep***xx*, where *xx* is one or two lowercase letters. The **/vicep***xx* directories must reside in the file server machine's root directory, not in one of its subdirectories (for example, **/usr/vicepa** is not an acceptable directory location). For additional information, see Performing Platform-Specific Procedures.

1. Create a directory called **/vicep***xx* for each AFS server partition you are configuring (there must be at least one). Repeat the command for each partition.

```
# mkdir /vicepxx
```

2. Add a line with the following format to the file systems registry file, **/etc/fstab**, for each directory just created. The entry maps the directory name to the disk partition to be mounted on it.

```
/dev/disk  /vicepxx  ext2  defaults  0  2
```

The following is an example for the first partition being configured.

```
/dev/sda8 /vicepa ext2 defaults 0 2
```

3. Create a file system on each partition that is to be mounted at a **/vicep***xx* directory. The following command is probably appropriate, but consult the Linux documentation for more information.

```
# mkfs -v /dev/disk
```

4. Mount each partition by issuing either the **mount -a** command to mount all partitions at once or the **mount** command to mount each partition in turn.

5. If you plan to retain client functionality on this machine after completing the installation, proceed to Enabling AFS Login on Linux Systems. Otherwise, proceed to Starting the BOS Server.

2.6.3 Enabling AFS Login on Linux Systems

Note
If you plan to remove client functionality from this machine after completing the installation, skip this section and proceed to Starting the BOS Server.

At this point you incorporate AFS into the operating system's Pluggable Authentication Module (PAM) scheme. PAM integrates all authentication mechanisms on the machine, including login, to provide the security infrastructure for authenticated access to and from the machine.

You should first configure your system to obtain Kerberos v5 tickets as part of the authentication process, and then run an AFS PAM module to obtain tokens from those tickets after authentication. Many Linux distributions come with a Kerberos v5 PAM module (usually called pam-krb5 or pam_krb5), or you can download and install Russ Allbery's Kerberos v5 PAM module, which is tested regularly with AFS. See the instructions of whatever PAM module you use for how to configure it.

Some Kerberos v5 PAM modules do come with native AFS support (usually requiring the Heimdal Kerberos implementation rather than the MIT Kerberos implementation). If you are using one of those PAM modules, you can configure it to obtain AFS tokens. It's more common, however, to separate the AFS token acquisition into a separate PAM module.

The recommended AFS PAM module is Russ Allbery's pam-afs-session module. It should work with any of the Kerberos v5 PAM modules. To add it to the PAM configuration, you often only need to add configuration to the session group:

Example 2.1 Linux PAM session example

```
session  required  pam_afs_session.so
```

If you also want to obtain AFS tokens for **scp** and similar commands that don't open a session, you will also need to add the AFS PAM module to the auth group so that the PAM `setcred` call will obtain tokens. The `pam_afs_session` module will always return success for authentication so that it can be added to the auth group only for `setcred`, so make sure that it's not marked as `sufficient`.

Example 2.2 Linux PAM auth example

```
auth  [success=ok default=1]  pam_krb5.so
auth  [default=done]          pam_afs_session.so
auth  required                pam_unix.so try_first_pass
```

This example will work if you want to try Kerberos v5 first and then fall back to regular Unix authentication. `success=ok` for the Kerberos PAM module followed by `default=done` for the AFS PAM module will cause a successful Kerberos login to run the AFS PAM module and then skip the Unix authentication module. `default=1` on the Kerberos PAM module causes failure of that module to skip the next module (the AFS PAM module) and fall back to the Unix module. If you want to try Unix authentication first and rearrange the order, be sure to use `default=die` instead.

The PAM configuration is stored in different places in different Linux distributions. On Red Hat, look in `/etc/pam.d/system-auth`. On Debian and derivatives, look in `/etc/pam.d/common-session` and `/etc/pam.d/common-auth`.

For additional configuration examples and the configuration options of the AFS PAM module, see its documentation. For more details on the available options for the PAM configuration, see the Linux PAM documentation.

Sites which still require the deprecated **kaserver** or external Kerberos v4 authentication should consult Enabling kaserver based AFS Login on Linux Systems for details of how to enable AFS login on Linux.

Proceed to Starting the BOS Server (or if referring to these instructions while installing an additional file server machine, return to Starting Server Programs).

2.7 Getting Started on Solaris Systems

Begin by running the AFS initialization script to call the **modload** program distributed by Sun Microsystems, which dynamically loads AFS modifications into the kernel. Then create partitions for storing AFS volumes, and install and configure the AFS-modified **fsck** program to run on AFS server partitions. If the machine is to remain an AFS client machine, incorporate AFS into the machine's Pluggable Authentication Module (PAM) scheme.

2.7.1 Loading AFS into the Solaris Kernel

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for Solaris systems. Solaris does not support incorporation of AFS modifications during a kernel build.

For AFS to function correctly, the **modload** program must run each time the machine reboots, so the AFS initialization script (included on the AFS CD-ROM) invokes it automatically. In this section you copy the appropriate AFS library file to the location where the **modload** program accesses it and then run the script.

In later sections you verify that the script correctly initializes all AFS components, then create the links that incorporate AFS into the Solaris startup and shutdown sequence.

1. Unpack the OpenAFS Solaris distribution tarball. The examples below assume that you have unpacked the files into the **/tmp/afsdist** directory. If you pick a diferent location, substitute this in all of the following exmaples. Once you have unpacked the distribution, change directory as indicated.

   ```
   # cd  /tmp/afsdist/sun4x_56/dest/root.client/usr/vice/etc
   ```

2. Copy the AFS initialization script to the local directory for initialization files (by convention, **/etc/init.d** on Solaris machines). Note the removal of the **.rc** extension as you copy the script.

   ```
   # cp -p  afs.rc  /etc/init.d/afs
   ```

3. Copy the appropriate AFS kernel library file to the local file **/kernel/fs/afs**.

   If the machine is running Solaris 11 on the x86_64 platform:

```
# cp -p modload/libafs64.o /kernel/drv/amd64/afs
```

If the machine is running Solaris 10 on the x86_64 platform:

```
# cp -p modload/libafs64.o /kernel/fs/amd64/afs
```

If the machine is running Solaris 2.6 or the 32-bit version of Solaris 7, its kernel supports NFS server functionality, and the **nfsd** process is running:

```
# cp -p modload/libafs.o /kernel/fs/afs
```

If the machine is running Solaris 2.6 or the 32-bit version of Solaris 7, and its kernel does not support NFS server functionality or the **nfsd** process is not running:

```
# cp -p modload/libafs.nonfs.o /kernel/fs/afs
```

If the machine is running the 64-bit version of Solaris 7, its kernel supports NFS server functionality, and the **nfsd** process is running:

```
# cp -p modload/libafs64.o /kernel/fs/sparcv9/afs
```

If the machine is running the 64-bit version of Solaris 7, and its kernel does not support NFS server functionality or the **nfsd** process is not running:

```
# cp -p modload/libafs64.nonfs.o /kernel/fs/sparcv9/afs
```

4. Run the AFS initialization script to load AFS modifications into the kernel. You can ignore any error messages about the inability to start the BOS Server or the Cache Manager or AFS client.

```
# /etc/init.d/afs start
```

When an entry called `afs` does not already exist in the local **/etc/name_to_sysnum** file, the script automatically creates it and reboots the machine to start using the new version of the file. If this happens, log in again as the superuser **root** after the reboot and run the initialization script again. This time the required entry exists in the **/etc/name_to_sysnum** file, and the **modload** program runs.

```
login: root
Password: root_password
# /etc/init.d/afs start
```

2.7.2 Configuring the AFS-modified fsck Program on Solaris Systems

In this section, you make modifications to guarantee that the appropriate **fsck** program runs on AFS server partitions. The **fsck** program provided with the operating system must never run on AFS server partitions. Because it does not recognize the structures that the File Server uses to organize volume data, it removes all of the data. To repeat:

Never run the standard fsck program on AFS server partitions. It discards AFS volumes.

1. Create the **/usr/lib/fs/afs** directory to house the AFS-modified **fsck** program and related files.

```
# mkdir /usr/lib/fs/afs
# cd /usr/lib/fs/afs
```

2. Copy the **vfsck** binary to the newly created directory, changing the name as you do so.

```
# cp  /tmp/afsdist/sun4x_56/dest/root.server/etc/vfsck  fsck
```

3. Working in the **/usr/lib/fs/afs** directory, create the following links to Solaris libraries:

```
# ln -s /usr/lib/fs/ufs/clri
# ln -s /usr/lib/fs/ufs/df
# ln -s /usr/lib/fs/ufs/edquota
# ln -s /usr/lib/fs/ufs/ff
# ln -s /usr/lib/fs/ufs/fsdb
# ln -s /usr/lib/fs/ufs/fsirand
# ln -s /usr/lib/fs/ufs/fstyp
# ln -s /usr/lib/fs/ufs/labelit
# ln -s /usr/lib/fs/ufs/lockfs
# ln -s /usr/lib/fs/ufs/mkfs
# ln -s /usr/lib/fs/ufs/mount
# ln -s /usr/lib/fs/ufs/ncheck
# ln -s /usr/lib/fs/ufs/newfs
# ln -s /usr/lib/fs/ufs/quot
# ln -s /usr/lib/fs/ufs/quota
# ln -s /usr/lib/fs/ufs/quotaoff
# ln -s /usr/lib/fs/ufs/quotaon
# ln -s /usr/lib/fs/ufs/repquota
# ln -s /usr/lib/fs/ufs/tunefs
# ln -s /usr/lib/fs/ufs/ufsdump
# ln -s /usr/lib/fs/ufs/ufsrestore
# ln -s /usr/lib/fs/ufs/volcopy
```

4. Append the following line to the end of the file **/etc/dfs/fstypes**.

```
afs AFS Utilities
```

5. Edit the **/sbin/mountall** file, making two changes.

   - Add an entry for AFS to the `case` statement for option 2, so that it reads as follows:

```
case "$2" in
ufs)    foptions="-o p"
        ;;
afs)    foptions="-o p"
        ;;
s5)     foptions="-y -t /var/tmp/tmp$$ -D"
        ;;
*)      foptions="-y"
        ;;
```

   - Edit the file so that all AFS and UFS partitions are checked in parallel. Replace the following section of code:

```
# For  fsck purposes, we make a distinction between ufs and
# other file systems
#
if [ "$fstype" = "ufs" ]; then
     ufs_fscklist="$ufs_fscklist $fsckdev"
     saveentry $fstype "$OPTIONS" $special $mountp
     continue
fi
```

   with the following section of code:

```
# For fsck purposes, we make a distinction between ufs/afs
# and other file systems.
#
if [ "$fstype" = "ufs" -o "$fstype" = "afs" ]; then
     ufs_fscklist="$ufs_fscklist $fsckdev"
     saveentry $fstype "$OPTIONS" $special $mountp
     continue
fi
```

2.7.3 Configuring Server Partitions on Solaris Systems

Every AFS file server machine must have at least one partition or logical volume dedicated to storing AFS volumes. Each server partition is mounted at a directory named **/vicep***xx*, where *xx* is one or two lowercase letters. The **/vicep***xx* directories must reside in the file server machine's root directory, not in one of its subdirectories (for example, **/usr/vicepa** is not an acceptable directory location). For additional information, see Performing Platform-Specific Procedures.

1. Create a directory called **/vicep***xx* for each AFS server partition you are configuring (there must be at least one). Repeat the command for each partition.

   ```
   # mkdir /vicepxx
   ```

2. Add a line with the following format to the file systems registry file, **/etc/vfstab**, for each partition to be mounted on a directory created in the previous step. Note the value `afs` in the fourth field, which tells Solaris to use the AFS-modified **fsck** program on this partition.

   ```
   /dev/dsk/disk   /dev/rdsk/disk   /vicepxx   afs   boot_order   yes
   ```

   The following is an example for the first partition being configured.

   ```
   /dev/dsk/c0t6d0s1 /dev/rdsk/c0t6d0s1 /vicepa afs 3 yes
   ```

3. Create a file system on each partition that is to be mounted at a **/vicep***xx* directory. The following command is probably appropriate, but consult the Solaris documentation for more information.

   ```
   # newfs -v /dev/rdsk/disk
   ```

4. Issue the **mountall** command to mount all partitions at once.

5. If you plan to retain client functionality on this machine after completing the installation, proceed to Enabling AFS Login and Editing the File Systems Clean-up Script on Solaris Systems. Otherwise, proceed to Starting the BOS Server.

2.7.4 Enabling AFS Login on Solaris Systems

Note

If you plan to remove client functionality from this machine after completing the installation, skip this section and proceed to Starting the BOS Server.

At this point you incorporate AFS into the operating system's Pluggable Authentication Module (PAM) scheme. PAM integrates all authentication mechanisms on the machine, including login, to provide the security infrastructure for authenticated access to and from the machine.

Explaining PAM is beyond the scope of this document. It is assumed that you understand the syntax and meanings of settings in the PAM configuration file (for example, how the `other` entry works, the effect of marking an entry as `required`, `optional`, or `sufficient`, and so on).

You should first configure your system to obtain Kerberos v5 tickets as part of the authentication process, and then run an AFS PAM module to obtain tokens from those tickets after authentication. Current versions of Solaris come with a Kerberos v5 PAM module that will work, or you can download and install Russ Allbery's Kerberos v5 PAM module, which is tested regularly with AFS. See the instructions of whatever PAM module you use for how to configure it.

Some Kerberos v5 PAM modules do come with native AFS support (usually requiring the Heimdal Kerberos implementation rather than the MIT Kerberos implementation). If you are using one of those PAM modules, you can configure it to obtain AFS tokens. It's more common, however, to separate the AFS token acquisition into a separate PAM module.

The recommended AFS PAM module is Russ Allbery's pam-afs-session module. It should work with any of the Kerberos v5 PAM modules. To add it to the PAM configuration, you often only need to add configuration to the session group in `pam.conf`:

Example 2.3 Solaris PAM session example

```
login session required pam_afs_session.so
```

This example enables PAM authentication only for console login. You may want to add a similar line for the ssh service and for any other login service that you use, including possibly the `other` service (which serves as a catch-all). You may also want to add options to the AFS PAM session module (particularly `retain_after_close`, which is necessary for some versions of Solaris.

For additional configuration examples and the configuration options of the AFS PAM module, see its documentation. For more details on the available options for the PAM configuration, see the `pam.conf` manual page.

Sites which still require **kaserver** or external Kerberos v4 authentication should consult "Enabling kaserver based AFS Login on Solaris Systems" for details of how to enable AFS login on Solaris.

Proceed to Editing the File Systems Clean-up Script on Solaris Systems

2.7.5 Editing the File Systems Clean-up Script on Solaris Systems

1. Some Solaris distributions include a script that locates and removes unneeded files from various file systems. Its conventional location is **/usr/lib/fs/nfs/nfsfind**. The script generally uses an argument to the **find** command to define which file systems to search. In this step you modify the command to exclude the **/afs** directory. Otherwise, the command traverses the AFS filespace of every cell that is accessible from the machine, which can take many hours. The following alterations are possibilities, but you must verify that they are appropriate for your cell.

   The first possible alteration is to add the **-local** flag to the existing command, so that it looks like the following:

   ```
   find $dir -local -name .nfs\* -mtime +7 -mount -exec rm -f {} \;
   ```

   Another alternative is to exclude any directories whose names begin with the lowercase letter **a** or a non-alphabetic character.

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

   Do not use the following command, which still searches under the **/afs** directory, looking for a subdirectory of type **4.2**.

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

2. Proceed to Starting the BOS Server (or if referring to these instructions while installing an additional file server machine, return to Starting Server Programs).

2.8 Starting the BOS Server

You are now ready to start the AFS server processes on this machine. If you are not working from a packaged distribution, begin by installing the AFS server binaries to the conventional local disk location, the **/usr/afs/bin** directory. The following instructions also create files in other subdirectories of the **/usr/afs** directory.

Then obtain a krb5 keytab for use by the servers in the cell. Once the keytab is in place, issue the **bosserver** command to initialize the Basic OverSeer (BOS) Server, which monitors and controls other AFS server processes on its server machine. Because you have not yet configured your cell's AFS authentication and authorization mechanisms, you must always use the **-localauth** flag to commands, to use a printed token that does not correspond to a normal krb5 identity.

Older versions of these instructions used the **-noauth** flag, which completely disables all authentication and authorization checking, allowing anyone at all to control the system. Do not use this flag! It is highly insecure, and is no longer needed.

As it initializes for the first time, the BOS Server creates the following directories and files, setting the owner to the local superuser **root** and the mode bits to limit the ability to write (and in some cases, read) them. For a description of the contents and function of these directories and files, see the chapter in the *OpenAFS Administration Guide* about administering server machines. For further discussion of the mode bit settings, see Protecting Sensitive AFS Directories.

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

The BOS Server also creates symbolic links called **/usr/vice/etc/ThisCell** and **/usr/vice/etc/CellServDB** to the corresponding files in the **/usr/afs/etc** directory. The AFS command interpreters consult the **CellServDB** and **ThisCell** files in the **/usr/vice/etc** directory because they generally run on client machines. On machines that are AFS servers only (as this machine currently is), the files reside only in the **/usr/afs/etc** directory; the links enable the command interpreters to retrieve the information they need. Later instructions for installing the client functionality replace the links with actual files.

2.8.1 Generating the Cell's Kerberos V5 Keys

This guide uses krb5 for authentication; do not use the legacy **kaserver** for new installations.

This section creates only the cell-wide shared secret key; administrative users will be created later in the procedure. This cell-wide key has the principal name **afs/*cell***. No user logs in under this identity, but it is used to encrypt the server tickets that the KDC grants to AFS clients for presentation to server processes during mutual authentication. (The chapter in the *OpenAFS Administration Guide* about cell configuration and administration describes the role of server encryption keys in mutual authentication.)

The OpenAFS 1.8.x series stores the cell-wide shared keys in the file **/usr/afs/etc/KeyFileExt**, whereas the 1.6.x series uses a krb5 keytab format file in **/usr/afs/etc/rxkad.keytab**. These instructions create both files, but populating the **KeyFileExt** file will only succeed using the version of **asetkey** from OpenAFS 1.8.x.

The examples below assume you are using MIT Kerberos. Please refer to the documentation for your KDC's administrative interface if you are using a different vendor

1. Enter **kadmin** interactive mode.

```
    # kadmin
Authenticating as principal you/admin@YOUR REALM with password
Password for you/admin@REALM: your_password
```

2. Issue the **add_principal** command to create a Kerberos Database entry for **afs/<*cell name*>**.

   Note that when creating the **afs/<*cell name*>** entry, the encryption type list does not include any single-DES encryption types. If such encryption types are included, additional **asetkey** commands will be needed to place those keys in the legacy **KeyFile** and ensure proper operation of the cell. For more details regarding encryption types, see the documentation for your Kerberos installation.

```
kadmin: add_principal -randkey -e aes256-cts-hmac-sha1-96:normal,aes128-cts-hmac- ↩
    sha1-96:normal afs/<cell name>
Principal "afs/cell name@REALM" created.
```

3. Extract the newly created key for **afs/*cell*** to a keytab on the local machine.

   The keytab contains the key material that ensures the security of your AFS cell. You should ensure that it is kept in a secure location at all times.

```
  kadmin: ktadd -k /usr/afs/etc/rxkad.keytab -e aes256-cts-hmac-sha1-96:normal,aes128 ↩
     -cts-hmac-sha1-96:normal afs/<cell name>
Entry for principal afs/<cell name> with kvno 2, encryption type aes256-cts-hmac-sha1 ↩
    -96 added to keytab WRFILE:/usr/afs/etc/rxkad.keytab
Entry for principal afs/<cell name> with kvno 2, encryption type aes128-cts-hmac-sha1 ↩
    -96 added to keytab WRFILE:/usr/afs/etc/rxkad.keytab
```

Make a note of the key version number (kvno) given in the response, as you will need it to load the key into the **KeyFileExt** in a later step

---

Note

Note that each time you run **ktadd** a new key is generated for the item being extracted. This means that you cannot run ktadd multiple times and end up with the same key material each time.

---

4. Issue the **quit** command to leave **kadmin** interactive mode.

```
kadmin: quit
```

5. Issue the **asetkey** command to set the AFS server encryption key in the **/usr/afs/etc/KeyFileExt** file. This key is created from the **rxkad.keytab** file created earlier.

   asetkey requires the key version number (or kvno) of the **afs/***cell* key, as well as the encryption type number of the key. You should have made note of the kvno when creating the key earlier. The key version number can also be found by running the **kvno** command

```
# kvno -kt /usr/afs/etc/rxkad.keytab
```

   The encryption type numbers can be found in the local krb5 headers or the IANA registry. The most common numbers are 18 for **aes256-cts-hmac-sha1-96** and 17 for **aes128-cts-hmac-sha1-96**.

   Once the kvno and enctypes are known, the keys can then be extracted using asetkey

```
# asetkey add rxkad_krb5 <kvno>  18 /usr/afs/etc/rxkad.keytab afs/<cell name>
# asetkey add rxkad_krb5 <kvno>  17 /usr/afs/etc/rxkad.keytab afs/<cell name>
```

2.8.2 Starting the Server Processes

Now that the keys are in place, proceed to start the server processes:

1. If you are building from source, you need to install the compiled files to the local **/usr/afs** directory.
2. Issue the **bosserver** command.

```
# /usr/afs/bin/bosserver
```

3. Verify that the BOS Server created **/usr/vice/etc/ThisCell** and **/usr/vice/etc/CellServDB** as symbolic links to the corresponding files in the **/usr/afs/etc** directory.

```
# ls -l  /usr/vice/etc
```

   If either or both of **/usr/vice/etc/ThisCell** and **/usr/vice/etc/CellServDB** do not exist, or are not links, issue the following commands.

```
# cd /usr/vice/etc
# ln -s /usr/afs/etc/ThisCell
# ln -s /usr/afs/etc/CellServDB
```

2.9 Defining Cell Name and Membership for Server Processes

Now assign your cell's name. The chapter in the *OpenAFS Administration Guide* about cell configuration and administration issues discusses the important considerations, explains why changing the name is difficult, and outlines the restrictions on name format. Two of the most important restrictions 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
- **/usr/afs/etc/CellServDB**, which lists the cell's database server machines; the machine named on the command line is placed on the list automatically

---

Note
In the following and every instruction in this guide, for the *machine name* argument substitute the fully-qualified hostname (such as **fs1.example.com**) of the machine you are installing. For the *cell name* argument substitute your cell's complete name (such as **example.com**).

---

1. If necessary, add the directory containing the **bos** command to your path.

```
# export PATH=$PATH:/usr/afs/bin
```

2. Issue the **bos setcellname** command to set the cell name.

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

3. Issue the **bos listhosts** command to verify that the machine you are installing is now registered as the cell's first database server machine.

```
# bos listhosts <machine name> -localauth
Cell name is cell_name
    Host 1 is machine_name
```

2.10 Starting the Database Server Processes

Next use the **bos create** command to create entries for the three database server processes in the **/usr/afs/local/BosConfig** file and start them running. The three processes run on database server machines only:

- The Protection Server (the **ptserver** process) maintains the Protection Database
- The Volume Location (VL) Server (the **vlserver** process) maintains the Volume Location Database (VLDB)
- The optional Backup Server (the **buserver** process) maintains the Backup Database

---

Note
AFS ships with an additional database server named 'kaserver', which was historically used to provide authentication services to AFS cells. kaserver was based on *Kerberos v4*, as such, it is not recommended for new cells. This guide assumes you have already configured a Kerberos v5 realm for your site, and details the procedures required to use AFS with this realm. If you do wish to use **kaserver**, please see the modifications to these instructions detailed in Starting the kaserver Database Server Process

---

The remaining instructions in this chapter include the **-cell** argument on all applicable commands. Provide the cell name you assigned in Defining Cell Name and Membership for Server Processes. If a command appears on multiple lines, it is only for legibility.

1. Issue the **bos create** command to start the Protection Server.

```
# ./bos create <machine name> ptserver simple /usr/afs/bin/ptserver -localauth
```

2. Issue the **bos create** command to start the VL Server.

```
# ./bos create <machine name> vlserver simple /usr/afs/bin/vlserver -localauth
```

3. Optionally, issue the **bos create** command to start the Backup Server.

```
# ./bos create <machine name> buserver simple /usr/afs/bin/buserver  -localauth
```

2.11 Initializing Cell Security

If you are working with an existing cell which uses **kaserver** or Kerberos v4 for authentication, please see Initializing Cell Security with kaserver for installation instructions which replace this section.

Now finish initializing the cell's security mechanisms. Begin by creating the following entry in your site's Kerberos database:

- A generic administrative account, called **admin** by convention. If you choose to assign a different name, substitute it throughout the remainder of this document.

  After you complete the installation of the first machine, you can continue to have all administrators use the **admin** account, or you can create a separate administrative account for each of them. The latter scheme implies somewhat more overhead, but provides a more informative audit trail for administrative operations.

You also issue several commands that enable the new **admin** user to issue privileged commands in all of the AFS suites.

The following instructions do not configure all of the security mechanisms related to the AFS Backup System. See the chapter in the *OpenAFS Administration Guide* about configuring the Backup System.

The examples below assume you are using MIT Kerberos. Please refer to the documentation for your KDC's administrative interface if you are using a different vendor

1. Enter **kadmin** interactive mode.

```
   # kadmin
Authenticating as principal you/admin@YOUR REALM with password
Password for you/admin@REALM: your_password
```

2. Issue the **add_principal** command to create the Kerberos Database entry for **admin**.

   You should make the *admin_passwd* as long and complex as possible, but keep in mind that administrators need to enter it often. It must be at least six characters long.

```
kadmin:  add_principal admin
Enter password for principal "admin@REALM": admin_password
Principal "admin@REALM" created.
```

3. Issue the **quit** command to leave **kadmin** interactive mode.

```
kadmin: quit
```

4. Issue the **bos adduser** command to add the **admin** user to the **/usr/afs/etc/UserList** file. This enables the **admin** user to issue privileged **bos** and **vos** commands.

```
# ./bos adduser <machine name> admin -localauth
```

2.12 Initializing the Protection Database

Now continue to configure your cell's security systems by populating the Protection Database with the newly created **admin** user, and permitting it to issue priviledged commands on the AFS filesystem. There is nothing special about the name "admin"; it is just a convenient name for these instructions. An other name could be used throughout this document, or multiple privileged accounts created.

1. Issue the **pts createuser** command to create a Protection Database entry for the **admin** user.

   By default, the Protection Server assigns AFS UID 1 (one) to the **admin** user, because it is the first user entry you are creating. If the local password file (**/etc/passwd** or equivalent) already has an entry for **admin** that assigns it a UNIX UID other than 1, it is best to use the **-id** argument to the **pts createuser** command to make the new AFS UID match the existing UNIX UID. Otherwise, it is best to accept the default.

   ```
   # pts createuser -name admin [-id <AFS UID>]  -localauth
   User admin has id AFS UID
   ```

2. Issue the **pts adduser** command to make the **admin** user a member of the **system:administrators** group, and the **pts membership** command to verify the new membership. Membership in the group enables the **admin** user to issue privileged **pts** commands and some privileged **fs** commands.

   ```
   # ./pts adduser admin system:administrators -localauth
   # ./pts membership admin -localauth
   Groups admin (id: 1) is a member of:
     system:administrators
   ```

2.13 Starting the File Server processes

Start the **dafs** process. The **dafs** process consists of the Demand-Attach File Server, Volume Server, Salvage Server, and Salvager (**dafileserver**, **davolserver**, **salvageserver**, and **dasalvager** processes). Most sites should run the Demand-Attach File Server, but the traditional/legacy File Server remains an option. If you are uncertain whether to run the legacy File Server, see Appendix C, The Demand-Attach File Server.

1. Issue the **bos create** command to start the **dafs** process. The commands appear here on multiple lines only for legibility.

   - Create the **dafs** process:

     ```
     # ./bos create  <machine name> dafs dafs /usr/afs/bin/dafileserver   \
                     /usr/afs/bin/davolserver /usr/afs/bin/salvageserver \
                     /usr/afs/bin/dasalvager -localauth
     ```

   Sometimes a message about Volume Location Database (VLDB) initialization appears, along with one or more instances of an error message similar to the following:

   ```
   FSYNC_clientInit temporary failure (will retry)
   ```

   This message appears when the **volserver** process tries to start before the **fileserver** process has completed its initialization. Wait a few minutes after the last such message before continuing, to guarantee that both processes have started successfully.

   You can verify that the **dafs** process has started successfully by issuing the **bos status** command. Its output mentions two `proc starts`.

   ```
   # ./bos status <machine name> dafs -long -localauth
   ```

2. Your next action depends on whether you have ever run AFS file server machines in the cell:

- If you are installing the first AFS server machine ever in the cell (that is, you are not upgrading the AFS software from a previous version), create the first AFS volume, **root.afs**.
  For the *partition name* argument, substitute the name of one of the machine's AFS server partitions (such as **/vicepa**).

```
# ./vos create  <machine name> <partition name> root.afs   \
                -localauth
```

  The Volume Server produces a message confirming that it created the volume on the specified partition.

2.14 Clock Sync Considerations

Keeping the clocks on all server and client machines in your cell synchronized is crucial to several functions, and in particular to the correct operation of AFS's distributed database technology, Ubik. The chapter in the *OpenAFS Administration Guide* about administering server machines explains how time skew can disturb Ubik's performance and cause service outages in your cell.

You should install and configure your time service independently of AFS. Your Kerberos realm will also require a reliable time source, so your site may already have one available.

2.15 Overview: Installing Client Functionality

The machine you are installing is now an AFS file server machine and database server machine. Now make it a client machine by completing the following tasks:

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

2.16 Copying Client Files to the Local Disk

You need only undertake the steps in this section, if you are using a tar file distribution, or one built from scratch. Packaged distributions, such as RPMs or DEBs will already have installed the necessary files in the correct locations.

Before installing and configuring the AFS client, copy the necessary files from the tarball to the local **/usr/vice/etc** directory.

1. If you have not already done so, unpack the distribution tarball for this machine's system type into a suitable location on the filesystem, such as **/tmp/afsdist**. If you use a different location, substitue that in the examples that follow.
2. Copy files to the local **/usr/vice/etc** directory.

   This step places a copy of the AFS initialization script (and related files, if applicable) into the **/usr/vice/etc** directory. In the preceding instructions for incorporating AFS into the kernel, you copied the script directly to the operating system's conventional location for initialization files. When you incorporate AFS into the machine's startup sequence in a later step, you can choose to link the two files.

   On some system types that use a dynamic kernel loader program, you previously copied AFS library files into a subdirectory of the **/usr/vice/etc** directory. On other system types, you copied the appropriate AFS library file directly to the directory where the operating system accesses it. The following commands do not copy or recopy the AFS library files into the **/usr/vice/etc** directory, because on some system types the library files consume a large amount of space. If you want to copy them, add the **-r** flag to the first **cp** command and skip the second **cp** command.

```
# cd /tmp/afsdist/sysname/root.client/usr/vice/etc
# cp -p  *  /usr/vice/etc
# cp -rp  C  /usr/vice/etc
```

2.17 Defining Cell Membership for Client Processes

Every AFS client machine has a copy of the **/usr/vice/etc/ThisCell** file on its local disk to define the machine's cell membership for the AFS client programs that run on it. The **ThisCell** file you created in the **/usr/afs/etc** directory (in Defining Cell Name and Membership for Server Processes) is used only by server processes.

Among other functions, the **ThisCell** file on a client machine determines the following:

- The cell in which users gain tokens when they log onto the machine, assuming it is using an AFS-modified login utility
- The cell in which users gain tokens by default when they issue the **aklog** command
- The cell membership of the AFS server processes that the AFS command interpreters on this machine contact by default

1. Change to the **/usr/vice/etc** directory and remove the symbolic link created in Starting the BOS Server.

```
# cd /usr/vice/etc
# rm ThisCell
```

2. Create the **ThisCell** file as a copy of the **/usr/afs/etc/ThisCell** file. Defining the same local cell for both server and client processes leads to the most consistent AFS performance.

```
# cp  /usr/afs/etc/ThisCell  ThisCell
```

2.18 Creating the Client CellServDB File

The **/usr/vice/etc/CellServDB** file on a client machine's local disk lists the database server machines for each cell that the local Cache Manager can contact. If there is no entry in the file for a cell, or if the list of database server machines is wrong, then users working on this machine cannot access the cell. The chapter in the *OpenAFS Administration Guide* about administering client machines explains how to maintain the file after creating it.

As the **afsd** program initializes the Cache Manager, it copies the contents of the **CellServDB** file into kernel memory. The Cache Manager always consults the list in kernel memory rather than the **CellServDB** file itself. Between reboots of the machine, you can use the **fs newcell** command to update the list in kernel memory directly; see the chapter in the *OpenAFS Administration Guide* about administering client machines.

The AFS distribution includes the file **CellServDB.dist**. It includes an entry for all AFS cells that agreed to share their database server machine information at the time the distribution was created. The definitive copy of this file is maintained at grand.central.org, and updates may be obtained from /afs/grand.central.org/service/CellServDB or http://grand.central.org/dl/cellservdb/CellServDB

The **CellServDB.dist** file can be a good basis for the client **CellServDB** file, because all of the entries in it use the correct format. You can add or remove cell entries as you see fit. Depending on your cache manager configuration, additional steps (as detailed in Enabling Access to Foreign Cells) may be required to enable the Cache Manager to actually reach the cells.

In this section, you add an entry for the local cell to the local **CellServDB** file. The current working directory is still **/usr/vice/etc**.

1. Remove the symbolic link created in Starting the BOS Server and rename the **CellServDB.sample** file to **CellServDB**.

```
# rm CellServDB
# mv CellServDB.sample CellServDB
```

2. Add an entry for the local cell to the **CellServDB** file. One easy method is to use the **cat** command to append the contents of the server **/usr/afs/etc/CellServDB** file to the client version.

```
# cat  /usr/afs/etc/CellServDB >>  CellServDB
```

Then open the file in a text editor to verify that there are no blank lines, and that all entries have the required format, which is described just following. The ordering of cells is not significant, but it can be convenient to have the client machine's home cell at the top; move it there now if you wish.

- The first line of a cell's entry has the following format:

```
>cell_name      #organization
```

where *cell_name* is the cell's complete Internet domain name (for example, **example.com**) and *organization* is an optional field that follows any number of spaces and the number sign (#). By convention it names the organization to which the cell corresponds (for example, the Example Corporation).

- After the first line comes a separate line for each database server machine. Each line has the following format:

```
IP_address   #machine_name
```

where *IP_address* is the machine's IP address in dotted decimal format (for example, 192.12.105.3). Following any number of spaces and the number sign (#) is *machine_name*, the machine's fully-qualified hostname (for example, **db1.example.com**). In this case, the number sign does not indicate a comment; *machine_name* is a required field.

3. If the file includes cells that you do not wish users of this machine to access, remove their entries.

The following example shows entries for two cells, each of which has three database server machines:

```
>example.com       #Example Corporation (home cell)
192.12.105.3      #db1.example.com
192.12.105.4      #db2.example.com
192.12.105.55     #db3.example.com
>example.org    #Example Organization cell
138.255.68.93     #serverA.example.org
138.255.68.72     #serverB.example.org
138.255.33.154    #serverC.example.org
```

2.19 Configuring the Cache

The Cache Manager uses a cache on the local disk or in machine memory to store local copies of files fetched from file server machines. As the **afsd** program initializes the Cache Manager, it sets basic cache configuration parameters according to definitions in the local **/usr/vice/etc/cacheinfo** file. The file has three fields:

1. The first field names the local directory on which to mount the AFS filespace. The conventional location is the **/afs** directory.
2. The second field defines the local disk directory to use for the disk cache. The conventional location is the **/usr/vice/cache** directory, but you can specify an alternate directory if another partition has more space available. There must always be a value in this field, but the Cache Manager ignores it if the machine uses a memory cache.
3. The third field specifies the number of kilobyte (1024 byte) blocks to allocate for the cache.

The values you define must meet the following requirements.

- On a machine using a disk cache, the Cache Manager expects always to be able to use the amount of space specified in the third field. Failure to meet this requirement can cause serious problems, some of which can be repaired only by rebooting. You must prevent non-AFS processes from filling up the cache partition. The simplest way is to devote a partition to the cache exclusively.
- The amount of space available in memory or on the partition housing the disk cache directory imposes an absolute limit on cache size.
- The maximum supported cache size can vary in each AFS release; see the *OpenAFS Release Notes* for the current version.

- For a disk cache, you cannot specify a value in the third field that exceeds 95% of the space available on the partition mounted at the directory named in the second field. If you violate this restriction, the **afsd** program exits without starting the Cache Manager and prints an appropriate message on the standard output stream. A value of 90% is more appropriate on most machines. Some operating systems do not automatically reserve some space to prevent the partition from filling completely; for them, a smaller value (say, 80% to 85% of the space available) is more appropriate.
- For a memory cache, you must leave enough memory for other processes and applications to run. If you try to allocate more memory than is actually available, the **afsd** program exits without initializing the Cache Manager and produces the following message on the standard output stream.

  ```
  afsd: memCache allocation failure at number KB
  ```

  The *number* value is how many kilobytes were allocated just before the failure, and so indicates the approximate amount of memory available.

Within these hard limits, the factors that determine appropriate cache size include the number of users working on the machine, the size of the files with which they work, and (for a memory cache) the number of processes that run on the machine. The higher the demand from these factors, the larger the cache needs to be to maintain good performance.

Disk caches smaller than 10 MB do not generally perform well. Machines serving multiple users usually perform better with a cache of at least 60 to 70 MB. The point at which enlarging the cache further does not really improve performance depends on the factors mentioned previously and is difficult to predict.

Memory caches smaller than 1 MB are nonfunctional, and the performance of caches smaller than 5 MB is usually unsatisfactory. Suitable upper limits are similar to those for disk caches but are probably determined more by the demands on memory from other sources on the machine (number of users and processes). Machines running only a few processes possibly can use a smaller memory cache.

2.19.1 Configuring a Disk Cache

Note
Not all file system types that an operating system supports are necessarily supported for use as the cache partition. For possible restrictions, see the *OpenAFS Release Notes*.

To configure the disk cache, perform the following procedures:

1. Create the local directory to use for caching. The following instruction shows the conventional location, **/usr/vice/cache**. If you are devoting a partition exclusively to caching, as recommended, you must also configure it, make a file system on it, and mount it at the directory created in this step.

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

2. Create the **cacheinfo** file to define the configuration parameters discussed previously. The following 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
   ```

   The following example defines the disk cache size as 50,000 KB:

   ```
   # echo "/afs:/usr/vice/cache:50000" > /usr/vice/etc/cacheinfo
   ```

2.19.2 Configuring a Memory Cache

To configure a memory cache, create the **cacheinfo** file to define the configuration parameters discussed previously. The following instruction shows the standard mount location, **/afs**, and the standard cache location, **/usr/vice/cache** (though the exact value of the latter is irrelevant for a memory cache).

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

The following example allocates 25,000 KB of memory for the cache.

```
# echo "/afs:/usr/vice/cache:25000" > /usr/vice/etc/cacheinfo
```

2.20 Configuring the Cache Manager

By convention, the Cache Manager mounts the AFS filespace on the local **/afs** directory. In this section you create that directory.

The **afsd** program sets several cache configuration parameters as it initializes the Cache Manager, and starts daemons that improve performance. You can use the **afsd** command's arguments to override the parameters' default values and to change the number of some of the daemons. Depending on the machine's cache size, its amount of RAM, and how many people work on it, you can sometimes improve Cache Manager performance by overriding the default values. For a discussion of all of the **afsd** command's arguments, see its reference page in the *OpenAFS Administration Reference*.

On platforms using the standard 'afs' initialisation script (this does not apply to Fedora or RHEL based distributions), the **afsd** command line in the AFS initialization script on each system type includes an `OPTIONS` variable. You can use it to set nondefault values for the command's arguments, in one of the following ways:

- You can create an **afsd** *options file* that sets values for arguments to the **afsd** command. If the file exists, its contents are automatically substituted for the `OPTIONS` variable in the AFS initialization script. The AFS distribution for some system types includes an options file; on other system types, you must create it.

  You use two variables in the AFS initialization script to specify the path to the options file: `CONFIG` and `AFSDOPT`. On system types that define a conventional directory for configuration files, the `CONFIG` variable indicates it by default; otherwise, the variable indicates an appropriate location.

  List the desired **afsd** options on a single line in the options file, separating each option with one or more spaces. The following example sets the **-stat** argument to 2500, the **-daemons** argument to 4, and the **-volumes** argument to 100.

  ```
  -stat 2500 -daemons 4 -volumes 100
  ```

- On a machine that uses a disk cache, you can set the `OPTIONS` variable in the AFS initialization script to one of `$SMALL`, `$MEDIUM`, or `$LARGE`. The AFS initialization script uses one of these settings if the **afsd** options file named by the `AFSDOPT` variable does not exist. In the script as distributed, the `OPTIONS` variable is set to the value `$MEDIUM`.

  Note
  Do not set the `OPTIONS` variable to `$SMALL`, `$MEDIUM`, or `$LARGE` on a machine that uses a memory cache. The arguments it sets are appropriate only on a machine that uses a disk cache.

  The script (or on some system types the **afsd** options file named by the `AFSDOPT` variable) defines a value for each of `SMALL`, `MEDIUM`, and `LARGE` that sets **afsd** command arguments appropriately for client machines of different sizes:

  - `SMALL` is suitable for a small machine that serves one or two users and has approximately 8 MB of RAM and a 20-MB cache
  - `MEDIUM` is suitable for a medium-sized machine that serves two to six users and has 16 MB of RAM and a 40-MB cache
  - `LARGE` is suitable for a large machine that serves five to ten users and has 32 MB of RAM and a 100-MB cache

- You can choose not to create an **afsd** options file and to set the `OPTIONS` variable in the initialization script to a null value rather than to the default `$MEDIUM` value. You can then either set arguments directly on the **afsd** command line in the script, or set no arguments (and so accept default values for all Cache Manager parameters).

Note
If you are running on a Fedora or RHEL based system, the openafs-client initialization script behaves differently from that described above. It sources /etc/sysconfig/openafs, in which the AFSD_ARGS variable may be set to contain any, or all, of the afsd options detailed. Note that this script does not support setting an OPTIONS variable, or the SMALL, MEDIUM and LARGE methods of defining cache size

1. Create the local directory on which to mount the AFS filespace, by convention **/afs**. If the directory already exists, verify that it is empty.

```
# mkdir /afs
```

2. On non-package based Linux systems, copy the **afsd** options file from the **/usr/vice/etc** directory to the **/etc/sysconfig** directory, removing the **.conf** extension as you do so.

```
# cp /usr/vice/etc/afs.conf /etc/sysconfig/afs
```

3. Edit the machine's AFS initialization script or **afsd** options file to set appropriate values for **afsd** command parameters. The script resides in the indicated location on each system type:

   - On Fedora and RHEL systems, **/etc/sysconfg/openafs**
   - On non-package based Linux systems, **/etc/sysconfig/afs** (the **afsd** options file)
   - On Solaris systems, **/etc/init.d/afs**

   Use one of the methods described in the introduction to this section to add the following flags to the **afsd** command line. If you intend for the machine to remain an AFS client, also set any performance-related arguments you wish.

   - Add the **-memcache** flag if the machine is to use a memory cache.
   - Add the **-verbose** flag to display a trace of the Cache Manager's initialization on the standard output stream.

Note
In order to successfully complete the instructions in the remainder of this guide, it is important that the machine does not have a synthetic root (as discussed in Enabling Access to Foreign Cells). As some distributions ship with this enabled, it may be necessary to remove any occurences of the **-dynroot** and **-afsdb** options from both the AFS initialisation script and options file. If this functionality is required it may be renabled as detailed in Enabling Access to Foreign Cells.

2.21 Overview: Completing the Installation of the First AFS Machine

The machine is now configured as an AFS file server and client machine. In this final phase of the installation, you initialize the Cache Manager and then create the upper levels of your AFS filespace, among other procedures. The procedures are:

1. Verify that the initialization script works correctly, and incorporate it into the operating system's startup and shutdown sequence
2. Create and mount top-level volumes
3. Create and mount volumes to store system binaries in AFS
4. Enable access to foreign cells
5. Institute additional security measures
6. Remove client functionality if desired

2.22 Verifying the AFS Initialization Script

At this point you run the AFS initialization script to verify that it correctly invokes all of the necessary programs and AFS processes, and that they start correctly. The following are the relevant commands:

- The command that dynamically loads AFS modifications into the kernel, on some system types (not applicable if the kernel has AFS modifications built in)
- The **bosserver** command, which starts the BOS Server; it in turn starts the server processes for which you created entries in the **/usr/afs/local/BosConfig** file
- The **afsd** command, which initializes the Cache Manager

On system types that use a dynamic loader program, you must reboot the machine before running the initialization script, so that it can freshly load AFS modifications into the kernel.

If there are problems during the initialization, attempt to resolve them. The OpenAFS mailing lists can provide assistance if necessary.

1. Issue the **bos shutdown** command to shut down the AFS server processes other than the BOS Server. Include the **-wait** flag to delay return of the command shell prompt until all processes shut down completely.

   ```
   # /usr/afs/bin/bos shutdown <machine name> -wait
   ```

2. Issue the **ps** command to learn the **bosserver** process's process ID number (PID), and then the **kill** command to stop it.

   ```
   # ps appropriate_ps_options | grep bosserver
   # kill bosserver_PID
   ```

3. Issue the appropriate commands to run the AFS initialization script for this system type.

   On Linux systems:

   (a) Reboot the machine and log in again as the local superuser **root**.

   ```
   # cd /
   # shutdown -r now
   login: root
   Password: root_password
   ```

   (b) Run the AFS initialization scripts.

   ```
   # /etc/rc.d/init.d/openafs-client  start
   # /etc/rc.d/init.d/openafs-server  start
   ```

   On Solaris systems:

   (a) Reboot the machine and log in again as the local superuser **root**.

   ```
   # cd /
   # shutdown -i6 -g0 -y
   login: root
   Password: root_password
   ```

   (b) Run the AFS initialization script.

   ```
   # /etc/init.d/afs  start
   ```

4. Wait for the message that confirms that Cache Manager initialization is complete.

   On machines that use a disk cache, it can take a while to initialize the Cache Manager for the first time, because the **afsd** program must create all of the **V***n* files in the cache directory. Subsequent Cache Manager initializations do not take nearly as long, because the **V***n* files already exist.

5. If you are working with an existing cell which uses **kaserver** for authentication, please recall the note in Using this Appendix detailing the substitution of **kinit** and **aklog** with **klog**.

   As a basic test of correct AFS functioning, issue the **kinit** and **aklog** commands to authenticate as the **admin** user. Provide the password (*admin_passwd*) you defined in Initializing Cell Security.

   ```
   # kinit admin
   Password:  admin_passwd
   # aklog
   ```

6. Issue the **tokens** command to verify that the **aklog** command worked correctly. If it did, the output looks similar to the following example for the **example.com** cell, where **admin**'s AFS UID is 1. If the output does not seem correct, resolve the problem. Changes to the AFS initialization script are possibly necessary. The OpenAFS mailing lists can provide assistance as necessary.

   ```
   # tokens
   Tokens held by the Cache Manager:
   User's (AFS ID 1) tokens for afs@example.com [Expires May 22 11:52]
       --End of list--
   ```

7. Issue the **bos status** command to verify that the output for each process reads `Currently running normally`.

   ```
   # /usr/afs/bin/bos status <machine name>
   ```

8. Change directory to the local file system root (/) and issue the **fs checkvolumes** command.

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

2.23 Activating the AFS Initialization Script

Now that you have confirmed that the AFS initialization script works correctly, take the action necessary to have it run automatically at each reboot. Proceed to the instructions for your system type:

- Activating the Script on Linux Systems
- Activating the Script on Solaris Systems

2.23.1 Activating the Script on Linux Systems

1. Issue the **chkconfig** command to activate the **openafs-client** and **openafs-server** configuration variables. Based on the instruction in the AFS initialization file that begins with the string `#chkconfig`, the command automatically creates the symbolic links that incorporate the script into the Linux startup and shutdown sequence.

   ```
   # /sbin/chkconfig  --add openafs-client
   # /sbin/chkconfig  --add openafs-server
   ```

2. **(Optional)** There are now copies of the AFS initialization file in both the **/usr/vice/etc** and **/etc/rc.d/init.d** directories, and copies of the **afsd** options file in both the **/usr/vice/etc** and **/etc/sysconfig** directories. If you want to avoid potential confusion by guaranteeing that the two copies of each file are always the same, create a link between them. You can always retrieve the original script or options file from the AFS CD-ROM if necessary.

   ```
   # cd /usr/vice/etc
   # rm afs.rc afs.conf
   # ln -s  /etc/rc.d/init.d/afs  afs.rc
   # ln -s  /etc/sysconfig/afs  afs.conf
   ```

3. Proceed to Configuring the Top Levels of the AFS Filespace.

2.23.2 Activating the Script on Solaris Systems

1. Change to the **/etc/init.d** directory and issue the **ln -s** command to create symbolic links that incorporate the AFS initialization script into the Solaris startup and shutdown sequence.

```
# cd /etc/init.d
# ln -s ../init.d/afs /etc/rc3.d/S99afs
# ln -s ../init.d/afs /etc/rc0.d/K66afs
```

2. **(Optional)** There are now copies of the AFS initialization file in both the **/usr/vice/etc** and **/etc/init.d** directories. If you want to avoid potential confusion by guaranteeing that they are always the same, create a link between them. You can always retrieve the original script from the AFS CD-ROM if necessary.

```
# cd /usr/vice/etc
# rm afs.rc
# ln -s  /etc/init.d/afs  afs.rc
```

2.24 Configuring the Top Levels of the AFS Filespace

If you have not previously run AFS in your cell, you now configure the top levels of your cell's AFS filespace. If you have run a previous version of AFS, the filespace is already configured. Proceed to Storing AFS Binaries in AFS.

You created the **root.afs** volume in Starting the File Server, Volume Server, and Salvager, and the Cache Manager mounted it automatically on the local **/afs** directory when you ran the AFS initialization script in Verifying the AFS Initialization Script. You now set the access control list (ACL) on the **/afs** directory; creating, mounting, and setting the ACL are the three steps required when creating any volume.

After setting the ACL on the **root.afs** volume, you create your cell's **root.cell** volume, mount it as a subdirectory of the **/afs** directory, and set the ACL. Create both a read/write and a regular mount point for the **root.cell** volume. The read/write mount point enables you to access the read/write version of replicated volumes when necessary. Creating both mount points essentially creates separate read-only and read-write copies of your filespace, and enables the Cache Manager to traverse the filespace on a read-only path or read/write path as appropriate. For further discussion of these concepts, see the chapter in the *OpenAFS Administration Guide* about administering volumes.

Then replicate both the **root.afs** and **root.cell** volumes. 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.

When the **root.afs** volume is replicated, the Cache Manager is programmed to access its read-only version (**root.afs.readonly**) whenever possible. To make changes to the contents of the **root.afs** volume (when, for example, you mount another cell's **root.cell** volume at the second level in your filespace), you must mount the **root.afs** volume temporarily, make the changes, release the volume and remove the temporary mount point. For instructions, see Enabling Access to Foreign Cells.

1. Issue the **fs setacl** command to edit the ACL on the **/afs** directory. Add an entry that grants the **l** (**lookup**) and **r** (**read**) permissions to the **system:anyuser** group, to enable all AFS users who can reach your cell to traverse through the directory. If you prefer to enable access only to locally authenticated users, substitute the **system:authuser** group.

   Note that there is already an ACL entry that grants all seven access rights to the **system:administrators** group. It is a default entry that AFS places on every new volume's root directory.

   The top-level AFS directory, typically /afs, is a special case: when the client is configured to run in dynroot mode (e.g. **afsd -dynroot**, attempts to set the ACL on this directory will return **Connection timed out**. This is because the dynamically-generated root directory is not a part of the global AFS space, and cannot have an access control list set on it.

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

2. Issue the **vos create** command to create the **root.cell** volume. Then issue the **fs mkmount** command to mount it as a subdirectory of the **/afs** directory, where it serves as the root of your cell's local AFS filespace. Finally, issue the **fs setacl** command to create an ACL entry for the **system:anyuser** group (or **system:authuser** group).

   For the *partition name* argument, substitute the name of one of the machine's AFS server partitions (such as **/vicepa**). For the *cellname* argument, substitute your cell's fully-qualified Internet domain name (such as **example.com**).

   ```
   # /usr/afs/bin/vos create  <machine name> <partition name> root.cell
   # /usr/afs/bin/fs mkmount /afs/cellname  root.cell
   # /usr/afs/bin/fs setacl /afs/cellname  system:anyuser rl
   ```

3. **(Optional)** Create a symbolic link to a shortened cell name, to reduce the length of pathnames for users in the local cell. For example, in the **example.com** cell, **/afs/example** is a link to **/afs/example.com**.

   ```
   # cd /afs
   # ln -s  full_cellname  short_cellname
   ```

4. Issue the **fs mkmount** command to create a read/write mount point for the **root.cell** volume (you created a regular mount point in Step 2).

   By convention, the name of a read/write mount point begins with a period, both to distinguish it from the regular mount point and to make it visible only when the **-a** flag is used on the **ls** command.

   Change directory to **/usr/afs/bin** to make it easier to access the command binaries.

   ```
   # cd /usr/afs/bin
   # ./fs mkmount   /afs/.cellname   root.cell -rw
   ```

5. Issue the **vos addsite** command to define a replication site for both the **root.afs** and **root.cell** volumes. In each case, substitute for the *partition name* argument the partition where the volume's read/write version resides. When you install additional file server machines, it is a good idea to create replication sites on them as well.

   ```
   # ./vos addsite <machine name> <partition name> root.afs
   # ./vos addsite <machine name> <partition name> root.cell
   ```

6. Issue the **fs examine** command to verify that the Cache Manager can access both the **root.afs** and **root.cell** volumes, before you attempt to replicate them. The output lists each volume's name, volumeID number, quota, size, and the size of the partition that houses them. If you get an error message instead, do not continue before taking corrective action.

   ```
   # ./fs examine /afs
   # ./fs examine /afs/cellname
   ```

7. Issue the **vos release** command to release a replica of the **root.afs** and **root.cell** volumes to the sites you defined in Step 5.

   ```
   # ./vos release root.afs
   # ./vos release root.cell
   ```

8. Issue the **fs checkvolumes** to force the Cache Manager to notice that you have released read-only versions of the volumes, then issue the **fs examine** command again. This time its output mentions the read-only version of the volumes (**root.afs.readonly** and **root.cell.readonly**) instead of the read/write versions, because of the Cache Manager's bias to access the read-only version of the **root.afs** volume if it exists.

   ```
   # ./fs checkvolumes
   # ./fs examine /afs
   # ./fs examine /afs/cellname
   ```

2.25 Storing AFS Binaries in AFS

Note
Sites with existing binary distribution mechanisms, including those which use packaging systems such as RPM, may wish to skip this step, and use tools native to their operating system to manage AFS configuration information.

In the conventional configuration, you make AFS client binaries and configuration files available in the subdirectories of the **/usr/afsws** directory on client machines (**afsws** is an acronym for **AFS w***orks**tation*). You can conserve local disk space by creating **/usr/afsws** as a link to an AFS volume that houses the AFS client binaries and configuration files for this system type.

In this section you create the necessary volumes. The conventional location to which to link **/usr/afsws** is **/afs/***cellname***/***sysname***/usr/a** where *sysname* is the appropriate system type name as specified in the *OpenAFS Release Notes*. The instructions in Installing Additional Client Machines assume that you have followed the instructions in this section.

If you have previously run AFS in the cell, the volumes possibly already exist. If so, you need to perform Step 8 only.

The current working directory is still **/usr/afs/bin**, which houses the **fs** and **vos** command suite binaries. In the following commands, it is possible you still need to specify the pathname to the commands, depending on how your PATH environment variable is set.

1. Issue the **vos create** command to create volumes for storing the AFS client binaries for this system type. The following example instruction creates volumes called *sysname*, *sysname*.**usr**, and *sysname*.**usr.afsws**. Refer to the *OpenAFS Release Notes* to learn the proper value of *sysname* for this system type.

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

2. Issue the **fs mkmount** command to mount the newly created volumes. Because the **root.cell** volume is replicated, you must precede the *cellname* part of the pathname with a period to specify the read/write mount point, as shown. Then issue the **vos release** command to release a new replica of the **root.cell** volume, and the **fs checkvolumes** command to force the local Cache Manager to access them.

```
# fs mkmount -dir /afs/.cellname/sysname -vol sysname
# fs mkmount -dir /afs/.cellname/sysname/usr  -vol sysname.usr
# fs mkmount -dir /afs/.cellname/sysname/usr/afsws -vol sysname.usr.afsws
# vos release root.cell
# fs checkvolumes
```

3. Issue the **fs setacl** command to grant the **l** (**lookup**) and **r** (**read**) permissions to the **system:anyuser** group on each new directory's ACL.

```
# cd /afs/.cellname/sysname
# fs setacl  -dir  .  usr  usr/afsws  -acl  system:anyuser rl
```

4. Issue the **fs setquota** command to set an unlimited quota on the volume mounted at the **/afs/***cellname***/***sysname***/usr/afsws** directory. This enables you to copy all of the appropriate files from the CD-ROM into the volume without exceeding the volume's quota.

   If you wish, you can set the volume's quota to a finite value after you complete the copying operation. At that point, use the **vos examine** command to determine how much space the volume is occupying. Then issue the **fs setquota** command to set a quota that is slightly larger.

```
# fs setquota /afs/.cellname/sysname/usr/afsws  0
```

5. Unpack the distribution tarball into the **/tmp/afsdist** directory, if it is not already.

6. Copy the contents of the indicated directories from the distribution into the **/afs/***cellname***/***sysname***/usr/afsws** directory.

```
# cd /afs/.cellname/sysname/usr/afsws
# cp -rp /tmp/afsdist/sysname/bin  .
# cp -rp /tmp/afsdist/sysname/etc  .
# cp -rp /tmp/afsdist/sysname/include  .
# cp -rp /tmp/afsdist/sysname/lib  .
```

7. Create **/usr/afsws** on the local disk as a symbolic link to the directory **/afs/***cellname***/@sys/usr/afsws**. You can specify the actual system name instead of **@sys** if you wish, but the advantage of using **@sys** is that it remains valid if you upgrade this machine to a different system type.

```
# ln -s /afs/cellname/@sys/usr/afsws  /usr/afsws
```

8. **(Optional)** To enable users to issue commands from the AFS suites (such as **fs**) without having to specify a pathname to their binaries, include the **/usr/afsws/bin** and **/usr/afsws/etc** directories in the PATH environment variable you define in each user's shell initialization file (such as **.cshrc**).

2.26 Storing AFS Documents in AFS

The AFS distribution includes the following documents:

- *OpenAFS Release Notes*
- *OpenAFS Quick Beginnings*
- *OpenAFS User Guide*
- *OpenAFS Administration Reference*
- *OpenAFS Administration Guide*

Note
OpenAFS Documentation is not currently provided with all distributions, but may be downloaded separately from the OpenAFS website

---

The OpenAFS Documentation Distribution has a directory for each document format provided. The different formats are suitable for online viewing, printing, or both.

This section explains how to create and mount a volume to house the documents, making them available to your users. The recommended mount point for the volume is **/afs/***cellname***/afsdoc**. If you wish, you can create a link to the mount point on each client machine's local disk, called **/usr/afsdoc**. Alternatively, you can create a link to the mount point in each user's home directory. You can also choose to permit users to access only certain documents (most probably, the *OpenAFS User Guide*) by creating different mount points or setting different ACLs on different document directories.

The current working directory is still **/usr/afs/bin**, which houses the **fs** and **vos** command suite binaries you use to create and mount volumes. In the following commands, it is possible you still need to specify the pathname to the commands, depending on how your PATH environment variable is set.

1. Issue the **vos create** command to create a volume for storing the AFS documentation. Include the **-maxquota** argument to set an unlimited quota on the volume. This enables you to copy all of the appropriate files from the CD-ROM into the volume without exceeding the volume's quota.

   If you wish, you can set the volume's quota to a finite value after you complete the copying operations. At that point, use the **vos examine** command to determine how much space the volume is occupying. Then issue the **fs setquota** command to set a quota that is slightly larger.

```
# vos create  <machine name> <partition name>  afsdoc  -maxquota  0
```

2. Issue the **fs mkmount** command to mount the new volume. Because the **root.cell** volume is replicated, you must precede the *cellname* with a period to specify the read/write mount point, as shown. Then issue the **vos release** command to release a new replica of the **root.cell** volume, and the **fs checkvolumes** command to force the local Cache Manager to access them.

```
# fs mkmount -dir /afs/.cellname/afsdoc -vol afsdoc
# vos release root.cell
# fs checkvolumes
```

3. Issue the **fs setacl** command to grant the **rl** permissions to the **system:anyuser** group on the new directory's ACL.

```
# cd /afs/.cellname/afsdoc
# fs setacl  .  system:anyuser rl
```

4. Unpack the OpenAFS documentation distribution into the **/tmp/afsdocs** directory. You may use a different directory, in which case the location you use should be subsituted in the following examples. For instructions on unpacking the distribution, consult the documentation for your operating system's **tar** command.

5. Copy the AFS documents in one or more formats from the unpacked distribution into subdirectories of the **/afs/***cellname***/afsdoc** directory. Repeat the commands for each format.

```
# mkdir format_name
# cd format_name
# cp -rp /tmp/afsdocs/format  .
```

   If you choose to store the HTML version of the documents in AFS, note that in addition to a subdirectory for each document there are several files with a **.gif** extension, which enable readers to move easily between sections of a document. The file called **index.htm** is an introductory HTML page that contains a hyperlink to each of the documents. For online viewing to work properly, these files must remain in the top-level HTML directory (the one named, for example, **/afs/***cellname***/afsdoc/html**).

6. **(Optional)** If you believe it is helpful to your users to access the AFS documents in a certain format via a local disk directory, create **/usr/afsdoc** on the local disk as a symbolic link to the documentation directory in AFS (**/afs/***cellname***/afsdoc/***format*

```
# ln -s /afs/cellname/afsdoc/format_name /usr/afsdoc
```

   An alternative is to create a link in each user's home directory to the **/afs/***cellname***/afsdoc/***format_name* directory.

2.27 Storing System Binaries in AFS

You can also choose to store other system binaries in AFS volumes, such as the standard UNIX programs conventionally located in local disk directories such as **/etc**, **/bin**, and **/lib**. Storing such binaries in an AFS volume not only frees local disk space, but makes it easier to update binaries on all client machines.

The following is a suggested scheme for storing system binaries in AFS. It does not include instructions, but you can use the instructions in Storing AFS Binaries in AFS (which are for AFS-specific binaries) as a template.

Some files must remain on the local disk for use when AFS is inaccessible (during bootup and file server or network outages). The required binaries include the following:

- A text editor, network commands, and so on
- Files used during the boot sequence before the **afsd** program runs, such as initialization and configuration files, and binaries for commands that mount file systems
- Files used by dynamic kernel loader programs

In most cases, it is more secure to enable only locally authenticated users to access system binaries, by granting the **l** (**lookup**) and **r** (**read**) permissions to the **system:authuser** group on the ACLs of directories that contain the binaries. If users need to access a binary while unauthenticated, however, the ACL on its directory must grant those permissions to the **system:anyuser** group.

The following chart summarizes the suggested volume and mount point names for storing system binaries. It uses a separate volume for each directory. You already created a volume called *sysname* for this machine's system type when you followed the instructions in Storing AFS Binaries in AFS.

You can name volumes in any way you wish, and mount them at other locations than those suggested here. However, this scheme has several advantages:

- Volume names clearly identify volume contents
- Using the *sysname* prefix on every volume makes it is easy to back up all of the volumes together, because the AFS Backup System enables you to define sets of volumes based on a string included in all of their names
- It makes it easy to track related volumes, keeping them together on the same file server machine if desired
- There is a clear relationship between volume name and mount point name

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**

2.28 Enabling Access to Foreign Cells

With current OpenAFS releases, there exist a number of mechanisms for providing access to foreign cells. You may add mount points in your AFS filespace for each foreign cell you wish users to access, or you can enable a 'synthetic' AFS root, which contains mountpoints for either all AFS cells defined in the client machine's local **/usr/vice/etc/CellServDB**, or for all cells providing location information in the DNS.

2.28.1 Enabling a Synthetic AFS root

When a synthetic root is enabled, the client cache machine creates its own root.afs volume, rather than using the one provided with your cell. This allows clients to access all cells in the **CellServDB** file and, optionally, all cells registered in the DNS, without requiring system administrator action to enable this access. Using a synthetic root has the additional advantage that it allows a client to start its AFS service without a network available, as it is no longer necessary to contact a fileserver to obtain the root volume.

OpenAFS supports two complimentary mechanisms for creating the synthetic root. Starting the cache manager with the **-dynroot** option adds all cells listed in **/usr/vice/etc/CellServDB** to the client's AFS root. Adding the **-afsdb** option in addition to this enables DNS lookups for any cells that are not found in the client's CellServDB file. Both of these options are added to the AFS initialisation script, or options file, as detailed in Configuring the Cache Manager.

2.28.2 Adding foreign cells to a conventional root volume

In this section you create a mount point in your AFS filespace for the **root.cell** volume of each foreign cell that you want to enable your users to access. For users working on a client machine to access the cell, there must in addition be an entry for it in the client machine's local **/usr/vice/etc/CellServDB** file. (The instructions in Creating the Client CellServDB File suggest that you use the **CellServDB.sample** file included in the AFS distribution as the basis for your cell's client **CellServDB** file. The sample file lists all of the cells that had agreed to participate in the AFS global namespace at the time your AFS CD-ROM was created. As mentioned in that section, the AFS Product Support group also maintains a copy of the file, updating it as necessary.)

The chapter in the *OpenAFS Administration Guide* about cell administration and configuration issues discusses the implications of participating in the global AFS namespace. The chapter about administering client machines explains how to maintain knowledge of foreign cells on client machines, and includes suggestions for maintaining a central version of the file in AFS.

1. Issue the **fs mkmount** command to mount each foreign cell's **root.cell** volume on a directory called **/afs/***foreign_cell*. Because the **root.afs** volume is replicated, you must create a temporary mount point for its read/write version in a directory to which you have write access (such as your cell's **/afs/.***cellname* directory). Create the mount points, issue the **vos release** command to release new replicas to the read-only sites for the **root.afs** volume, and issue the **fs checkvolumes** command to force the local Cache Manager to access the new replica.

   Note
   You need to issue the **fs mkmount** command only once for each foreign cell's **root.cell** volume. You do not need to repeat the command on each client machine.

   Substitute your cell's name for *cellname*.

   ```
   # cd /afs/.cellname
   # /usr/afs/bin/fs  mkmount  temp  root.afs
   ```

   Repeat the **fs mkmount** command for each foreign cell you wish to mount at this time.

   ```
   # /usr/afs/bin/fs mkmount temp/foreign_cell root.cell -c foreign_cell
   ```

   Issue the following commands only once.

   ```
   # /usr/afs/bin/fs rmmount temp
   # /usr/afs/bin/vos release root.afs
   # /usr/afs/bin/fs checkvolumes
   ```

2. If this machine is going to remain an AFS client after you complete the installation, verify that the local **/usr/vice/etc/CellServDB** file includes an entry for each foreign cell.

   For each cell that does not already have an entry, complete the following instructions:

   (a) Create an entry in the **CellServDB** file. Be sure to comply with the formatting instructions in Creating the Client CellServDB File.

   (b) Issue the **fs newcell** command to add an entry for the cell directly to the list that the Cache Manager maintains in kernel memory. Provide each database server machine's fully qualified hostname.

   ```
   # /usr/afs/bin/fs newcell <foreign_cell> <dbserver1>    \
           [<dbserver2>] [<dbserver3>]
   ```

   (c) If you plan to maintain a central version of the **CellServDB** file (the conventional location is **/afs/***cellname***/common/etc/Cel** create it now as a copy of the local **/usr/vice/etc/CellServDB** file. Verify that it includes an entry for each foreign cell you want your users to be able to access.

   ```
   # mkdir common
   # mkdir common/etc
   # cp  /usr/vice/etc/CellServDB  common/etc
   # /usr/afs/bin/vos release root.cell
   ```

3. Issue the **ls** command to verify that the new cell's mount point is visible in your filespace. The output lists the directories at the top level of the new cell's AFS filespace.

```
# ls /afs/foreign_cell
```

4. If you wish to participate in the global AFS namespace, and only intend running one database server, please register your cell with grand.central.org at this time. To do so, email the **CellServDB** fragment describing your cell, together with a contact name and email address for any queries, to cellservdb@grand.central.org. If you intend on deploying multiple database servers, please wait until you have installed all of them before registering your cell.

5. If you wish to allow your cell to be located through DNS lookups, at this time you should also add the necessary configuration to your DNS.

   AFS database servers may be located by creating AFSDB records in the DNS for the domain name corresponding to the name of your cell. It's outside the scope of this guide to give an indepth description of managing, or configuring, your site's DNS. You should consult the documentation for your DNS server for further details on AFSDB records.

2.29 Improving Cell Security

This section discusses ways to improve the security of AFS data in your cell. Also see the chapter in the *OpenAFS Administration Guide* about configuration and administration issues.

2.29.1 Controlling root Access

As on any machine, it is important to prevent unauthorized users from logging onto an AFS server or client machine as the local superuser **root**. Take care to keep the **root** password secret.

The local **root** superuser does not have special access to AFS data through the Cache Manager (as members of the **system:administrators** group do), but it does have the following privileges:

- On client machines, the ability to issue commands from the **fs** suite that affect AFS performance
- On server machines, the ability to disable authorization checking, or to install rogue process binaries

2.29.2 Controlling System Administrator Access

Following are suggestions for managing AFS administrative privilege:

- Create an administrative account for each administrator named something like *username***.admin**. Administrators authenticate under these identities only when performing administrative tasks, and destroy the administrative tokens immediately after finishing the task (either by issuing the **unlog** command, or the **kinit** and **aklog** commands to adopt their regular identity).
- Set a short ticket lifetime for administrator accounts (for example, 20 minutes) by using the facilities of your KDC. For instance, with a MIT Kerberos KDC, this can be performed using the **--max-ticket-life** argument to the **kadmin modify_principal** command. Do not however, use a short lifetime for users who issue long-running **backup** commands.
- Limit the number of system administrators in your cell, especially those who belong to the **system:administrators** group. By default they have all ACL rights on all directories in the local AFS filespace, and therefore must be trusted not to examine private files.
- Limit the use of system administrator accounts on machines in public areas. It is especially important not to leave such machines unattended without first destroying the administrative tokens.
- Limit the use by administrators of standard UNIX commands that make connections to remote machines (such as the **telnet** utility). Many of these programs send passwords across the network without encrypting them.

2.29.3 Protecting Sensitive AFS Directories

Some subdirectories of the **/usr/afs** directory contain files crucial to cell security. Unauthorized users must not read or write to these files because of the potential for misuse of the information they contain.

As the BOS Server initializes for the first time on a server machine, it creates several files and directories (as mentioned in Starting the BOS Server). It sets their owner to the local superuser **root** and sets their mode bits to enable writing by the owner only; in some cases, it also restricts reading.

At each subsequent restart, the BOS Server checks that the owner and mode bits on these files are still set appropriately. If they are not, it write the following message to the **/usr/afs/logs/BosLog** file:

```
Bosserver reports inappropriate access on server directories
```

The BOS Server does not reset the mode bits, which enables you to set alternate values if you wish.

The following charts lists the expected mode bit settings. A question mark indicates that the BOS Server does not check that mode bit.

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

2.30 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 means that you cannot use this machine to access AFS files.

1. Remove the files from the **/usr/vice/etc** directory. The command does not remove the directory for files used by the dynamic kernel loader program, if it exists on this system type. Those files are still needed on a server-only machine.

```
# cd /usr/vice/etc
# rm  *
# rm -rf  C
```

2. Create symbolic links to the **ThisCell** and **CellServDB** files in the **/usr/afs/etc** directory. This makes it possible to issue commands from the AFS command suites (such as **bos** and **fs**) on this machine.

```
# ln -s /usr/afs/etc/ThisCell ThisCell
# ln -s /usr/afs/etc/CellServDB CellServDB
```

3. Reboot the machine. Most system types use the **shutdown** command, but the appropriate options vary.

```
# cd /
# shutdown appropriate_options
```

Chapter 3

Installing Additional Server Machines

Instructions for the following procedures appear in the indicated section of this chapter.

- Installing an Additional File Server Machine
- Installing Database Server Functionality
- Removing Database Server Functionality

The instructions make the following assumptions.

- You have already installed your cell's first file server machine by following the instructions in Installing the First AFS Machine
- You are logged in as the local superuser **root**
- You are working at the console
- A standard version of one of the operating systems supported by the current version of AFS is running on the machine
- You can access the data on the OpenAFS Binary Distribution for your operating system, either on the local filesystem or via an NFS mount of the distribution's contents.

3.1 Installing an Additional File Server Machine

The procedure for installing a new file server machine is similar to installing the first file server machine in your cell. There are a few parts of the installation that differ depending on whether the machine is the same AFS system type as an existing file server machine or is the first file server machine of its system type in your cell. The differences mostly concern the source for the needed binaries and files, and what portions of the Update Server you install:

- On a new system type, you must load files and binaries from the OpenAFS distribution. You may install the server portion of the Update Server to make this machine the binary distribution machine for its system type.
- On an existing system type, you can copy files and binaries from a previously installed file server machine, rather than from the OpenAFS distribution. You may install the client portion of the Update Server to accept updates of binaries, because a previously installed machine of this type was installed as the binary distribution machine.
- On some system types, distribtution of the appropriate binaries may be acheived using the system's own package management system. In these cases, it is recommended that this system is used, rather than installing the binaries by hand.

These instructions are brief; for more detailed information, refer to the corresponding steps in Installing the First AFS Machine.

To install a new file server machine, perform the following procedures:

1. Copy needed binaries and files onto this machine's local disk, as required.
2. Incorporate AFS modifications into the kernel
3. Configure partitions for storing volumes
4. Replace the standard **fsck** utility with the AFS-modified version on some system types
5. Start the Basic OverSeer (BOS) Server
6. Start the appropriate portion of the Update Server, if required
7. Start the **fs** process, which incorporates three component processes: the File Server, Volume Server, and Salvager

After completing the instructions in this section, you can install database server functionality on the machine according to the instructions in Installing Database Server Functionality.

3.1.1 Creating AFS Directories and Performing Platform-Specific Procedures

If your operating systems AFS distribution is supplied as packages, such as .rpms or .debs, you should just install those packages as detailed in the previous chapter.

Create the **/usr/afs** and **/usr/vice/etc** directories on the local disk. Subsequent instructions copy files from the AFS distribution into them, at the appropriate point for each system type.

```
# mkdir /usr/afs
# mkdir /usr/afs/bin
# mkdir /usr/vice
# mkdir /usr/vice/etc
# mkdir /tmp/afsdist
```

As on the first file server machine, the initial procedures in installing an additional file server machine vary a good deal from platform to platform. For convenience, the following sections group together all of the procedures for a system type. Most of the remaining procedures are the same on every system type, but differences are noted as appropriate. The initial procedures are the following.

- Incorporate AFS modifications into the kernel, either by using a dynamic kernel loader program or by building a new static kernel
- Configure server partitions to house AFS volumes
- Replace the operating system vendor's **fsck** program with a version that recognizes AFS data
- If the machine is to remain an AFS client machine, modify the machine's authentication system so that users obtain an AFS token as they log into the local file system. (For this procedure only, the instructions direct you to the platform-specific section in Installing the First AFS Machine.)

To continue, proceed to the section for this system type:

- Getting Started on Linux Systems
- Getting Started on Solaris Systems

3.1.1.1 Getting Started on Linux Systems

Begin by running the AFS initialization script to call the **insmod** program, which dynamically loads AFS modifications into the kernel. Then create partitions for storing AFS volumes. You do not need to replace the Linux **fsck** program.

The procedure for starting up OpenAFS depends upon your distribution

1. For Fedora and RedHat Enterprise Linux systems (or their derivateds), download and install the RPM set for your operating system from the OpenAFS distribution site. You will need the **openafs** and **openafs-server** packages, along with an **openafs-kernel** package matching your current, running, kernel. If you wish to install client functionality, you will also require the **openafs-client** package.

   You can find the version of your current kernel by running

   ```
   # uname -r
   2.6.20-1.2933.fc6
   ```

   Once downloaded, the packages may be installed with the **rpm** command

   ```
   # rpm -U openafs-* openafs-client-* openafs-server-* openafs-kernel-*
   ```

2. For systems which are provided as a tarball, or built from source, unpack the distribution tarball. The examples below assume that you have unpacked the files into the **/tmp/afsdist**directory. If you pick a different location, substitute this in all of the following examples. Once you have unpacked the distribution, change directory as indicated.

   ```
   # cd /tmp/afsdist/linux/dest/root.client/usr/vice/etc
   ```

   Copy the AFS kernel library files to the local **/usr/vice/etc/modload** directory. The filenames for the libraries have the format **libafs-***version***.o**, where *version* indicates the kernel build level. The string **.mp** in the *version* indicates that the file is appropriate for machines running a multiprocessor kernel.

   ```
   # cp -rp  modload  /usr/vice/etc
   ```

   Copy the AFS initialization script to the local directory for initialization files (by convention, **/etc/rc.d/init.d** on Linux machines). Note the removal of the **.rc** extension as you copy the script.

   ```
   # cp -p   afs.rc  /etc/rc.d/init.d/afs
   ```

3. Create a directory called **/vicep***xx* for each AFS server partition you are configuring (there must be at least one). Repeat the command for each partition.

   ```
   # mkdir /vicepxx
   ```

4. Add a line with the following format to the file systems registry file, **/etc/fstab**, for each directory just created. The entry maps the directory name to the disk partition to be mounted on it.

   ```
   /dev/disk  /vicepxx  ext2  defaults  0  2
   ```

   The following is an example for the first partition being configured.

   ```
   /dev/sda8 /vicepa ext2 defaults 0 2
   ```

5. Create a file system on each partition that is to be mounted at a **/vicep***xx* directory. The following command is probably appropriate, but consult the Linux documentation for more information.

   ```
   # mkfs -v /dev/disk
   ```

6. Mount each partition by issuing either the **mount -a** command to mount all partitions at once or the **mount** command to mount each partition in turn.

7. If the machine is to remain an AFS client, incorporate AFS into its authentication system, following the instructions in Enabling AFS Login on Linux Systems.

8. Proceed to Starting Server Programs.

3.1.1.2 Getting Started on Solaris Systems

Begin by running the AFS initialization script to call the **modload** program, which dynamically loads AFS modifications into the kernel. Then configure partitions and replace the Solaris **fsck** program with a version that correctly handles AFS volumes.

1. Unpack the OpenAFS Solaris distribution tarball. The examples below assume that you have unpacked the files into the **/tmp/afsdist** directory. If you pick a diferent location, substitute this in all of the following exmaples. Once you have unpacked the distribution, change directory as indicated.

```
# cd  /tmp/afsdist/sun4x_56/dest/root.client/usr/vice/etc
```

2. Copy the AFS initialization script to the local directory for initialization files (by convention, **/etc/init.d** on Solaris machines). Note the removal of the **.rc** extension as you copy the script.

```
# cp -p  afs.rc  /etc/init.d/afs
```

3. Copy the appropriate AFS kernel library file to the local file **/kernel/fs/afs**.

   If the machine is running Solaris 11 on the x86_64 platform:

```
# cp -p modload/libafs64.o /kernel/drv/amd64/afs
```

   If the machine is running Solaris 10 on the x86_64 platform:

```
# cp -p modload/libafs64.o /kernel/fs/amd64/afs
```

   If the machine is running Solaris 2.6 or the 32-bit version of Solaris 7, its kernel supports NFS server functionality, and the **nfsd** process is running:

```
# cp -p modload/libafs.o /kernel/fs/afs
```

   If the machine is running Solaris 2.6 or the 32-bit version of Solaris 7, and its kernel does not support NFS server functionality or the **nfsd** process is not running:

```
# cp -p modload/libafs.nonfs.o /kernel/fs/afs
```

   If the machine is running the 64-bit version of Solaris 7, its kernel supports NFS server functionality, and the **nfsd** process is running:

```
# cp -p modload/libafs64.o /kernel/fs/sparcv9/afs
```

   If the machine is running the 64-bit version of Solaris 7, and its kernel does not support NFS server functionality or the **nfsd** process is not running:

```
# cp -p modload/libafs64.nonfs.o /kernel/fs/sparcv9/afs
```

4. Run the AFS initialization script to load AFS modifications into the kernel. You can ignore any error messages about the inability to start the BOS Server or the Cache Manager or AFS client.

```
# /etc/init.d/afs start
```

   When an entry called `afs` does not already exist in the local **/etc/name_to_sysnum** file, the script automatically creates it and reboots the machine to start using the new version of the file. If this happens, log in again as the superuser **root** after the reboot and run the initialization script again. This time the required entry exists in the **/etc/name_to_sysnum** file, and the **modload** program runs.

```
login: root
Password: root_password
# /etc/init.d/afs start
```

5. Create the **/usr/lib/fs/afs** directory to house the AFS-modified **fsck** program and related files.

```
# mkdir /usr/lib/fs/afs
# cd /usr/lib/fs/afs
```

6. Copy the **vfsck** binary to the newly created directory, changing the name as you do so.

```
# cp  /cdrom/sun4x_56/dest/root.server/etc/vfsck  fsck
```

7. Working in the **/usr/lib/fs/afs** directory, create the following links to Solaris libraries:

```
# ln -s /usr/lib/fs/ufs/clri
# ln -s /usr/lib/fs/ufs/df
# ln -s /usr/lib/fs/ufs/edquota
# ln -s /usr/lib/fs/ufs/ff
# ln -s /usr/lib/fs/ufs/fsdb
# ln -s /usr/lib/fs/ufs/fsirand
# ln -s /usr/lib/fs/ufs/fstyp
# ln -s /usr/lib/fs/ufs/labelit
# ln -s /usr/lib/fs/ufs/lockfs
# ln -s /usr/lib/fs/ufs/mkfs
# ln -s /usr/lib/fs/ufs/mount
# ln -s /usr/lib/fs/ufs/ncheck
# ln -s /usr/lib/fs/ufs/newfs
# ln -s /usr/lib/fs/ufs/quot
# ln -s /usr/lib/fs/ufs/quota
# ln -s /usr/lib/fs/ufs/quotaoff
# ln -s /usr/lib/fs/ufs/quotaon
# ln -s /usr/lib/fs/ufs/repquota
# ln -s /usr/lib/fs/ufs/tunefs
# ln -s /usr/lib/fs/ufs/ufsdump
# ln -s /usr/lib/fs/ufs/ufsrestore
# ln -s /usr/lib/fs/ufs/volcopy
```

8. Append the following line to the end of the file **/etc/dfs/fstypes**.

```
afs AFS Utilities
```

9. Edit the **/sbin/mountall** file, making two changes.

   - Add an entry for AFS to the `case` statement for option 2, so that it reads as follows:

```
case "$2" in
ufs)    foptions="-o p"
        ;;
afs)    foptions="-o p"
        ;;
s5)     foptions="-y -t /var/tmp/tmp$$ -D"
        ;;
*)      foptions="-y"
        ;;
```

   - Edit the file so that all AFS and UFS partitions are checked in parallel. Replace the following section of code:

```
# For  fsck purposes, we make a distinction between ufs and
# other file systems
#
if [ "$fstype" = "ufs" ]; then
     ufs_fscklist="$ufs_fscklist $fsckdev"
     saveentry $fstype "$OPTIONS" $special $mountp
     continue
fi
```

with the following section of code:

```
# For fsck purposes, we make a distinction between ufs/afs
# and other file systems.
#
if [ "$fstype" = "ufs" -o "$fstype" = "afs" ]; then
     ufs_fscklist="$ufs_fscklist $fsckdev"
     saveentry $fstype "$OPTIONS" $special $mountp
     continue
fi
```

10. Create a directory called **/vicep***xx* for each AFS server partition you are configuring (there must be at least one). Repeat the command for each partition.

```
# mkdir /vicepxx
```

11. Add a line with the following format to the file systems registry file, **/etc/vfstab**, for each partition to be mounted on a directory created in the previous step. Note the value `afs` in the fourth field, which tells Solaris to use the AFS-modified **fsck** program on this partition.

```
/dev/dsk/disk   /dev/rdsk/disk   /vicepxx   afs   boot_order  yes
```

The following is an example for the first partition being configured.

```
/dev/dsk/c0t6d0s1 /dev/rdsk/c0t6d0s1 /vicepa afs 3 yes
```

12. Create a file system on each partition that is to be mounted at a **/vicep***xx* directory. The following command is probably appropriate, but consult the Solaris documentation for more information.

```
# newfs -v /dev/rdsk/disk
```

13. Issue the **mountall** command to mount all partitions at once.
14. If the machine is to remain an AFS client, incorporate AFS into its authentication system, following the instructions in Enabling AFS Login and Editing the File Systems Clean-up Script on Solaris Systems.
15. Proceed to Starting Server Programs.

3.1.2 Starting Server Programs

In this section you initialize the BOS Server, the Update Server, and the **fs** process. You begin by copying the necessary server files to the local disk.

1. Copy file server binaries to the local **/usr/afs/bin** directory.
   - On a machine of an existing system type, you can either copy files from the OpenAFS binary distribution or use a remote file transfer protocol to copy files from an existing server machine of the same system type. To load from the binary distribution, see the instructions just following for a machine of a new system type. If using a remote file transfer protocol, copy the complete contents of the existing server machine's **/usr/afs/bin** directory.
   - If you are working from a tarball distribtion, rather than one distributed in a packaged format, you must use the following instructions to copy files from the OpenAFS Binary Distribution.
     (a) Unpack the distribution tarball. The examples below assume that you have unpacked the files into the **/tmp/afsdist** directory. If you pick a different location, substitute this in all of the following examples.
     (b) Copy files from the distribution to the local **/usr/afs** directory.

```
# cd /tmp/afsdist/sysname/root.server/usr/afs
# cp -rp  *  /usr/afs
```

2. Copy the contents of the **/usr/afs/etc** directory from an existing file server machine, using a remote file transfer protocol such as **sftp** or **scp**. If you use a system control machine, it is best to copy the contents of its **/usr/afs/etc** directory. If you choose not to run a system control machine, copy the directory's contents from any existing file server machine.

3. Change to the **/usr/afs/bin** directory and start the BOS Server (**bosserver** process). Include the **-noauth** flag to prevent the AFS processes from performing authorization checking. This is a grave compromise of security; finish the remaining instructions in this section in an uninterrupted pass.

```
# cd /usr/afs/bin
# ./bosserver -noauth
```

4. If you run a system control machine, create the **upclientetc** process as an instance of the client portion of the Update Server. It accepts updates of the common configuration files stored in the system control machine's **/usr/afs/etc** directory from the **upserver** process (server portion of the Update Server) running on that machine. The cell's first file server machine was installed as the system control machine in Starting the Server Portion of the Update Server. (If you do not run a system control machine, you must update the contents of the **/usr/afs/etc** directory on each file server machine, using the appropriate **bos** commands.)

   By default, the Update Server performs updates every 300 seconds (five minutes). Use the **-t** argument to specify a different number of seconds. For the *machine name* argument, substitute the name of the machine you are installing. The command appears on multiple lines here only for legibility reasons.

```
# ./bos create  <machine name> upclientetc simple  \
      "/usr/afs/bin/upclient  <system control machine>  \
      [-t  <time>]  /usr/afs/etc" -cell  <cell name>  -noauth
```

5. Create an instance of the Update Server to handle distribution of the file server binaries stored in the **/usr/afs/bin** directory. If your architecture using a package management system such as 'rpm' or 'apt' to maintain its binaries, note that distributing binaries via this system may interfere with your local package management tools.

   - If this is the first file server machine of its AFS system type, create the **upserver** process as an instance of the server portion of the Update Server. It distributes its copy of the file server process binaries to the other file server machines of this system type that you install in future. Creating this process makes this machine the binary distribution machine for its type.

```
# ./bos create  <machine name> upserver  simple  \
      "/usr/afs/bin/upserver -clear /usr/afs/bin"   \
      -cell <cell name>  -noauth
```

   - If this machine is an existing system type, create the **upclientbin** process as an instance of the client portion of the Update Server. It accepts updates of the AFS binaries from the **upserver** process running on the binary distribution machine for its system type. For distribution to work properly, the **upserver** process must already by running on that machine.

     Use the **-clear** argument to specify that the **upclientbin** process requests unencrypted transfer of the binaries in the **/usr/afs/bin** directory. Binaries are not sensitive and encrypting them is time-consuming.

     By default, the Update Server performs updates every 300 seconds (five minutes). Use the **-t** argument to specify an different number of seconds.

```
# ./bos create  <machine name> upclientbin simple  \
      "/usr/afs/bin/upclient <binary distribution machine>   \
      [-t <time>] -clear /usr/afs/bin" -cell <cell name>  -noauth
```

6. Issue the **bos create** command to start the **fs** process or the **dafs** process, depending on if you want to run the Demand-Attach File Server or not. See Appendix C, The Demand-Attach File Server for more information on whether you want to run it or not.

   - If you do not want to run the Demand-Attach File Server, start the **fs** process, which binds together the File Server, Volume Server, and Salvager.

```
# ./bos create  <machine name> fs fs    \
      /usr/afs/bin/fileserver /usr/afs/bin/volserver  \
      /usr/afs/bin/salvager -cell <cell name>  -noauth
```

- If you want to run the Demand-Attach File Server, start the **dafs** process, which binds together the File Server, Volume Server, Salvage Server, and Salvager.

```
# ./bos create  <machine name> dafs dafs    \
      /usr/afs/bin/dafileserver /usr/afs/bin/davolserver  \
      /usr/afs/bin/salvageserver  \
      /usr/afs/bin/dasalvager -cell <cell name>  -noauth
```

3.1.3 Installing Client Functionality

If you want this machine to be a client as well as a server, follow the instructions in this section. Otherwise, skip to Completing the Installation.

Begin by loading the necessary client files to the local disk. Then create the necessary configuration files and start the Cache Manager. For more detailed explanation of the procedures involved, see the corresponding instructions in Installing the First AFS Machine (in the sections following Overview: Installing Client Functionality).

If another AFS machine of this machine's system type exists, the AFS binaries are probably already accessible in your AFS filespace (the conventional location is **/afs/***cellname*/*sysname***/usr/afsws**). If not, or if this is the first AFS machine of its type, copy the AFS binaries for this system type into an AFS volume by following the instructions in Storing AFS Binaries in AFS. Because this machine is not yet an AFS client, you must perform the procedure on an existing AFS machine. However, remember to perform the final step (linking the local directory **/usr/afsws** to the appropriate location in the AFS file tree) on this machine itself. If you also want to create AFS volumes to house UNIX system binaries for the new system type, see Storing System Binaries in AFS.

1. Copy client binaries and files to the local disk.

   - On a machine of an existing system type, you can either load files from the OpenAFS Binary Distribution or use a remote file transfer protocol to copy files from an existing server machine of the same system type. To load from the binary distribution, see the instructions just following for a machine of a new system type. If using a remote file transfer protocol, copy the complete contents of the existing client machine's **/usr/vice/etc** directory.
   - On a machine of a new system type, you must use the following instructions to copy files from the OpenAFS Binary Distribution. If your distribution is provided in a packaged format, then simply installing the packages will perform the necessary actions.

     (a) Unpack the distribution tarball. The examples below assume that you have unpacked the files into the **/tmp/afsdist** directory. If you pick a different location, substitute this in all of the following examples.

     (b) Copy files to the local **/usr/vice/etc** directory.
     This step places a copy of the AFS initialization script (and related files, if applicable) into the **/usr/vice/etc** directory. In the preceding instructions for incorporating AFS into the kernel, you copied the script directly to the operating system's conventional location for initialization files. When you incorporate AFS into the machine's startup sequence in a later step, you can choose to link the two files.
     On some system types that use a dynamic kernel loader program, you previously copied AFS library files into a subdirectory of the **/usr/vice/etc** directory. On other system types, you copied the appropriate AFS library file directly to the directory where the operating system accesses it. The following commands do not copy or recopy the AFS library files into the **/usr/vice/etc** directory, because on some system types the library files consume a large amount of space. If you want to copy them, add the **-r** flag to the first **cp** command and skip the second **cp** command.

```
# cd /tmp/afsdist/sysname/root.client/usr/vice/etc
# cp -p  *  /usr/vice/etc
# cp -rp  C  /usr/vice/etc
```

2. Change to the **/usr/vice/etc** directory and create the **ThisCell** file as a copy of the **/usr/afs/etc/ThisCell** file. You must first remove the symbolic link to the **/usr/afs/etc/ThisCell** file that the BOS Server created automatically in Starting Server Programs.

```
# cd  /usr/vice/etc
# rm  ThisCell
# cp  /usr/afs/etc/ThisCell  ThisCell
```

3. Remove the symbolic link to the **/usr/afs/etc/CellServDB** file.

```
# rm  CellServDB
```

4. Create the **/usr/vice/etc/CellServDB** file. Use a network file transfer program such as **sftp** or **scp** to copy it from one of the following sources, which are listed in decreasing order of preference:

   - Your cell's central **CellServDB** source file (the conventional location is **/afs/***cellname***/common/etc/CellServDB**)
   - The global **CellServDB** file maintained at grand.central.org
   - An existing client machine in your cell
   - The **CellServDB.sample** file included in the *sysname***/root.client/usr/vice/etc** directory of each OpenAFS distribution; add an entry for the local cell by following the instructions in Creating the Client CellServDB File

5. Create the **cacheinfo** file for either a disk cache or a memory cache. For a discussion of the appropriate values to record in the file, see Configuring the Cache.

   To configure a disk cache, issue the following commands. If you are devoting a partition exclusively to caching, as recommended, you must also configure it, make a file system on it, and mount it at the directory created in this step.

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

   To configure a memory cache:

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

6. Create the local directory on which to mount the AFS filespace, by convention **/afs**. If the directory already exists, verify that it is empty.

```
# mkdir /afs
```

7. On non-packaged Linux systems, copy the **afsd** options file from the **/usr/vice/etc** directory to the **/etc/sysconfig** directory, removing the **.conf** extension as you do so.

```
# cp /usr/vice/etc/afs.conf /etc/sysconfig/afs
```

8. Edit the machine's AFS initialization script or **afsd** options file to set appropriate values for **afsd** command parameters. The script resides in the indicated location on each system type:

   - On Fedora and RHEL systems, **/etc/sysconfig/openafs**. Note that this file has a different format from a standard afsd options file.
   - On non-packaged Linux systems, **/etc/sysconfig/afs** (the **afsd** options file)
   - On Solaris systems, **/etc/init.d/afs**

   Use one of the methods described in Configuring the Cache Manager to add the following flags to the **afsd** command line. If you intend for the machine to remain an AFS client, also set any performance-related arguments you wish.

   - Add the **-memcache** flag if the machine is to use a memory cache.
   - Add the **-verbose** flag to display a trace of the Cache Manager's initialization on the standard output stream.
   - Add the **--dynroot** or **--afsdb** options if you wish to have a synthetic AFS root, as discussed in Enabling Access to Foreign Cells

9. If appropriate, follow the instructions in Storing AFS Binaries in AFS to copy the AFS binaries for this system type into an AFS volume. See the introduction to this section for further discussion.

3.1.4 Completing the Installation

At this point you run the machine's AFS initialization script to verify that it correctly loads AFS modifications into the kernel and starts the BOS Server, which starts the other server processes. If you have installed client files, the script also starts the Cache Manager. If the script works correctly, perform the steps that incorporate it into the machine's startup and shutdown sequence. If there are problems during the initialization, attempt to resolve them. The AFS Product Support group can provide assistance if necessary.

If the machine is configured as a client using a disk cache, it can take a while for the **afsd** program to create all of the **V**n files in the cache directory. Messages on the console trace the initialization process.

1. Issue the **bos shutdown** command to shut down the AFS server processes other than the BOS Server. Include the **-wait** flag to delay return of the command shell prompt until all processes shut down completely.

```
# /usr/afs/bin/bos shutdown <machine name> -wait
```

2. Issue the **ps** command to learn the BOS Server's process ID number (PID), and then the **kill** command to stop the **bosserver** process.

```
# ps appropriate_ps_options | grep bosserver
# kill bosserver_PID
```

3. Run the AFS initialization script by issuing the appropriate commands for this system type.

   On Fedora or RHEL Linux systems:

   (a) Reboot the machine and log in again as the local superuser **root**.

```
# cd /
# shutdown -r now
login: root
Password: root_password
```

   (b) Run the OpenAFS initialization scripts.

```
# /etc/rc.d/init.d/openafs-client  start
# /etc/rc.d/init.d/openafs-server  start
```

   (c) Issue the **chkconfig** command to activate the **openafs-client** and **openafs-server** configuration variables. Based on the instruction in the AFS initialization files that begins with the string `#chkconfig`, the command automatically creates the symbolic links that incorporate the script into the Linux startup and shutdown sequence.

```
# /sbin/chkconfig  --add openafs-client
# /sbin/chkconfig  --add openafs-server
```

   On Linux systems:

   (a) Reboot the machine and log in again as the local superuser **root**.

```
# cd /
# shutdown -r now
login: root
Password: root_password
```

   (b) Run the OpenAFS initialization script.

```
# /etc/rc.d/init.d/afs  start
```

   (c) Issue the **chkconfig** command to activate the **afs** configuration variable. Based on the instruction in the AFS initialization file that begins with the string `#chkconfig`, the command automatically creates the symbolic links that incorporate the script into the Linux startup and shutdown sequence.

```
# /sbin/chkconfig  --add afs
```

(d) **(Optional)** There are now copies of the AFS initialization file in both the **/usr/vice/etc** and **/etc/rc.d/init.d** directories, and copies of the **afsd** options file in both the **/usr/vice/etc** and **/etc/sysconfig** directories. If you want to avoid potential confusion by guaranteeing that the two copies of each file are always the same, create a link between them. You can always retrieve the original script or options file from the AFS CD-ROM if necessary.

```
# cd /usr/vice/etc
# rm afs.rc afs.conf
# ln -s  /etc/rc.d/init.d/afs  afs.rc
# ln -s  /etc/sysconfig/afs  afs.conf
```

(e) Proceed to Step 4.

On Solaris systems:

(a) Reboot the machine and log in again as the local superuser **root**.

```
# cd /
# shutdown -i6 -g0 -y
login: root
Password: root_password
```

(b) Run the AFS initialization script.

```
# /etc/init.d/afs  start
```

(c) Change to the **/etc/init.d** directory and issue the **ln -s** command to create symbolic links that incorporate the AFS initialization script into the Solaris startup and shutdown sequence.

```
# cd /etc/init.d
# ln -s ../init.d/afs /etc/rc3.d/S99afs
# ln -s ../init.d/afs /etc/rc0.d/K66afs
```

(d) **(Optional)** There are now copies of the AFS initialization file in both the **/usr/vice/etc** and **/etc/init.d** directories. If you want to avoid potential confusion by guaranteeing that they are always the same, create a link between them. You can always retrieve the original script from the OpenAFS Binary Distribution if necessary.

```
# cd /usr/vice/etc
# rm afs.rc
# ln -s  /etc/init.d/afs  afs.rc
```

4. Verify that **/usr/afs** and its subdirectories on the new file server machine meet the ownership and mode bit requirements outlined in Protecting Sensitive AFS Directories. If necessary, use the **chmod** command to correct the mode bits.

5. To configure this machine as a database server machine, proceed to Installing Database Server Functionality.

3.2 Installing Database Server Functionality

This section explains how to install database server functionality. Database server machines have two defining characteristics. First, they run the Protection Server, and Volume Location (VL) Server processes. They also run the Backup Server if the cell uses the AFS Backup System, as is assumed in these instructions. Second, they 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).

Note the following requirements for database server machines.

- In the conventional configuration, database server machines also serve as file server machines (run the File Server, Volume Server and Salvager processes). If you choose not to run file server functionality on a database server machine, then the kernel does not have to incorporate AFS modifications, but the local **/usr/afs** directory must house most of the standard files and subdirectories. In particular, the **/usr/afs/etc/KeyFile** file must contain the same keys as all other server machines in the cell. If you run a system control machine, run the **upclientetc** process on every database server machine other than the system control machine; if you do not run a system control machine, use the **bos addkey** command as instructed in the chapter in the *OpenAFS Administration Guide* about maintaining server encryption keys.

  The instructions in this section assume that the machine on which you are installing database server functionality is already a file server machine. Contact the OpenAFS mailing list 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 (synchronization site) for each database server process. This can cause a system outage, which usually lasts less than 5 minutes.

- Updating the kernel memory list of database server machines on each client machine is generally the most time-consuming part of installing a new database server machine. It is, however, 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 AFS commands.

  You update a client's kernel memory list by changing the **/usr/vice/etc/CellServDB** file and then either rebooting or issuing the **fs newcell** command. For instructions, see the chapter in the *OpenAFS Administration Guide* about administering client machines.

  The point at which you update your clients' knowledge of database server machines depends on which of the database server machines has the lowest IP address. The following instructions indicate the appropriate place to update your client machines in either case.

  – If the new database server machine has a lower IP address than any existing database server machine, update the **CellServDB** file on every client machine before restarting the database server processes. If you do not, users can become unable to update (write to) any of the AFS databases. This is because the machine with the lowest IP address is usually elected as the Ubik coordinator, and only the Coordinator accepts database writes. On client machines that do not have the new list of database server machines, the Cache Manager cannot locate the new coordinator. (Be aware that if clients contact the new coordinator before it is actually in service, they 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. Client machines do not start using the new database server machine until you update their kernel memory list, but that does not usually cause timeouts or update problems (because the new machine is not likely to become the coordinator).

3.2.1 Summary of Procedures

To install a database server machine, perform the following procedures.

1. Install the **bos** suite of commands locally, as a precaution
2. Add the new machine to the **/usr/afs/etc/CellServDB** file on existing file server machines
3. Update your cell's central **CellServDB** source file and the file you make available to foreign cells
4. Update every client machine's **/usr/vice/etc/CellServDB** file and kernel memory list of database server machines
5. Start the database server processes (Backup Server, Protection Server, and Volume Location Server)
6. Restart the database server processes on every database server machine
7. If required, request that grand.central.org add details of your new database server machine to the global CellServDB
8. If required, add details of your new database server to the AFS database location records in your site's DNS

3.2.2 Instructions

Note
It is assumed that your PATH environment variable includes the directory that houses the AFS command binaries. If not, you possibly need to precede the command names with the appropriate pathname.

1. You can perform the following instructions on either a server or client machine. Login as an AFS administrator who is listed in the **/usr/afs/etc/UserList** file on all server machines.

```
% kinit admin_user
Password: admin_password
% aklog
```

2. If you are working on a client machine configured in the conventional manner, the **bos** command suite resides in the **/usr/afsws/bin** directory, a symbolic link to an AFS directory. An error during installation can potentially block access to AFS, in which case it is helpful to have a copy of the **bos** binary on the local disk. This step is not necessary if you are working on a server machine, where the binary resides in the local **/usr/afs/bin** directory.

```
% cp  /usr/afsws/bin/bos   /tmp
```

3. Issue the **bos addhost** command to add the new database server machine to the **/usr/afs/etc/CellServDB** file on existing server machines (as well as the new database server machine itself).

   Substitute the new database server machine's fully-qualified hostname for the *host name* argument. If you run a system control machine, substitute its fully-qualified hostname for the *machine name* argument. If you do not run a system control machine, repeat the **bos addhost** command once for each server machine in your cell (including the new database server machine itself), by substituting each one's fully-qualified hostname for the *machine name* argument in turn.

```
% bos addhost <machine name>  <host name>
```

   If you run a system control machine, wait for the Update Server to distribute the new **CellServDB** file, which takes up to five minutes by default. If you are issuing individual **bos addhost** commands, attempt to issue all of them within five minutes.

   Note
   It is best to maintain a one-to-one mapping between hostnames and IP addresses on a multihomed database server machine (the conventional configuration for any AFS machine). The BOS Server uses the **gethostbyname( )** routine to obtain the IP address associated with the *host name* argument. If there is more than one address, the BOS Server records in the **CellServDB** entry the one that appears first in the list of addresses returned by the routine. The routine possibly returns addresses in a different order on different machines, which can create inconsistency.

4. **(Optional)** Issue the **bos listhosts** command on each server machine to verify that the new database server machine appears in its **CellServDB** file.

```
% bos listhosts <machine name>
```

5. Add the new database server machine to your cell's central **CellServDB** source file, if you use one. The standard location is **/afs/***cellname***/common/etc/CellServDB**.

   If you are willing to make your cell accessible to users in foreign cells, add the new database server machine to the file that lists your cell's database server machines. The conventional location is **/afs/***cellname***/service/etc/CellServDB.local**.

6. If this machine's IP address is lower than any existing database server machine's, update every client machine's **/usr/vice/etc/CellServDB** file and kernel memory list to include this machine. (If this machine's IP address is not the lowest, it is acceptable to wait until Step 12.)

   There are several ways to update the **CellServDB** file on client machines, as detailed in the chapter of the *OpenAFS Administration Guide* about administering client machines. One option is to copy over the central update source (which you updated in Step 5). To update the machine's kernel memory list, you can either reboot after changing the **CellServDB** file or issue the **fs newcell** command.

7. If you are running a cell which still relies upon **kaserver** see Starting the Authentication Service for an additional installation step.
8. Start the Backup Server (the **buserver** process). You must perform other configuration procedures before actually using the AFS Backup System, as detailed in the *OpenAFS Administration Guide*.

   ```
   % bos create <machine name> buserver simple /usr/afs/bin/buserver
   ```

9. Start the Protection Server (the **ptserver** process).

   ```
   % bos create <machine name> ptserver simple /usr/afs/bin/ptserver
   ```

10. Start the Volume Location (VL) Server (the **vlserver** process).

    ```
    % bos create <machine name> vlserver simple /usr/afs/bin/vlserver
    ```

11. Issue the **bos restart** command on every database server machine in the cell, including the new machine. The command restarts the Authentication, Backup, Protection, and VL Servers, which forces an election of a new Ubik coordinator for each process. 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 coordinator for the VL Server, but it normally lasts 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 appear on the console.

    Repeat this command on each of your cell's database server machines in quick succession. Begin with the machine with the lowest IP address.

    ```
    %  bos restart <machine name> kaserver buserver ptserver vlserver
    ```

    If an error occurs, restart all server processes on the database server machines again by using one of the following methods:

    - Issue the **bos restart** command with the **-bosserver** flag for each database server machine
    - Reboot each database server machine, either using the **bos exec** command or at its console

12. If you did not update the **CellServDB** file on client machines in Step 6, do so now.
13. If you wish to participate in the AFS global name space, send the new database server machine's name and IP address to grand.central.org. Do so, by emailing an updated **CellServDB** fragment for your cell to cellservdb@grand.central.org

    More details on the registration procedures for the CellServDB maintained by grand.central.org are available from http://grand.central.org/dynamic/servers.html

3.3 Removing Database Server Functionality

Removing database server machine functionality is nearly the reverse of installing it.

3.3.1 Summary of Procedures

To decommission a database server machine, perform the following procedures.

1. Install the **bos** suite of commands locally, as a precaution
2. If you participate in the global AFS namespace, notify grand.central.org that you are decommissioning a database server machine
3. Update your cell's central **CellServDB** source file and the file you make available to foreign cells
4. Update every client machine's **/usr/vice/etc/CellServDB** file and kernel memory list of database server machines
5. Remove the machine from the **/usr/afs/etc/CellServDB** file on file server machines
6. Stop the database server processes and remove them from the **/usr/afs/local/BosConfig** file if desired
7. Restart the database server processes on the remaining database server machines

3.3.2 Instructions

Note
It is assumed that your PATH environment variable includes the directory that houses the AFS command binaries. If not, you possibly need to precede the command names with the appropriate pathname.

1. You can perform the following instructions on either a server or client machine. Login as an AFS administrator who is listed in the **/usr/afs/etc/UserList** file on all server machines.

```
% kinit admin_user
Password: admin_password
% aklog
```

2. If you are working on a client machine configured in the conventional manner, the **bos** command suite resides in the **/usr/afsws/bin** directory, a symbolic link to an AFS directory. An error during installation can potentially block access to AFS, in which case it is helpful to have a copy of the **bos** binary on the local disk. This step is not necessary if you are working on a server machine, where the binary resides in the local **/usr/afs/bin** directory.

```
% cp  /usr/afsws/bin/bos   /tmp
```

3. If your cell is included in the global **CellServDB**, send the revised list of your cell's database server machines to grand.central.org

   If the administrators in foreign cells do not learn about the change in your cell, they cannot update the **CellServDB** file on their client machines. Users in foreign cells continue to send database requests to the decommissioned machine, which creates needless network traffic and activity on the machine. Also, the users experience time-out delays while their request is forwarded to a valid database server machine.

4. Remove the decommissioned machine from your cell's central **CellServDB** source file, if you use one. The conventional location is **/afs/***cellname***/common/etc/CellServDB**.

   If you maintain a file that users in foreign cells can access to learn about your cell's database server machines, update it also. The conventional location is **/afs/***cellname***/service/etc/CellServDB.local**.

5. Update every client machine's **/usr/vice/etc/CellServDB** file and kernel memory list to exclude this machine. Altering the **CellServDB** file and kernel memory list before stopping the actual database server processes avoids possible time-out delays that result when users send requests to a decommissioned database server machine that is still listed in the file.

   There are several ways to update the **CellServDB** file on client machines, as detailed in the chapter of the *OpenAFS Administration Guide* about administering client machines. One option is to copy over the central update source (which you updated in Step 5). To update the machine's kernel memory list, you can either reboot after changing the **CellServDB** file or issue the **fs newcell** command.

6. Issue the **bos removehost** command to remove the decommissioned database server machine from the **/usr/afs/etc/-CellServDB** file on server machines.

   Substitute the decommissioned database server machine's fully-qualified hostname for the *host name* argument. If you run a system control machine, substitute its fully-qualified hostname for the *machine name* argument. If you do not run a system control machine, repeat the **bos removehost** command once for each server machine in your cell (including the decommissioned database server machine itself), by substituting each one's fully-qualified hostname for the *machine name* argument in turn.

```
% bos removehost <machine name>  <host name>
```

   If you run a system control machine, wait for the Update Server to distribute the new **CellServDB** file, which takes up to five minutes by default. If issuing individual **bos removehost** commands, attempt to issue all of them within five minutes.

7. **(Optional)** Issue the **bos listhosts** command on each server machine to verify that the decommissioned database server machine no longer appears in its **CellServDB** file.

```
% bos listhosts <machine name>
```

8. Issue the **bos stop** command to stop the database server processes on the machine, by substituting its fully-qualified hostname for the *machine name* argument. The command changes each process's status in the **/usr/afs/local/BosConfig** file to NotRun, but does not remove its entry from the file.

```
% bos stop <machine name> kaserver buserver ptserver vlserver
```

9. **(Optional)** Issue the **bos delete** command to remove the entries for database server processes from the **BosConfig** file. This step is unnecessary if you plan to restart the database server functionality on this machine in future.

```
% bos delete <machine name> buserver ptserver vlserver
```

10. Issue the **bos restart** command on every database server machine in the cell, to restart the Backup, Protection, and VL Servers. 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.

    A cell-wide service outage is possible during the election of a new coordinator for the VL Server, but it normally lasts less than five minutes. Messages tracing the progress of the election appear on the console.

    Repeat this command on each of your cell's database server machines in quick succession. Begin with the machine with the lowest IP address.

```
%  bos restart <machine name> buserver ptserver vlserver
```

    If an error occurs, restart all server processes on the database server machines again by using one of the following methods:

    - Issue the **bos restart** command with the **-bosserver** flag for each database server machine
    - Reboot each database server machine, either using the **bos exec** command or at its console

Chapter 4

Installing Additional Client Machines

This chapter describes how to install AFS client machines after you have installed the first AFS machine. Some parts of the installation differ depending on whether or not the new client is of the same AFS system type (uses the same AFS binaries) as a previously installed client machine.

4.1 Summary of Procedures

1. Incorporate AFS into the machine's kernel
2. Define the machine's cell membership
3. Define cache location and size
4. Create the **/usr/vice/etc/CellServDB** file, which determines which foreign cells the client can access in addition to the local cell
5. Create the **/afs** directory and start the Cache Manager
6. Create and mount volumes for housing AFS client binaries (necessary only for clients of a new system type)
7. Create a link from the local **/usr/afsws** directory to the AFS directory housing the AFS client binaries
8. Modify the machine's authentication system to enable AFS users to obtain tokens at login

4.2 Creating AFS Directories on the Local Disk

If you are not installing from a packaged distribution, create the **/usr/vice/etc** directory on the local disk, to house client binaries and configuration files. Subsequent instructions copy files from the OpenAFS binary distribution into them. Create the **/tmp/afsdist** directory as a location to uncompress this distribution, if it does not already exist.

```
# mkdir /usr/vice
# mkdir /usr/vice/etc
# mkdir /tmp/afsdist
```

4.3 Performing Platform-Specific Procedures

Every AFS client machine's kernel must incorporate AFS modifications. Some system types use a dynamic kernel loader program, whereas on other system types you build AFS modifications into a static kernel. Some system types support both methods.

Also modify the machine's authentication system so that users obtain an AFS token as they log into the local file system. Using AFS is simpler and more convenient for your users if you make the modifications on all client machines. Otherwise, users must perform a two or three step login procedure (login to the local system, obtain Kerberos credentials, and then issue the **klog** command). For further discussion of AFS authentication, see the chapter in the *OpenAFS Administration Guide* about cell configuration and administration issues.

For convenience, the following sections group the two procedures by system type. Proceed to the appropriate section.

- Getting Started on Linux Systems
- Getting Started on Solaris Systems

4.4 Getting Started on Linux Systems

In this section you load AFS into the Linux kernel. Then incorporate AFS modifications into the machine's Pluggable Authentication Module (PAM) system, if you wish to enable AFS login.

4.4.1 Loading AFS into the Linux Kernel

The **modprobe** program is the dynamic kernel loader for Linux. Linux does not support incorporation of AFS modifications during a kernel build.

For AFS to function correctly, the **modprobe** program must run each time the machine reboots, so your distributions's AFS initialization script invokes it automatically. The script also includes commands that select the appropriate AFS library file automatically. In this section you run the script.

In a later section you also verify that the script correctly initializes the Cache Manager, then activate a configuration variable, which results in the script being incorporated into the Linux startup and shutdown sequence.

The procedure for starting up OpenAFS depends upon your distribution

4.4.1.1 Fedora and RedHat Enterprise Linux

OpenAFS ships RPMS for all current Fedora and RHEL releases.

1. Download and install the RPM set for your operating system. RPMs are available from the OpenAFS web site. You will need the **openafs**, **openafs-server**, **openafs-client** and **openafs-krb5** packages, along with an **kmod-openafs** package matching your current, running ,kernel.

   You can find the version of your current kernel by running

   ```
     # uname -r
   2.6.20-1.2933.fc6
   ```

   Once downloaded, the packages may be installed with the **rpm** command

   ```
     # rpm -U openafs-* openafs-client-* openafs-server-* openafs-krb5-* kmod-openafs-*
   ```

4.4.1.2 Systems packaged as tar files

If you are running a system where the OpenAFS Binary Distribution is provided as a tar file, or where you have built the system from source yourself, you need to install the relevant components by hand

1. Unpack the distribution tarball. The examples below assume that you have unpacked the files into the **/tmp/afsdist**directory. If you pick a different location, substitute this in all of the following examples. Once you have unpacked the distribution, change directory as indicated.

```
# cd /tmp/afsdist/linux/dest/root.client/usr/vice/etc
```

2. Copy the AFS kernel library files to the local **/usr/vice/etc/modload** directory. The filenames for the libraries have the format **libafs-***version***.o**, where *version* indicates the kernel build level. The string **.mp** in the *version* indicates that the file is appropriate for machines running a multiprocessor kernel.

```
# cp -rp  modload  /usr/vice/etc
```

3. Copy the AFS initialization script to the local directory for initialization files (by convention, **/etc/rc.d/init.d** on Linux machines). Note the removal of the **.rc** extension as you copy the script.

```
# cp -p   afs.rc  /etc/rc.d/init.d/afs
```

4.4.2 Enabling AFS Login on Linux Systems

At this point you incorporate AFS into the operating system's Pluggable Authentication Module (PAM) scheme. PAM integrates all authentication mechanisms on the machine, including login, to provide the security infrastructure for authenticated access to and from the machine.

At this time, we recommend that new sites requiring AFS credentials to be gained as part of PAM authentication use Russ Alberry's pam_afs_session, rather than utilising the bundled pam_afs2 module. A typical PAM stack should authenticate the user using an external Kerberos V service, and then use the AFS PAM module to obtain AFS credentials in the `session` section

If you are at a site which still requires **kaserver** or external Kerberos v4 based authentication, please consult Enabling kaserver based AFS Login on Linux Systems for further installation instructions.

Proceed to Loading and Creating Client Files.

4.5 Getting Started on Solaris Systems

In this section you load AFS into the Solaris kernel. Then incorporate AFS modifications into the machine's Pluggable Authentication Module (PAM) system, if you wish to enable AFS login.

4.5.1 Loading AFS into the Solaris Kernel

The **modload** program is the dynamic kernel loader provided by Sun Microsystems for Solaris systems. Solaris does not support incorporation of AFS modifications during a kernel build.

For AFS to function correctly, the **modload** program must run each time the machine reboots, so the AFS initialization script (included on the AFS CD-ROM) invokes it automatically. In this section you copy the appropriate AFS library file to the location where the **modload** program accesses it and then run the script.

In a later section you verify that the script correctly initializes the Cache Manager, then create the links that incorporate AFS into the Solaris startup and shutdown sequence.

1. Unpack the OpenAFS Solaris distribution tarball. The examples below assume that you have unpacked the files into the **/tmp/afsdist** directory. If you pick a diferent location, substitute this in all of the following exmaples. Once you have unpacked the distribution, change directory as indicated.

```
# cd  /tmp/afsdist/sun4x_56/dest/root.client/usr/vice/etc
```

2. Copy the AFS initialization script to the local directory for initialization files (by convention, **/etc/init.d** on Solaris machines). Note the removal of the **.rc** extension as you copy the script.

```
# cp -p  afs.rc  /etc/init.d/afs
```

3. Copy the appropriate AFS kernel library file to the local file **/kernel/fs/afs**.

   If the machine is running Solaris 11 on the x86_64 platform:

```
# cp -p modload/libafs64.o /kernel/drv/amd64/afs
```

   If the machine is running Solaris 10 on the x86_64 platform:

```
# cp -p modload/libafs64.o /kernel/fs/amd64/afs
```

   If the machine is running Solaris 2.6 or the 32-bit version of Solaris 7, its kernel supports NFS server functionality, and the **nfsd** process is running:

```
# cp -p modload/libafs.o /kernel/fs/afs
```

   If the machine is running Solaris 2.6 or the 32-bit version of Solaris 7, and its kernel does not support NFS server functionality or the **nfsd** process is not running:

```
# cp -p modload/libafs.nonfs.o /kernel/fs/afs
```

   If the machine is running the 64-bit version of Solaris 7, its kernel supports NFS server functionality, and the **nfsd** process is running:

```
# cp -p modload/libafs64.o /kernel/fs/sparcv9/afs
```

   If the machine is running the 64-bit version of Solaris 7, and its kernel does not support NFS server functionality or the **nfsd** process is not running:

```
# cp -p modload/libafs64.nonfs.o /kernel/fs/sparcv9/afs
```

4. Run the AFS initialization script to load AFS modifications into the kernel. You can ignore any error messages about the inability to start the BOS Server or the Cache Manager or AFS client.

```
# /etc/init.d/afs start
```

   When an entry called `afs` does not already exist in the local **/etc/name_to_sysnum** file, the script automatically creates it and reboots the machine to start using the new version of the file. If this happens, log in again as the superuser **root** after the reboot and run the initialization script again. This time the required entry exists in the **/etc/name_to_sysnum** file, and the **modload** program runs.

```
login: root
Password: root_password
# /etc/init.d/afs start
```

4.5.2 Enabling AFS Login on Solaris Systems

At this point you incorporate AFS into the operating system's Pluggable Authentication Module (PAM) scheme. PAM integrates all authentication mechanisms on the machine, including login, to provide the security infrastructure for authenticated access to and from the machine.

In modern AFS installations, you should be using Kerberos v5 for user login, and obtaining AFS tokens subsequent to this authentication step. OpenAFS does not currently distribute a PAM module allowing AFS tokens to be automatically gained at login. Some of these, such as pam-krb5 and pam-afs-session from http://www.eyrie.org/~eagle/software/ or pam_afs2 from ftp://achilles.ctd.anl.gov/pub/DEE/pam_afs2-0.1.tar, have been tested with Solaris.

If you are at a site which still requires **kaserver** or external Kerberos v4 based authentication, please consult Enabling kaserver based AFS Login on Solaris Systems for further installation instructions.

4.5.3 Editing the File Systems Clean-up Script on Solaris Systems

1. Some Solaris distributions include a script that locates and removes unneeded files from various file systems. Its conventional location is **/usr/lib/fs/nfs/nfsfind**. The script generally uses an argument to the **find** command to define which file systems to search. In this step you modify the command to exclude the **/afs** directory. Otherwise, the command traverses the AFS filespace of every cell that is accessible from the machine, which can take many hours. The following alterations are possibilities, but you must verify that they are appropriate for your cell.

   The first possible alteration is to add the **-local** flag to the existing command, so that it looks like the following:

   ```
   find $dir -local -name .nfs\* -mtime +7 -mount -exec rm -f {} \;
   ```

   Another alternative is to exclude any directories whose names begin with the lowercase letter **a** or a non-alphabetic character.

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

   Do not use the following command, which still searches under the **/afs** directory, looking for a subdirectory of type **4.2**.

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

2. Proceed to Loading and Creating Client Files.

4.6 Loading and Creating Client Files

If you are using a non-packaged distribution (that is, one provided as a tarball) you should now copy files from the istribution to the **/usr/vice/etc** directory. On some platforms that use a dynamic loader program to incorporate AFS modifications into the kernel, you have already copied over some the files. Copying them again does no harm.

Every AFS client machine has a copy of the **/usr/vice/etc/ThisCell** file on its local disk to define the machine's cell membership for the AFS client programs that run on it. Among other functions, this file determines the following:

- The cell in which users authenticate when they log onto the machine, assuming it is using an AFS-modified login utility
- The cell in which users authenticate by default when they issue the **aklog** command
- The cell membership of the AFS server processes that the AFS command interpreters on this machine contact by default

Similarly, the **/usr/vice/etc/CellServDB** file on a client machine's local disk lists the database server machines in each cell that the local Cache Manager can contact. If there is no entry in the file for a cell, or the list of database server machines is wrong, then users working on this machine cannot access the cell. The chapter in the *OpenAFS Administration Guide* about administering client machines explains how to maintain the file after creating it. A version of the client **CellServDB** file was created during the installation of your cell's first machine (in Creating the Client CellServDB File). It is probably also appropriate for use on this machine.

Remember that the Cache Manager consults the **/usr/vice/etc/CellServDB** file 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 chapter in the *OpenAFS Administration Guide* about administering client machines for instructions on updating this file, with or without rebooting.

1. If you have not already done so, unpack the distribution tarball for this machine's system type into a suitable location on the filesystem, such as **/tmp/afsdist**. If you use a different location, substitue that in the examples that follow.

2. Copy files to the local **/usr/vice/etc** directory.

   This step places a copy of the AFS initialization script (and related files, if applicable) into the **/usr/vice/etc** directory. In the preceding instructions for incorporating AFS into the kernel, you copied the script directly to the operating system's conventional location for initialization files. When you incorporate AFS into the machine's startup sequence in a later step, you can choose to link the two files.

On some system types that use a dynamic kernel loader program, you previously copied AFS library files into a subdirectory of the **/usr/vice/etc** directory. On other system types, you copied the appropriate AFS library file directly to the directory where the operating system accesses it. The following commands do not copy or recopy the AFS library files into the **/usr/vice/etc** directory, because on some system types the library files consume a large amount of space. If you want to copy them, add the **-r** flag to the first **cp** command and skip the second **cp** command.

```
# cd /cdrom/sysname/root.client/usr/vice/etc
# cp -p  *  /usr/vice/etc
# cp -rp  C  /usr/vice/etc
```

3. Create the **/usr/vice/etc/ThisCell** file.

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

4. Create the **/usr/vice/etc/CellServDB** file. Use a network file transfer program such as **sftp** or **scp** to copy it from one of the following sources, which are listed in decreasing order of preference:
   - Your cell's central **CellServDB** source file (the conventional location is **/afs/***cellname***/common/etc/CellServDB**)
   - The global **CellServDB** file maintained at grand.central.org
   - An existing client machine in your cell
   - The **CellServDB.sample** file included in the *sysname***/root.client/usr/vice/etc** directory of each OpenAFS distribution; add an entry for the local cell by following the instructions in Creating the Client CellServDB File

4.7 Configuring the Cache

The Cache Manager uses a cache on the local disk or in machine memory to store local copies of files fetched from file server machines. As the **afsd** program initializes the Cache Manager, it sets basic cache configuration parameters according to definitions in the local **/usr/vice/etc/cacheinfo** file. The file has three fields:

1. The first field names the local directory on which to mount the AFS filespace. The conventional location is the **/afs** directory.
2. The second field defines the local disk directory to use for the disk cache. The conventional location is the **/usr/vice/cache** directory, but you can specify an alternate directory if another partition has more space available. There must always be a value in this field, but the Cache Manager ignores it if the machine uses a memory cache.
3. The third field specifies the number of kilobyte (1024 byte) blocks to allocate for the cache.

The values you define must meet the following requirements.

- On a machine using a disk cache, the Cache Manager expects always to be able to use the amount of space specified in the third field. Failure to meet this requirement can cause serious problems, some of which can be repaired only by rebooting. You must prevent non-AFS processes from filling up the cache partition. The simplest way is to devote a partition to the cache exclusively.
- The amount of space available in memory or on the partition housing the disk cache directory imposes an absolute limit on cache size.
- The maximum supported cache size can vary in each AFS release; see the *OpenAFS Release Notes* for the current version.
- For a disk cache, you cannot specify a value in the third field that exceeds 95% of the space available on the partition mounted at the directory named in the second field. If you violate this restriction, the **afsd** program exits without starting the Cache Manager and prints an appropriate message on the standard output stream. A value of 90% is more appropriate on most machines. Some operating systems do not automatically reserve some space to prevent the partition from filling completely; for them, a smaller value (say, 80% to 85% of the space available) is more appropriate.

- For a memory cache, you must leave enough memory for other processes and applications to run. If you try to allocate more memory than is actually available, the **afsd** program exits without initializing the Cache Manager and produces the following message on the standard output stream.

```
afsd: memCache allocation failure at number KB
```

  The *number* value is how many kilobytes were allocated just before the failure, and so indicates the approximate amount of memory available.

Within these hard limits, the factors that determine appropriate cache size include the number of users working on the machine, the size of the files with which they work, and (for a memory cache) the number of processes that run on the machine. The higher the demand from these factors, the larger the cache needs to be to maintain good performance.

Disk caches smaller than 10 MB do not generally perform well. Machines serving multiple users usually perform better with a cache of at least 60 to 70 MB. The point at which enlarging the cache further does not really improve performance depends on the factors mentioned previously and is difficult to predict.

Memory caches smaller than 1 MB are nonfunctional, and the performance of caches smaller than 5 MB is usually unsatisfactory. Suitable upper limits are similar to those for disk caches but are probably determined more by the demands on memory from other sources on the machine (number of users and processes). Machines running only a few processes possibly can use a smaller memory cache.

4.7.1 Configuring a Disk Cache

Note
Not all file system types that an operating system supports are necessarily supported for use as the cache partition. For possible restrictions, see the *OpenAFS Release Notes*.

To configure the disk cache, perform the following procedures:

1. Create the local directory to use for caching. The following instruction shows the conventional location, **/usr/vice/cache**. If you are devoting a partition exclusively to caching, as recommended, you must also configure it, make a file system on it, and mount it at the directory created in this step.

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

2. Create the **cacheinfo** file to define the configuration parameters discussed previously. The following 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
```

   The following example defines the disk cache size as 50,000 KB:

```
# echo "/afs:/usr/vice/cache:50000" > /usr/vice/etc/cacheinfo
```

4.7.2 Configuring a Memory Cache

To configure a memory cache, create the **cacheinfo** file to define the configuration parameters discussed previously. The following instruction shows the standard mount location, **/afs**, and the standard cache location, **/usr/vice/cache** (though the exact value of the latter is irrelevant for a memory cache).

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

The following example allocates 25,000 KB of memory for the cache.

```
# echo "/afs:/usr/vice/cache:25000" > /usr/vice/etc/cacheinfo
```

4.8 Configuring the Cache Manager

By convention, the Cache Manager mounts the AFS filespace on the local **/afs** directory. In this section you create that directory.

The **afsd** program sets several cache configuration parameters as it initializes the Cache Manager, and starts daemons that improve performance. You can use the **afsd** command's arguments to override the parameters' default values and to change the number of some of the daemons. Depending on the machine's cache size, its amount of RAM, and how many people work on it, you can sometimes improve Cache Manager performance by overriding the default values. For a discussion of all of the **afsd** command's arguments, see its reference page in the *OpenAFS Administration Reference*.

On platforms using the standard 'afs' initialisation script (this does not apply to Fedora or RHEL based distributions), the **afsd** command line in the AFS initialization script on each system type includes an `OPTIONS` variable. You can use it to set nondefault values for the command's arguments, in one of the following ways:

- You can create an **afsd** *options file* that sets values for arguments to the **afsd** command. If the file exists, its contents are automatically substituted for the `OPTIONS` variable in the AFS initialization script. The AFS distribution for some system types includes an options file; on other system types, you must create it.

  You use two variables in the AFS initialization script to specify the path to the options file: `CONFIG` and `AFSDOPT`. On system types that define a conventional directory for configuration files, the `CONFIG` variable indicates it by default; otherwise, the variable indicates an appropriate location.

  List the desired **afsd** options on a single line in the options file, separating each option with one or more spaces. The following example sets the **-stat** argument to 2500, the **-daemons** argument to 4, and the **-volumes** argument to 100.

  ```
  -stat 2500 -daemons 4 -volumes 100
  ```

- On a machine that uses a disk cache, you can set the `OPTIONS` variable in the AFS initialization script to one of `$SMALL`, `$MEDIUM`, or `$LARGE`. The AFS initialization script uses one of these settings if the **afsd** options file named by the `AFSDOPT` variable does not exist. In the script as distributed, the `OPTIONS` variable is set to the value `$MEDIUM`.

  Note
  Do not set the `OPTIONS` variable to `$SMALL`, `$MEDIUM`, or `$LARGE` on a machine that uses a memory cache. The arguments it sets are appropriate only on a machine that uses a disk cache.

  The script (or on some system types the **afsd** options file named by the `AFSDOPT` variable) defines a value for each of `SMALL`, `MEDIUM`, and `LARGE` that sets **afsd** command arguments appropriately for client machines of different sizes:

  - `SMALL` is suitable for a small machine that serves one or two users and has approximately 8 MB of RAM and a 20-MB cache
  - `MEDIUM` is suitable for a medium-sized machine that serves two to six users and has 16 MB of RAM and a 40-MB cache
  - `LARGE` is suitable for a large machine that serves five to ten users and has 32 MB of RAM and a 100-MB cache

- You can choose not to create an **afsd** options file and to set the `OPTIONS` variable in the initialization script to a null value rather than to the default `$MEDIUM` value. You can then either set arguments directly on the **afsd** command line in the script, or set no arguments (and so accept default values for all Cache Manager parameters).

Note
If you are running on a Fedora or RHEL based system, the openafs-client initialization script behaves differently from that described above. It sources **/etc/sysconfig/openafs**, in which the AFSD_ARGS variable may be set to contain any, or all, of the afsd options detailed above. Note that this script does not support setting an `OPTIONS` variable, or the `SMALL`, `MEDIUM` and `LARGE` methods of defining cache size.

1. Create the local directory on which to mount the AFS filespace, by convention **/afs**. If the directory already exists, verify that it is empty.

```
# mkdir /afs
```

2. On non-package based Linux systems, copy the **afsd** options file from the **/usr/vice/etc** directory to the **/etc/sysconfig** directory, removing the **.conf** extension as you do so.

```
# cp /usr/vice/etc/afs.conf /etc/sysconfig/afs
```

3. Edit the machine's AFS initialization script or **afsd** options file to set appropriate values for **afsd** command parameters. The appropriate file for each system type is as follows:

   - On Fedora and RHEL systems, **/etc/sysconfig/openafs**
   - On Linux systems, **/etc/sysconfig/afs** (the **afsd** options file)
   - On Solaris systems, **/etc/init.d/afs**

   Use one of the methods described in the introduction to this section to add the following flags to the **afsd** command line. Also set any performance-related arguments you wish.

   - Add the **-memcache** flag if the machine is to use a memory cache.
   - Add the **-verbose** flag to display a trace of the Cache Manager's initialization on the standard output stream.

4.9 Starting the Cache Manager and Installing the AFS Initialization Script

In this section you run the AFS initialization script to start the Cache Manager. If the script works correctly, perform the steps that incorporate it into the machine's startup and shutdown sequence. If there are problems during the initialization, attempt to resolve them. The AFS Product Support group can provide assistance if necessary.

On machines that use a disk cache, it can take a while for the **afsd** program to run the first time on a machine, because it must create all of the **V***n* files in the cache directory. Subsequent Cache Manager initializations do not take nearly as long, because the **V***n* files already exist.

On system types that use a dynamic loader program, you must reboot the machine before running the initialization script, so that it can freshly load AFS modifications into the kernel.

Proceed to the instructions for your system type:

- Running the Script on Linux Systems
- Running the Script on Solaris Systems

4.9.1 Running the Script on Fedora / RHEL Systems

1. Reboot the machine and log in again as the local superuser **root**.

```
# cd /
# shutdown -r now
login: root
Password: root_password
```

2. Run the AFS initialization script.

```
# /etc/rc.d/init.d/openafs-client  start
```

3. Issue the **chkconfig** command to activate the **openafs-client** configuration variable. Based on the instruction in the AFS initialization file that begins with the string `#chkconfig`, the command automatically creates the symbolic links that incorporate the script into the Linux startup and shutdown sequence.

```
# /sbin/chkconfig  --add openafs-client
```

4.9.2 Running the Script on other Linux Systems

1. Reboot the machine and log in again as the local superuser **root**.

```
# cd /
# shutdown -r now
login: root
Password: root_password
```

2. Run the AFS initialization script.

```
# /etc/rc.d/init.d/afs  start
```

3. Issue the **chkconfig** command to activate the **afs** configuration variable. Based on the instruction in the AFS initialization file that begins with the string `#chkconfig`, the command automatically creates the symbolic links that incorporate the script into the Linux startup and shutdown sequence.

```
# /sbin/chkconfig  --add afs
```

4. **(Optional)** There are now copies of the AFS initialization file in both the **/usr/vice/etc** and **/etc/rc.d/init.d** directories, and copies of the **afsd** options file in both the **/usr/vice/etc** and **/etc/sysconfig** directories. If you want to avoid potential confusion by guaranteeing that the two copies of each file are always the same, create a link between them. You can always retrieve the original script or options file from the AFS CD-ROM if necessary.

```
# cd /usr/vice/etc
# rm afs.rc afs.conf
# ln -s  /etc/rc.d/init.d/afs  afs.rc
# ln -s  /etc/sysconfig/afs  afs.conf
```

5. If a volume for housing AFS binaries for this machine's system type does not already exist, proceed to Setting Up Volumes and Loading Binaries into AFS. Otherwise, the installation is complete.

4.9.3 Running the Script on Solaris Systems

1. Reboot the machine and log in again as the local superuser **root**.

```
# cd /
# shutdown -i6 -g0 -y
login: root
Password: root_password
```

2. Run the AFS initialization script.

```
# /etc/init.d/afs  start
```

3. Change to the **/etc/init.d** directory and issue the **ln -s** command to create symbolic links that incorporate the AFS initialization script into the Solaris startup and shutdown sequence.

```
# cd /etc/init.d
# ln -s ../init.d/afs /etc/rc3.d/S99afs
# ln -s ../init.d/afs /etc/rc0.d/K66afs
```

4. **(Optional)** There are now copies of the AFS initialization file in both the **/usr/vice/etc** and **/etc/init.d** directories. If you want to avoid potential confusion by guaranteeing that they are always the same, create a link between them. You can always retrieve the original script from the OpenAFS Binary Distribution if necessary.

```
# cd /usr/vice/etc
# rm afs.rc
# ln -s  /etc/init.d/afs  afs.rc
```

5. If a volume for housing AFS binaries for this machine's system type does not already exist, proceed to Setting Up Volumes and Loading Binaries into AFS. Otherwise, the installation is complete.

4.10 Setting Up Volumes and Loading Binaries into AFS

Note

If you are using an operating system which uses packaged binaries, such as .rpms or .debs, you should allow these package management systems to maintain your AFS binaries, rather than following the instructions in this section.

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 conventional name for the AFS directory is **/afs/***cellname***/***sysname***/usr/afsws**.

If this machine is an existing system type, the AFS directory presumably already exists. You can simply create a link from the local **/usr/afsws** directory to it. Follow the instructions in Linking /usr/afsws on an Existing System Type.

If this machine is a new system type (there are no AFS 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. See Creating Binary Volumes for a New System Type.

You can also store UNIX system binaries (the files normally stored in local disk directories such as **/bin**, **/etc**, and **/lib**) in volumes mounted under **/afs/***cellname***/***sysname*. See Storing System Binaries in AFS .

4.10.1 Linking /usr/afsws on an Existing System Type

If this client machine is an existing system type, there is already a volume mounted in the AFS filespace that houses AFS client binaries for it.

1. Create **/usr/afsws** on the local disk as a symbolic link to the directory **/afs/***cellname***/@sys/usr/afsws**. You can specify the actual system name instead of **@sys** if you wish, but the advantage of using **@sys** is that it remains valid if you upgrade this machine to a different system type.

   ```
   # ln -s /afs/cellname/@sys/usr/afsws  /usr/afsws
   ```

2. **(Optional)** If you believe it is helpful to your users to access the AFS documents in a certain format via a local disk directory, create **/usr/afsdoc** on the local disk as a symbolic link to the documentation directory in AFS (**/afs/***cellname***/afsdoc/***format*

   ```
   # ln -s /afs/cellname/afsdoc/format_name /usr/afsdoc
   ```

   An alternative is to create a link in each user's home directory to the **/afs/***cellname***/afsdoc/***format_name* directory.

4.10.2 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** directory to an AFS directory.

To create and mount the volumes, you use the **kinit** command to authenticate as an administrator, followed by the **aklog** command to gain tokens, and then issue commands from the **vos** and **fs** command suites. However, the command binaries are not yet available on this machine (by convention, they are accessible via the **/usr/afsws** link that you are about to create). You have two choices:

- Perform all steps except the last one (Step 10) on an existing AFS machine. On a file server machine, the **aklog**, **fs** and **vos** binaries reside in the **/usr/afs/bin** directory. On client machines, the **aklog** and **fs** binaries reside in the **/usr/afsws/bin** directory and the **vos** binary in the **/usr/afsws/etc** directory. Depending on how your PATH environment variable is set, you possibly need to precede the command names with a pathname.

  If you work on another AFS machine, be sure to substitute the new system type name for the *sysname* argument in the following commands, not the system type of the machine on which you are issuing the commands.

- Copy the necessary command binaries to a temporary location on the local disk, which enables you to perform the steps on the local machine. The following procedure installs them in the **/tmp** directory and removes them at the end. Depending on how your PATH environment variable is set, you possibly need to precede the command names with a pathname.

Perform the following steps to create a volume for housing AFS binaries.

1. Working either on the local machine or another AFS machine, extract the Open AFS distribtion tarball onto a directory on that machine. The following instructions assume that you are using the **/tmp/afsdist** directory.

2. If working on the local machine, copy the necessary binaries to a temporary location on the local disk. Substitute a different directory name for **/tmp** if you wish.

```
# cd  /tmp/afsdist/new_sysname/root.server/usr/afs/bin
# cp -p  aklog  /tmp
# cp -p  fs  /tmp
# cp -p  vos  /tmp
```

3. Authenticate as the user **admin**.

```
# kinit admin
Password: admin_password
# aklog
```

4. Issue the **vos create** command to create volumes for storing the AFS client binaries for this system type. The following example instruction creates volumes called *sysname*, *sysname*.**usr**, and *sysname*.**usr.afsws**. Refer to the *OpenAFS Release Notes* to learn the proper value of *sysname* for this system type.

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

5. Issue the **fs mkmount** command to mount the newly created volumes. Because the **root.cell** volume is replicated, you must precede the *cellname* part of the pathname with a period to specify the read/write mount point, as shown. Then issue the **vos release** command to release a new replica of the **root.cell** volume, and the **fs checkvolumes** command to force the local Cache Manager to access them.

```
# fs mkmount -dir /afs/.cellname/sysname -vol sysname
# fs mkmount -dir /afs/.cellname/sysname/usr  -vol sysname.usr
# fs mkmount -dir /afs/.cellname/sysname/usr/afsws -vol sysname.usr.afsws
# vos release root.cell
# fs checkvolumes
```

6. Issue the **fs setacl** command to grant the **l** (**lookup**) and **r** (**read**) permissions to the **system:anyuser** group on each new directory's ACL.

```
# cd /afs/.cellname/sysname
# fs setacl  -dir  .  usr  usr/afsws  -acl  system:anyuser rl
```

7. Issue the **fs setquota** command to set an unlimited quota on the volume mounted at the **/afs/***cellname*/*sysname*/**usr/afsws** directory. This enables you to copy all of the appropriate files from the CD-ROM into the volume without exceeding the volume's quota.

   If you wish, you can set the volume's quota to a finite value after you complete the copying operation. At that point, use the **vos examine** command to determine how much space the volume is occupying. Then issue the **fs setquota** command to set a quota that is slightly larger.

```
# fs setquota /afs/.cellname/sysname/usr/afsws  0
```

8. Copy the contents of the indicated directories from the OpenAFS binary distribution into the **/afs/***cellname***/***sysname***/usr/afsws** directory.

```
# cd /afs/.cellname/sysname/usr/afsws
# cp -rp /cdrom/sysname/bin  .
# cp -rp /cdrom/sysname/etc  .
# cp -rp /cdrom/sysname/include  .
# cp -rp /cdrom/sysname/lib  .
```

9. Issue the **fs setacl** command to set the ACL on each directory appropriately. If you wish to enable access to the software for locally authenticated users only, set the ACL on the **etc**, **include**, and **lib** subdirectories to grant the **l** and **r** permissions to the **system:authuser** group rather than the **system:anyuser** group. The **system:anyuser** group must retain the **l** and **r** permissions on the **bin** subdirectory to enable unauthenticated users to access the **aklog** binary.

```
# cd /afs/.cellname/sysname/usr/afsws
# fs setacl  -dir etc include lib  -acl  system:authuser rl  \
           system:anyuser none
```

10. Perform this step on the new client machine even if you have performed the previous steps on another machine. Create **/usr/afsws** on the local disk as a symbolic link to the directory **/afs/***cellname***/@sys/usr/afsws**. You can specify the actual system name instead of **@sys** if you wish, but the advantage of using **@sys** is that it remains valid if you upgrade this machine to a different system type.

```
# ln -s /afs/cellname/@sys/usr/afsws  /usr/afsws
```

11. **(Optional)** To enable users to issue commands from the AFS suites (such as **fs**) without having to specify a pathname to their binaries, include the **/usr/afsws/bin** and **/usr/afsws/etc** directories in the PATH environment variable you define in each user's shell initialization file (such as **.cshrc**).

12. **(Optional)** If you believe it is helpful to your users to access the AFS documents in a certain format via a local disk directory, create **/usr/afsdoc** on the local disk as a symbolic link to the documentation directory in AFS (**/afs/***cellname***/afsdoc/***format*

```
# ln -s /afs/cellname/afsdoc/format_name /usr/afsdoc
```

An alternative is to create a link in each user's home directory to the **/afs/***cellname***/afsdoc/***format_name* directory.

13. **(Optional)** If working on the local machine, remove the AFS binaries from the temporary location. They are now accessible in the **/usr/afsws** directory.

```
# cd  /tmp
# rm  klog  fs  vos
```

Appendix A

Appendix A. Building OpenAFS from Source Code

This chapter describes how to build OpenAFS from source code.

A.1 Loading the Source Files

Working on an AFS client machine, login to AFS as a administrative user, then perform these steps to load the OpenAFS source tree from the OpenAFS Source Distribution.

1. Create and mount a volume for housing the OpenAFS source tree. These instructions name the volume **src.afs** and mount it at the **/afs/***cellname***/afs/src** directory.

   Setting the **-maxquota** argument to **0** (zero) sets an unlimited quota on the volume, which enables you to copy all of the files into the volume without exceeding its quota. If you wish, you can set the volume's quota to a finite value after you complete the copying operation. At that point, use the **vos examine** command to determine how much space the volume is occupying. Then issue the **fs setquota** command to set a quota that is slightly larger.

   ```
   # vos create <machine name> <partition name> src.afs -maxquota 0
   # cd /afs/.cellname
   # mkdir afs
   # fs mkmount afs/src src.afs
   # vos release root.cell
   # fs checkvolumes
   ```

2. Download the latest stable OpenAFS source distribution (openafs-src.*X.Y.Z*.tar.gz) from openafs.org to the local **/tmp** directory.

3. In the local **/tmp** directory, unpack the source archive.

   ```
   # cd /tmp
   # gzip -dc openafs-src-X.Y.Z.tar.gz | tar xvf -
   ```

4. Copy the source files from the unpacked archive into the newly created volume.

   ```
   # cd /tmp/openafs-X.Y.Z
   # cp -rp  *  /afs/.cellname/afs/src
   ```

A.2 Compiling OpenAFS Binaries Using Configure and Make

The OpenAFS distribution uses the **autoconf** program and Makefiles for compiling the OpenAFS software.

1. Create a subdirectory under the **/afs/.***cellname***/afs** directory for each system type for which you will build AFS binaries. Creating and mounting a volume for each system type is recommended, but you can also simply use the **mkdir** command. If you create a new volume, grant it an unlimited quota to avoid running out of space during the build process.

   ```
   # cd /afs/.cellname/afs
   ```

   If creating a new volume:

   ```
   # vos create <machine name> <partition name> sysname -maxquota 0
   # fs mkmount sysname sysname
   ```

   If not creating a new volume:

   ```
   # mkdir sysname
   ```

2. In the directory for each system type, create subdirectories called **dest**, **dest/bin**, and **obj**. If you plan to use the *@sys* variable in pathnames that refer to these directories, then you must use the conventional system names listed in the *OpenAFS Release Notes*.

   ```
   # cd sysname
   # mkdir dest
   # mkdir dest/bin
   # mkdir obj
   ```

3. Create the indicated directories and symbolic links in the **/afs/.***cellname***/afs** directory.

   ```
   # cd /afs/.cellname/afs
   # ln -s  @sys/dest  dest
   # ln -s  @sys/obj  obj
   # ln -s  .  PARENT
   # ln -s  src/Makefile  Makefile
   ```

   The following is an example directory listing for the **/afs/.***cellname***/afs** directory after completing the preceding steps. It includes two example system types.

   ```
   lrwxr-xr-x admin   12 Jun 18 11:26 Makefile->src/Makefile
   lrwxr-xr-x admin    1 Jun 18 11:26 PARENT -> .
   lrwxr-xr-x admin    9 Jun 18 11:25 dest -> @sys/dest
   lrwxr-xr-x admin    8 Jun 18 11:25 obj -> @sys/obj
   drwxrwxrwx admin 4096 Jun 18 11:24 rcs
   drwxrwxrwx admin 2048 Jun 18 11:27 rs_aix42
   drwxrwxrwx admin 2048 Jun 18 11:10 src
   drwxrwxrwx admin 2048 Jun 18 11:27 sun4x_56
   ```

4. **(Optional)** By default, the build procedure writes its results into a destination directory for each system type called **/afs/.***cellname***/afs/***sysname***/dest**. To write the results to a different destination directory, create a link from the **dest** directory to it.

   ```
   # cd /afs/.cellname/afs/sysname
   # ln -s full_path_of_alternate_directory dest
   ```

5. For each system type you plan to build, run the following commands on a machine of that system type:

   ```
   # cd  /afs/cellname/sysname
   # ../src/configure
   # make
   # make dest
   ```

6. Working in the **/afs/.***cellname***/afs** directory on a machine of the system type for which you are building AFS, issue the **make install** command.

Appendix B

Appendix B. Configuring Legacy Components

This chapter describes how to configure a number of deprecated components in OpenAFS. Whilst these components are not recommended for sites performing new installations, it is recognised that there are a number of installations which have not yet transitioned from using these, for whom continued provision of installation instructions my be useful

B.1 kaserver and Legacy Kerberos 4 Authentication

This section contains instructions for installing server and client machines in sites which use either the deprecated AFS **kaserver** or legacy Kerberos 4 authentication systems

This should be used in conjuction with the installation instructures in earlier chapters, whose format it mirrors.

B.1.1 Background

As detailed in the OpenAFS "No more DES" roadmap, OpenAFS is moving away from the single DES based security models of both **kaserver** and external Kerberos 4 KDCs, in favour of using external, Kerberos 5 KDCs for authentication.

AFS version 3 was designed and implemented during the late 80s and early 90s when the state of the art in distributed computer authentication and data security was Kerberos 4 and single DES. The RXKAD security class was specified to use a single DES key and the kauth authentication protocol is a derivative of MIT's Kerberos 4 protocol.

For the better part of the last decade there has been concern regarding the cryptographic strength of the DES cipher when used as a building block within systems intended to prove authentication and/or data integrity and privacy. Kerberos 4 and RXKAD are not extensible and cannot negotiate non-DES key types. As a result efforts to migrate away from Kerberos 4 based authentication at higher risk organizations have been underway since the mid to late 90s. Ken Hornstein issued the first of his Kerberos 5 migration kits for AFS in May 1999.

In March 2003, the continued use of single DES and kauth as the basis for OpenAFS security became a real-world threat when a significant Kerberos 4 crossrealm vulnerability was published. The OpenAFS community was notified in security advisory OPENAFS-SA-2003-001 which can be found at http://www.openafs.org/security.

As a result of the mounting concerns regarding the strength of DES, NIST announced in May 2003 the withdrawal of FIPS 43-3 "Data Encryption Standard (DES)" as well as the associated FIPS 74 and FIPS 81. In other words, NIST announced that DES and its derivatives could no longer be used by the United States Government and should no longer by those that trust its lead.

In July 2003 MIT announced the end of life of the Kerberos 4 protocol which is distributed for backward compatibility as part of the MIT Kerberos 5 distribution.

B.1.2 Using this Appendix

This appendix should be read in conjunction with the instructions contained in the earlier chapters. It contains additions and in some cases, modifications, to the directions contained in those chapters. It is organised into 3 main sections, corresponding to the topics of the earlier chapters.

1. Installing the First AFS Machine
2. Installing Additional Server Machines
3. Installing Additonal Client Machines

There is an additional section on installing AFS login functionality, which is relevant to all machines which are operating as AFS clients

In addition, some general substitions should be made

- References to **kinit**and **aklog** should be replaced with a single call to **klog**

  For example

```
# kinit admin
Password:  admin_passwd
# aklog
```

  becomes

```
# klog admin
Password:  admin_passwd
```

B.1.3 Installing the First AFS machine

This section details changes to the installation procedure for the first AFS machine which are required in order to use **kaserver** for authentication. As detailed above, new sites are strongly discouraged from deploying kaserver.

The structure of this section follows the structure of the earlier chapter.

B.1.3.1 Overview: Installing Server Functionality

In adddition to the items described, you must also create the Authentication Server as a database server process. The procedure for creating the initial security mechanisms is also changed.

B.1.3.2 Starting the kaserver Database Server Process

In addition to the database server processes described, you must also use the **bos create** command to create an entry for the following process, which runs on database server machines only:

- The Authentication Server (the **kaserver** process) maintains the Authentication Database

The following instructions include the **-cell** argument on all applicable commands. Provide the cell name you assigned in Defining Cell Name and Membership for Server Processes. If a command appears on multiple lines, it is only for legibility. The following commands should run before any of the **bos create** commands detailed in Starting the Database Server Processes.

1. Issue the **bos create** command to start the Authentication Server. The current working directory is still **/usr/afs/bin**.

```
# ./bos create <machine name> kaserver simple /usr/afs/bin/kaserver  \
                -cell <cell name>  -noauth
```

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

2. Return to Starting the Database Server Processes and follow the remaining instructions

B.1.3.3 Initialising Cell Security with kaserver

Note
The following instructions should be followed in place of those in Initializing Cell Security

Begin by creating the following two initial entries in the Authentication Database:

- A generic administrative account, called **admin** by convention. If you choose to assign a different name, substitute it throughout the remainder of this document.

  After you complete the installation of the first machine, you can continue to have all administrators use the **admin** account, or you can create a separate administrative account for each of them. The latter scheme implies somewhat more overhead, but provides a more informative audit trail for administrative operations.

- The entry for AFS server processes, called **afs**. No user logs in under this identity, but the Authentication Server's Ticket Granting Service (TGS) module uses the associated key to encrypt the server tickets that it grants to AFS clients for presentation to server processes during mutual authentication. (The chapter in the *OpenAFS Administration Guide* about cell configuration and administration describes the role of server encryption keys in mutual authentication.)

  In Step 7, you also place the initial AFS server encryption key into the **/usr/afs/etc/KeyFile** file. The AFS server processes refer to this file to learn the server encryption key when they need to decrypt server tickets.

You also issue several commands that enable the new **admin** user to issue privileged commands in all of the AFS suites.

The following instructions do not configure all of the security mechanisms related to the AFS Backup System. See the chapter in the *OpenAFS Administration Guide* about configuring the Backup System.

1. Enter **kas** interactive mode. Because the machine is in no-authorization checking mode, include the **-noauth** flag to suppress the Authentication Server's usual prompt for a password.

   ```
   # kas  -cell <cell name> -noauth
   ka>
   ```

2. Issue the **kas create** command to create Authentication Database entries called **admin** and **afs**.

   Do not provide passwords on the command line. Instead provide them as *afs_passwd* and *admin_passwd* in response to the **kas** command interpreter's prompts as shown, so that they do not appear on the standard output stream.

   You need to enter the *afs_passwd* string only in this step and in Step 7, so provide a value that is as long and complex as possible, preferably including numerals, punctuation characters, and both uppercase and lowercase letters. Also make the *admin_passwd* as long and complex as possible, but keep in mind that administrators need to enter it often. Both passwords must be at least six characters long.

   ```
   ka> create afs
   initial_password:  afs_passwd
   Verifying, please re-enter initial_password: afs_passwd
   ka> create admin
   initial_password: admin_passwd
   Verifying, please re-enter initial_password: admin_passwd
   ```

3. Issue the **kas examine** command to display the **afs** entry. The output includes a checksum generated by encrypting a constant with the server encryption key derived from the *afs_passwd* string. In Step 8 you issue the **bos listkeys** command to verify that the checksum in its output matches the checksum in this output.

   ```
   ka> examine afs
   User data for afs
    key (0) cksum is checksum . . .
   ```

4. Issue the **kas setfields** command to turn on the ADMIN flag in the **admin** entry. This enables the **admin** user to issue privileged **kas** commands. Then issue the **kas examine** command to verify that the ADMIN flag appears in parentheses on the first line of the output, as shown in the example.

```
ka> setfields admin -flags admin
ka> examine admin
User data for admin (ADMIN) . . .
```

5. Issue the **kas quit** command to leave **kas** interactive mode.

```
ka> quit
```

6. Issue the **bos adduser** command to add the **admin** user to the **/usr/afs/etc/UserList** file. This enables the **admin** user to issue privileged **bos** and **vos** commands.

```
# ./bos adduser <machine name> admin -cell <cell name> -noauth
```

7. Issue the **bos addkey** command to define the AFS server encryption key in the **/usr/afs/etc/KeyFile** file.

   Do not provide the password on the command line. Instead provide it as *afs_passwd* in response to the **bos** command interpreter's prompts, as shown. Provide the same string as in Step 2.

```
# ./bos addkey <machine name> -kvno 0 -cell <cell name>  -noauth
Input key: afs_passwd
Retype input key: afs_passwd
```

8. Issue the **bos listkeys** command to verify that the checksum for the new key in the **KeyFile** file is the same as the checksum for the key in the Authentication Database's **afs** entry, which you displayed in Step 3.

```
# ./bos listkeys <machine name> -cell <ce
ll name> -noauth
key 0 has cksum checksum
```

   You can safely ignore any error messages indicating that **bos** failed to get tickets or that authentication failed.

   If the keys are different, issue the following commands, making sure that the *afs_passwd* string is the same in each case. The *checksum* strings reported by the **kas examine** and **bos listkeys** commands must match; if they do not, repeat these instructions until they do, using the **-kvno** argument to increment the key version number each time.

```
# ./kas  -cell <cell name> -noauth
ka> setpassword afs -kvno 1
new_password: afs_passwd
Verifying, please re-enter initial_password: afs_passwd
ka> examine afs
User data for afs
 key (1) cksum is checksum . . .
ka> quit
# ./bos addkey <machine name> -kvno 1 -cell <cell name> -noauth
Input key: afs_passwd
Retype input key: afs_passwd
# ./bos listkeys <machine name> -cell <cell name> -noauth
key 1 has cksum checksum
```

9. Proceed to Initializing the Protection Database to continue with the installation process

B.1.4 Installing Additional Server Machines

B.1.4.1 Starting the Authenticxation Service

In addition to the instructions in the main guide, you must also start the Authentication Server on the new database machine, as detailed below

1. Start the Authentication Server (the **kaserver** process).

```
% bos create <machine name> kaserver simple /usr/afs/bin/kaserver
```

2. Return to starting the backup server

B.1.5 Enabling AFS login with kaserver

The authentication system of every machine should be modified so that users obtain an AFS token as they log into the local file system. Using AFS is simpler and more convenient for your users if you make the modifications on all client machines. Otherwise users must perform a two step login procedure (login to the local system, and then issue the **klog** command.

For convenience, the following sections group this procedure by system type. Proceed to the appropriate section.

- Enabling AFS Login on Linux Systems
- Enabling AFS login on Solaris Systems

B.1.6 Enabling kaserver based AFS Login on Linux Systems

At this point you incorporate AFS into the operating system's Pluggable Authentication Module (PAM) scheme. PAM integrates all authentication mechanisms on the machine, including login, to provide the security infrastructure for authenticated access to and from the machine.

Explaining PAM is beyond the scope of this document. It is assumed that you understand the syntax and meanings of settings in the PAM configuration file (for example, how the `other` entry works, the effect of marking an entry as `required`, `optional`, or `sufficient`, and so on).

The following instructions explain how to alter the entries in the PAM configuration file for each service for which you wish to use AFS authentication. Other configurations possibly also work, but the instructions specify the recommended and tested configuration.

The recommended AFS-related entries in the PAM configuration file make use of one or more of the following three attributes.

AUTHENTICATION MANAGEMENT

try_first_pass This is a standard PAM attribute that can be included on entries after the first one for a service; it directs the module to use the password that was provided to the first module. For the AFS module, it means that AFS authentication succeeds if the password provided to the module listed first is the user's correct AFS password. For further discussion of this attribute and its alternatives, see the operating system's PAM documentation.

ignore_root This attribute, specific to the AFS PAM module, directs it to ignore not only the local superuser **root**, but also any user with UID 0 (zero).

ignore_uid *uid* This option is an extension of the "ignore_root" switch. The additional parameter is a limit. Users with a uid up to the given parameter are ignored by *pam_afs.so*. Thus, a system administrator still has the opportunity to add local user accounts to his system by choosing between "low" and "high" user ids. An example /etc/passwd file for "ignore_uid 100" may have entries like these:

```
        .
        .
afsuserone:x:99:100::/afs/afscell/u/afsuserone:/bin/bash
afsusertwo:x:100:100::/afs/afscell/u/afsusertwo:/bin/bash
localuserone:x:101:100::/home/localuserone:/bin/bash
localusertwo:x:102:100::/home/localusertwo:/bin/bash
        .
        .
```

AFS accounts should be locked in the file /etc/shadow like this:

```
        .
        .
afsuserone:!!:11500:0:99999:7:::
afsusertwo:!!:11500:0:99999:7:::
localuserone:<thelocaluserone'skey>:11500:0:99999:7:::
localusertwo:<thelocalusertwo'skey>:11500:0:99999:7:::
        .
        .
```

There is no need to store a local key in this file since the AFS password is sent and verfied at the AFS cell server!

setenv_password_expires This attribute, specific to the AFS PAM module, sets the environment variable PASSWORD_EXPIRE to the expiration date of the user's AFS password, which is recorded in the Authentication Database.

set_token Some applications don't call *pam_setcred()* in order to retrieve the appropriate credentials (here the AFS token) for their session. This switch sets the credentials already in *pam_sm_authenticate()* obsoleting a call to *pam_setcred()*. **Caution: Don't use this switch for applications which do call *pam_setcred()*!** One example for an application not calling *pam_setcred()* are older versions of the samba server. Nevertheless, using applications with working pam session management is recommended as this setup conforms better with the PAM definitions.

refresh_token This options is identical to "set_token" except that no new PAG is generated. This is necessary to handle processes like xlock or xscreensaver. It is not enough to just unlock the screen for a user who reactivated his session by typing in the correct AFS password, but one may also need fresh tokens with a full lifetime in order to work on, and the new token must be refreshed in the already existing PAG for the processes that have been started. This is achieved using this option.

use_klog Activating this switch causes authentication to be done by calling the external program "klog". One program requiring this is for example *kdm* of KDE 2.x.

dont_fork Usually, the password verification and token establishment is performed in a sub process. Using this option pam_afs does not fork and performs all actions in a single process. **Only use this option in cases where you notice serious problems caused by the sub process.** This option has been developed in respect to the "mod_auth_pam"-project (see also mod_auth_pam). The mod_auth_pam module enables PAM authentication for the apache http server package.

SESSION MANAGEMENT

no_unlog Normally the tokens are deleted (in memory) after the session ends. Using this option causes the tokens to be left untouched. **This behaviour was the default in pam_afs until openafs-1.1.1!**

remainlifetime ***sec*** The tokens are kept active for *sec* seconds before they are deleted. X display managers i.e. are used to inform the applications started in the X session before the logout and then end themselves. If the token was deleted immediately the applications would have no chance to write back their settings to i.e. the user's AFS home space. This option may help to avoid the problem.

Perform the following steps to enable AFS login.

1. Unpack the OpenAFS Binary Distribution for Linux into the **/tmp/afsdist/** directory, if it is not already. Then change to the directory for PAM modules, which depends on which Linux distribution you are using.

   If you are using a Linux distribution from Red Hat Software:

   ```
   # cd /lib/security
   ```

   If you are using another Linux distribution:

   ```
   # cd /usr/lib/security
   ```

2. Copy the appropriate AFS authentication library file to the directory to which you changed in the previous step. Create a symbolic link whose name does not mention the version. Omitting the version eliminates the need to edit the PAM configuration file if you later update the library file.

   If you use the AFS Authentication Server (**kaserver** process):

   ```
   # cp /cdrom/i386_linux22/lib/pam_afs.so.1  .
   # ln -s pam_afs.so.1 pam_afs.so
   ```

   If you use a Kerberos implementation of AFS authentication:

   ```
   # cp /cdrom/i386_linux22/lib/pam_afs.krb.so.1   .
   # ln -s pam_afs.krb.so.1 pam_afs.so
   ```

3. For each service with which you want to use AFS authentication, insert an entry for the AFS PAM module into the `auth` section of the service's PAM configuration file. (Linux uses a separate configuration file for each service, unlike some other operating systems which list all services in a single file.) Mark the entry as `sufficient` in the second field.

   Place the AFS entry below any entries that impose conditions under which you want the service to fail for a user who does not meet the entry's requirements. Mark these entries `required`. Place the AFS entry above any entries that need to execute only if AFS authentication fails.

   Insert the following AFS entry if using the Red Hat distribution:

   ```
   auth  sufficient  /lib/security/pam_afs.so   try_first_pass  ignore_root
   ```

   Insert the following AFS entry if using another distribution:

   ```
   auth  sufficient  /usr/lib/security/pam_afs.so  try_first_pass  ignore_root
   ```

   Check the PAM config files also for "session" entries. If there are lines beginning with "session" then please insert this line too:

   ```
   session  optional  /lib/security/pam_afs.so
   ```

   or

   ```
   session  optional  /usr/lib/security/pam_afs.so
   ```

   This guarantees that the user's tokens are deleted from memory after his session ends so that no other user coincidently gets those tokens without authorization! The following examples illustrate the recommended configuration of the configuration file for several services:

   AUTHENTICATION MANAGEMENT

   (/etc/pam.d/login)

   ```
   #%PAM-1.0
   auth      required   /lib/security/pam_securetty.so
   auth      required   /lib/security/pam_nologin.so
   auth      sufficient /lib/security/pam_afs.so try_first_pass ignore_root
   #                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   #This enables AFS authentication for every user but root
   auth      required   /lib/security/pam_pwdb.so shadow nullok
   account   required   /lib/security/pam_pwdb.so
   password  required   /lib/security/pam_cracklib.so
   password  required   /lib/security/pam_pwdb.so shadow nullok use_authtok
   session   optional   /lib/security/pam_afs.so
   #Make sure tokens are deleted after the user logs out
   session   required   /lib/security/pam_pwdb.so
   ```

(/etc/pam.d/samba)

```
auth       required     /lib/security/pam_afs.so ignore_uid 100 set_token
#                                                ^^^^^^^^^^^^^^^^^^^^^^^^
#Here, users with uid>100 are considered to belong to the AFS and users
#with uid<=100 are ignored by pam_afs. The token is retrieved already in
#pam_sm_authenticate() (this is an example pam config for a samba version
#that does not call pam_setcred(), it also does no sense to include session
#entries here since they would be ignored by this version of samba ).
account    required     /lib/security/pam_pwdb.so
```

(/etc/pam.d/xscreensaver)

```
auth       sufficient   /lib/security/pam_afs.so ignore_uid 100 refresh_token
#                                                               ^^^^^^^^^^^^^
#Avoid generating a new PAG for the new tokens, use the already existing PAG and
#establish a fresh token in it.
auth       required     /lib/security/pam_pwdb.so try_first_pass
```

(/etc/pam.d/httpd)

```
auth       required   /lib/security/pam_afs.so ignore_uid 100 dont_fork
#                                                            ^^^^^^^^^
#Don't fork for the verification of the password.
```

SESSION MANAGEMENT

(/etc/pam.d/su)

```
auth       sufficient   /lib/security/pam_afs.so ignore_uid 100
auth       required     /lib/security/pam_pwdb.so try_first_pass
account    required     /lib/security/pam_pwdb.so
password   required     /lib/security/pam_cracklib.so
password   required     /lib/security/pam_pwdb.so use_authtok
session    required     /lib/security/pam_pwdb.so
session    optional     /lib/security/pam_afs.so no_unlog
#                                               ^^^^^^^^
#Don't delete the token in this case, since the user may still
#need it (for example if somebody logs in and changes to root
#afterwards he may still want to access his home space in AFS).
session    required     /lib/security/pam_login_access.so
session    optional     /lib/security/pam_xauth.so
```

(/etc/pam.d/xdm)

```
auth       required     /lib/security/pam_nologin.so
auth       required     /lib/security/pam_login_access.so
auth       sufficient   /lib/security/pam_afs.so ignore_uid 100 use_klog
auth       required     /lib/security/pam_pwdb.so try_first_pass
account    required     /lib/security/pam_pwdb.so
password   required     /lib/security/pam_cracklib.so
password   required     /lib/security/pam_pwdb.so shadow nullok use_authtok
session    optional     /lib/security/pam_afs.so remainlifetime 10
#                                                ^^^^^^^^^^^^^^^^^
#Wait 10 seconds before deleting the AFS tokens in order to give
#the programs of the X session some time to save their settings
#to AFS.
session    required     /lib/security/pam_pwdb.so
```

4. After taking any necessary action, proceed to Starting the BOS Server if you are installing your first file server; Starting Server Programs if you are installing an additional file server machine; or Loading and Creating Client Files if you are installing a client.

B.1.7 Enabling kaserver based AFS Login on Solaris Systems

At this point you incorporate AFS into the operating system's Pluggable Authentication Module (PAM) scheme. PAM integrates all authentication mechanisms on the machine, including login, to provide the security infrastructure for authenticated access to and from the machine.

Explaining PAM is beyond the scope of this document. It is assumed that you understand the syntax and meanings of settings in the PAM configuration file (for example, how the `other` entry works, the effect of marking an entry as `required`, `optional`, or `sufficient`, and so on).

The following instructions explain how to alter the entries in the PAM configuration file for each service for which you wish to use AFS authentication. Other configurations possibly also work, but the instructions specify the recommended and tested configuration.

Note

The instructions specify that you mark each entry as `optional`. However, marking some modules as optional can mean that they grant access to the corresponding service even when the user does not meet all of the module's requirements. In some operating system revisions, for example, if you mark as optional the module that controls login via a dial-up connection, it allows users to login without providing a password. See the *OpenAFS Release Notes* for a discussion of any limitations that apply to this operating system.

Also, with some operating system versions you must install patches for PAM to interact correctly with certain authentication programs. For details, see the *OpenAFS Release Notes*.

The recommended AFS-related entries in the PAM configuration file make use of one or more of the following three attributes.

AUTHENTICATION MANAGEMENT

try_first_pass This is a standard PAM attribute that can be included on entries after the first one for a service; it directs the module to use the password that was provided to the first module. For the AFS module, it means that AFS authentication succeeds if the password provided to the module listed first is the user's correct AFS password. For further discussion of this attribute and its alternatives, see the operating system's PAM documentation.

ignore_root This attribute, specific to the AFS PAM module, directs it to ignore not only the local superuser **root**, but also any user with UID 0 (zero).

setenv_password_expires This attribute, specific to the AFS PAM module, sets the environment variable PASSWORD_EXPIRE to the expiration date of the user's AFS password, which is recorded in the Authentication Database.

Perform the following steps to enable AFS login.

1. Unpack the OpenAFS Binary Distribution for Solaris into the **/cdrom** directory, if it is not already. Then change directory as indicated.

   ```
   # cd /usr/lib/security
   ```

2. Copy the AFS authentication library file to the **/usr/lib/security** directory. Then create a symbolic link to it whose name does not mention the version. Omitting the version eliminates the need to edit the PAM configuration file if you later update the library file.

   If you use the AFS Authentication Server (**kaserver** process):

   ```
   # cp /tmp/afsdist/sun4x_56/dest/lib/pam_afs.so.1 .
   # ln -s pam_afs.so.1 pam_afs.so
   ```

   If you use a Kerberos implementation of AFS authentication:

   ```
   # cp /tmp/afsdist/sun4x_56/dest/lib/pam_afs.krb.so.1 .
   # ln -s pam_afs.krb.so.1 pam_afs.so
   ```

3. Edit the `Authentication management` section of the Solaris PAM configuration file, **/etc/pam.conf** by convention. The entries in this section have the value `auth` in their second field.

   First edit the standard entries, which refer to the Solaris PAM module (usually, the file **/usr/lib/security/pam_unix.so.1**) in their fourth field. For each service for which you want to use AFS authentication, edit the third field of its entry to read `optional`. The **pam.conf** file in the Solaris distribution usually includes standard entries for the **login**, **rlogin**, and **rsh** services, for instance.

   If there are services for which you want to use AFS authentication, but for which the **pam.conf** file does not already include a standard entry, you must create that entry and place the value `optional` in its third field. For instance, the Solaris **pam.conf** file does not usually include standard entries for the **ftp** or **telnet** services.

   Then create an AFS-related entry for each service, placing it immediately below the standard entry. The following example shows what the `Authentication Management` section looks like after you have you edited or created entries for the services mentioned previously. Note that the example AFS entries appear on two lines only for legibility.

```
login   auth  optional  /usr/lib/security/pam_unix.so.1
login   auth  optional  /usr/lib/security/pam_afs.so       \
      try_first_pass  ignore_root  setenv_password_expires
rlogin  auth  optional  /usr/lib/security/pam_unix.so.1
rlogin  auth  optional  /usr/lib/security/pam_afs.so       \
      try_first_pass  ignore_root  setenv_password_expires
rsh     auth  optional  /usr/lib/security/pam_unix.so.1
rsh     auth  optional  /usr/lib/security/pam_afs.so       \
      try_first_pass  ignore_root
ftp     auth  optional  /usr/lib/security/pam_unix.so.1
ftp     auth  optional  /usr/lib/security/pam_afs.so       \
      try_first_pass  ignore_root
telnet  auth  optional  /usr/lib/security/pam_unix.so.1
telnet  auth  optional  /usr/lib/security/pam_afs.so       \
      try_first_pass  ignore_root  setenv_password_expires
```

4. If you use the Common Desktop Environment (CDE) on the machine and want users to obtain an AFS token as they log in, also add or edit the following four entries in the `Authentication management` section. Note that the AFS-related entries appear on two lines here only for legibility.

```
dtlogin   auth  optional  /usr/lib/security/pam_unix.so.1
dtlogin   auth  optional  /usr/lib/security/pam_afs.so     \
      try_first_pass  ignore_root
dtsession  auth  optional /usr/lib/security/pam_unix.so.1
dtsession  auth  optional /usr/lib/security/pam_afs.so     \
      try_first_pass  ignore_root
```

5. Proceed to Editing the File Systems Clean-up Script on Solaris Systems in the server instructions if you are installing your first file server; Starting Server Programs if you are installing an additional file server machine; or Editing the File Systems Clean-up Script on Solaris Systems in the client instructions if you are installing a client.

Appendix C

The Demand-Attach File Server

This chapter explains and describes the Demand-Attach File Server (DAFS for short), a new feature in OpenAFS 1.6. A few things are different when setting up and maintaining a DAFS fileserver vs a traditional fileserver, so it is important to know which one you are running.

If you are trying to decide whether or not to run DAFS on a fileserver, note that you can switch between DAFS and the traditional fileserver fairly easily at any time. Aside from performance differences, the two fileservers generally look and act similarly, so choosing one over the other in the beginning should not hurt you later on. So, if you are not overly concerned with performance, you can just pick one and follow the directions for it, and you should be fine.

C.1 Justification and Background

DAFS changes many things with how the fileserver and other server processes access volumes on disk. Most of these changes are only of concern to developers, as there are no visible differences to users or administrators. A few changes are only of concern to administrators while debugging a problem, and only one or two changes affect the day-to-day administration of a fileserver. See the sections on DAFS Binaries and Salvaging for the main functional differences.

Among other things, DAFS provides a new way for the fileserver and other server programs to access volumes. It does not change how file data is accessed inside a volume (like namei or inode do), but rather how a volume as a whole is accessed by the fileserver. When a traditional fileserver is started, it will locate and attach all AFS volumes it can find on all AFS partitions on the server. This attachment process involves reading the volume header and setting a flag to say that the volume is in use. On a clean shutdown, the fileserver detach all volumes it attached by clearing that flag. If the fileserver encounters a volume where the flag is already set before it attached the volume, it will know that volume was not detached cleanly, and that it needs to be salvaged.

This process of attaching and detaching at startup and shutdown takes time. When fileservers start to have thousands of volumes, it can take several minutes to start or cleanly shutdown. Additionally, when a fileserver is shutdown uncleanly, all of the volumes on the server must be salvaged before the fileserver can be brought up again. Even volumes that nobody has ever accessed must be salvaged.

To improve this situation, DAFS does not attach all volumes on startup. Instead, volumes are only attached when they are first accessed, and are detached if idle for a long period of time (see the -vlru* options for the fileserver; this is configurable but is generally several hours).

Thus, after an unclean shutdown with DAFS, only the volumes in use at the time of the shutdown need to be salvaged. In addition, the fileserver does not need to wait for salvaging to finish before starting up. Instead of salvaging everything before the fileserver starts, the fileserver causes salvages to be issued on a volume when a damanged volume is accessed (called "demand-salvages").

The result of all of this is that a fileserver running DAFS can generally be started and stopped in a manner of seconds, where a traditional fileserver could take dozens of minutes, or even hours in the case of an unclean shutdown.

C.2 DAFS Binaries

OpenAFS ships with binaries for DAFS and non-DAFS alongside each other. Programs that exist in both DAFS and non-DAFS variants typically have a **da** prefix to indicate the DAFS variant. For example, **dafileserver** is the DAFS version of the traditional **fileserver** binary. Similarly for **davolserver** and **volserver**, **dasalvager** and **salvager**, and even some other tools like **dafssync-debug** and **fssync-debug**.

A DAFS-enabled fileserver will run the **dafs** bnode in bosserver, instead of the traditional fileserver's **fs** bnode. The **dafs** bnode runs four different binaries as opposed to the **fs** bnode's three. Three of the programs in the **dafs** bnode are just DAFS equivalents of the corresponding **fs** bnode programs, but the fourth one is an entirely new daemon that has no non-DAFS equivalent: the salvageserver.

Traditional binary	DAFS binary
/usr/afs/bin/fileserver	**/usr/afs/bin/dafileserver**
/usr/afs/bin/volserver	**/usr/afs/bin/davolserver**
No equivalent	**/usr/afs/bin/salvageserver**
/usr/afs/bin/salvager	**/usr/afs/bin/dasalvager**

C.3 Salvaging

With a traditional fileserver, salvages usually occur in two situations:

1. The fileserver shuts down uncleanly, and when brought back up, all partitions are salvaged before the fileserver is available.
2. A volume experiences some corruption after it has been brought online, and an administrator manually schedules a salvage for an individual volume with **bos salvage**. Usually the way you notice this is that the fileserver noticed a volume has become corrupt and has taken it offline.

With DAFS, neither of these occur in normal operation. With DAFS, a volume will be salvaged automatically when a problem is detected in a volume. This occurs whether the fileserver detects the volume was not detached cleanly, or if the fileserver detects corrupt volume data.

In normal DAFS operation, you should not need to ever run **bos salvage**. However, if you suspect a bug, or that there is corruption in a volume that the fileserver has not detected, you can run **bos salvage** to manually issue a salvage.

All salvages scheduled from the salvageserver will be logged in **/usr/afs/logs/SalsrvLog**, in the same format as salvages for traditional fileservers are logged. If you issue a whole-partition or whole-server salvage, the logs for that will still be located in **/usr/afs/logs/SalvageLog**.

C.4 Converting a Fileserver to DAFS

If you have an existing traditional fileserver, you can convert it to DAFS fairly easily. The conversion is in-place, but there is at least a small amount of downtime involved even if nothing goes wrong, since the fileserver processes must be restarted. If you want to avoid any downtime, move any volumes on the fileserver to another fileserver before performing the conversion.

1. If the fileserver is currently running a version of OpenAFS earlier than 1.6.0, upgrade the fileserver to a version in the 1.6 series or later. This is not strictly necessary, but if you encounter problems here, it is helpful to catch them before converting to DAFS so as to isolate the problem.

   If you do not upgrade the fileserver before converting to DAFS, install the new OpenAFS binaries somewhere on the server.

2. If the current bosserver process running on the fileserver is older than OpenAFS 1.6.0, you must upgrade it first. This can be done by replacing the bosserver binary and running **bos restart -bosserver**. This will cause *all* AFS processes controlled by the bosserver to be restarted. You can do this at the same as upgrading the fileserver in the previous step if desired, to avoid restarting the fileserver twice.

3. Shutdown and stop the old **fs** bnode:

```
# bos stop <machine name> fs -wait -cell <cell name>
```

4. Create and start the new **dafs** bnode.

```
# bos create  <machine name> dafs dafs \
              -cmd '/usr/afs/bin/dafileserver <dafileserver options>' \
              -cmd '/usr/afs/bin/davolserver <davolserver options>' \
              -cmd '/usr/afs/bin/salvageserver <salvageserver options>' \
              -cmd '/usr/afs/bin/dasalvager <dasalvager options>' \
              -cell <cell name>
```

   You can use the same options for the **dafileserver** process as you did for the **fileserver** process, and the same options for **davolserver** as you did for **volserver**.

   You can also use most of the same options for the **salvageserver** and **dasalvager** processes as you did for the **salvager** process; see their respective man pages. However, if you are upgrading from the 1.4.x series of OpenAFS, be aware that the **-DontSalvage** option does not exist anymore for the salvager (with or without DAFS).

   Also note that the **dafileserver** process has many more options to tune dafs-specific parameters. See the **dafileserver** man page for information about them.

After you have performed these steps, switching back and forth between running a DAFS and a traditional fileserver is as simple a running **bos stop** on one bnode, and **bos start** on the other. Once you are confident that the DAFS processes are running smoothly and you do not anticipate switching back to the traditional fileserver, you can **bos delete** the **fs** bnode.

Chapter 5

Index