AFS Administration Guide

Version 3.6

AFS Administration Guide: Version 3.6

This edition applies to:
IBM AFS for AIX, Version 3.6
IBM AFS for Digital Unix, Version 3.6
IBM AFS for HP-UX, Version 3.6
IBM AFS for Linux, Version 3.6
IBM AFS for SGI IRIX, Version 3.6
IBM AFS for Solaris, Version 3.6

and to all subsequent releases and modifications until otherwise indicated in new editions.This softcopy version is based on the printed edition of this book. Some formatting amendments have been made to make this information more suitable for softcopy.

Revision History

Revision 3.6 April 2000
First Edition

Table of Contents

List of Tables

List of Figures

About This Guide

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

Audience and Purpose

This guide describes the concepts and procedures that an AFS(R) system administrator needs to know. It assumes familiarity with UNIX(R) administration, but no previous knowledge of AFS.

This document describes AFS commands in the context of specific tasks. Thus, it does not describe all commands in detail. Refer to the IBM AFS Administration Reference for detailed command descriptions.

Document Organization

This document groups AFS administrative tasks into the following conceptual sections:

- Concepts and Configuration Issues
- Managing File Server Machines
- Managing Client Machines
- Managing Users and Groups

The individual chapters in each section contain the following:

- A chapter overview
- A quick reference list of the tasks and commands described in the chapter
- An introduction to concepts that pertain to all of the tasks described in the chapter
- A set of sections devoted to specific tasks. Each section begins with a discussion of concepts specific to that task, followed by step-by-step instructions for performing the task. The instructions are as specific as has been judged practical. If two related procedures differ from one another in important details, separate sets of instructions are usually provided.

How to Use This Document

When you need to perform a specific administrative task, follow these steps:

1. Determine if the task concerns file server machines, client machines, or users and groups. Turn to the appropriate section in this document and then to the appropriate chapter.
2. Read or review the general introductory material at the beginning of the chapter.
3. Read or review the introductory material concerning the specific task you wish to perform.
4. Follow the step-by-step instructions for the task.

5. If necessary, refer to the IBM AFS Administration Reference for more detailed information about the commands.

Related Documents

The following documents are also included in the AFS documentation set.

IBM AFS 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 IBM AFS 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 IBM AFS Administration Guide: it does not include procedural information, but describes commands in more detail than the IBM AFS Administration Guide.

IBM AFS Quick Beginnings

This guide provides instructions for installing AFS server and client machines. It is assumed that the installer is an experienced UNIX(R) system administrator.

For predictable performance, machines must be installed and configured in accordance with the instructions in this guide.

IBM AFS 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.

IBM AFS 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.

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 "Appendix B, Using AFS Commands" on page 555.

I. Concepts and Configuration Issues

Chapter 1. An Overview of AFS Administration

This chapter provides a broad overview of the concepts and organization of AFS. It is strongly recommended that anyone involved in administering an AFS cell read this chapter before beginning to issue commands.

A Broad Overview of AFS

This section introduces most of the key terms and concepts necessary for a basic understanding of AFS. For a more detailed discussion, see "More Detailed Discussions of Some Basic Concepts" on page 3.

AFS: A Distributed File System

AFS is a distributed file system that enables users to share and access all of the files stored in a network of computers as easily as they access the files stored on their local machines. The file system is called distributed for this exact reason: files can reside on many different machines (be distributed across them), but are available to users on every machine.

Servers and Clients

In fact, AFS stores files on a subset of the machines in a network, called file server machines. File server machines provide file storage and delivery service, along with other specialized services, to the other subset of machines in the network, the client machines. These machines are called clients because they make use of the servers' services while doing their own work. In a standard AFS configuration, clients provide computational power, access to the files in AFS and other "general purpose" tools to the users seated at their consoles. There are generally many more client workstations than file server machines.

AFS file server machines run a number of server processes, so called because each provides a distinct specialized service: one handles file requests, another tracks file location, a third manages security, and so on. To avoid confusion, AFS documentation always refers to server machines and server processes, not simply to servers. For a more detailed description of the server processes, see "AFS Server Processes and the Cache Manager" on page 8.

Cells

A cell is an administratively independent site running AFS. As a cell's system administrator, you make many decisions about configuring and maintaining your cell in the way that best serves its users, without having to consult the administrators in other cells. For example, you determine how many clients and servers to have, where to put files, and how to allocate client machines to users.

Transparent Access and the Uniform Namespace

Although your AFS cell is administratively independent, you probably want to organize the local collection of files (your filespace or tree) so that users from other cells can also access the information in it. AFS enables cells to combine their local filespaces into a global filespace, and does so in such a way that file access is transparent--users do not need to know anything about a file's location in order to access it. All they need to know is the pathname of the file, which looks the same in every cell. Thus every user at every machine sees the collection of files in the same way, meaning that AFS provides a uniform namespace to its users.

Volumes

AFS groups files into volumes, making it possible to distribute files across many machines and yet maintain a uniform namespace. A volume is a unit of disk space that functions like a container for a set of related files, keeping them all together on one partition. Volumes can vary in size, but are (by definition) smaller than a partition.

Volumes are important to system administrators and users for several reasons. Their small size makes them easy to move from one partition to another, or even between machines. The system administrator can maintain maximum efficiency by moving volumes to keep the load balanced evenly. In addition, volumes correspond to directories in the filespace--most cells store the contents of each user home directory in a separate volume. Thus the complete contents of the directory move together when the volume moves, making it easy for AFS to keep track of where a file is at a certain time. Volume moves are recorded automatically, so users do not have to keep track of file locations.

Efficiency Boosters: Replication and Caching

AFS incorporates special features on server machines and client machines that help make it efficient and reliable.

On server machines, AFS enables administrators to replicate commonly-used volumes, such as those containing binaries for popular programs. Replication means putting an identical read-only copy (sometimes called a clone) of a volume on more than one file server machine. The failure of one file server machine housing the volume does not interrupt users' work, because the volume's contents are still available from other machines. Replication also means that one machine does not become overburdened with requests for files from a popular volume.

On client machines, AFS uses caching to improve efficiency. When a user on a client workstation requests a file, the Cache Manager on the client sends a request for the data to the File Server process running on the proper file server machine. The user does not need to know which machine this is; the Cache Manager determines file location automatically. The Cache Manager receives the file from the File Server process and puts it into the cache, an area of the client machine's local disk or memory dedicated to temporary file storage. Caching improves efficiency because the client does not need to send a request across the network every time the user wants the same file. Network traffic is minimized, and subsequent access to the file is especially fast because the file is stored locally. AFS has a way of ensuring that the cached file stays up-to-date, called a callback.

Security: Mutual Authentication and Access Control Lists

Even in a cell where file sharing is especially frequent and widespread, it is not desirable that every user have equal access to every file. One way AFS provides adequate security is by requiring that servers and clients prove their identities to one another before they exchange information. This procedure, called mutual authentication, requires that both server and client demonstrate knowledge of a "shared secret" (like a password) known only to the two of them. Mutual authentication guarantees that servers provide information only to authorized clients and that clients receive information only from legitimate servers.

Users themselves control another aspect of AFS security, by determining who has access to the directories they own. For any directory a user owns, he or she can build an access control list (ACL) that grants or denies access to the contents of the directory. An access control list pairs specific users with specific types of access privileges. There are seven separate permissions and up to twenty different people or groups of people can appear on an access control list.

For a more detailed description of AFS's mutual authentication procedure, see "A More Detailed Look at Mutual Authentication" on page 51. For further discussion of ACLs, see "Managing Access Control Lists" on page 513.

More Detailed Discussions of Some Basic Concepts

The previous section offered a brief overview of the many concepts that an AFS system administrator needs to understand. The following sections examine some important concepts in more detail. Although not all concepts are new to an experienced administrator, reading this section helps ensure a common understanding of term and concepts.

Networks

A *network* is a collection of interconnected computers able to communicate with each other and transfer information back and forth.

A networked computing environment contrasts with two types of computing environments: *mainframe* and *personal*.

- A *mainframe* computing environment is the most traditional. It uses a single powerful computer (the mainframe) to do the majority of the work in the system, both file storage and computation. It serves many users, who access their files and issue commands to the mainframe via terminals, which generally have only enough computing power to accept input from a keyboard and to display data on the screen.
- A *personal* computing environment is a single small computer that serves one (or, at the most, a few) users. Like a mainframe computer, the single computer stores all the files and performs all computation. Like a terminal, the personal computer provides access to the computer through a keyboard and screen.

A network can connect computers of any kind, but the typical network running AFS connects high-function personal workstations. Each workstation has some computing power and local disk space, usually more than a personal computer or terminal, but less than a mainframe. For more about the classes of machines used in an AFS environment, see "Servers and Clients" on page 4.

Distributed File Systems

A *file system* is a collection of files and the facilities (programs and commands) that enable users to access the information in the files. All computing environments have file systems. In a mainframe environment, the file system consists of all the files on the mainframe's storage disks, whereas in a personal computing environment it consists of the files on the computer's local disk.

Networked computing environments often use *distributed file systems* like AFS. A distributed file system takes advantage of the interconnected nature of the network by storing files on more than one computer in the network and making them accessible to all of them. In other words, the responsibility for file storage and delivery is "distributed" among multiple machines instead of relying on only one. Despite the distribution of responsibility, a distributed file system like AFS creates the illusion that there is a single filespace.

Servers and Clients

AFS uses a server/client model. In general, a server is a machine, or a process running on a machine, that provides specialized services to other machines. A client is a machine or process that makes use of a server's specialized service during the course of its own work, which is often of a more general nature than the server's. The functional distinction between clients and server is not always strict, however--a server can be considered the client of another server whose service it is using.

AFS divides the machines on a network into two basic classes, *file server machines* and *client machines*, and assigns different tasks and responsibilities to each.

File Server Machines

File server machines store the files in the distributed file system, and a *server process* running on the file server machine delivers and receives files. AFS file server machines run a number of *server processes*. Each process has a special function, such as maintaining databases important to AFS administration, managing security or handling volumes. This modular design enables each server process to specialize in one area, and thus perform more efficiently. For a description of the function of each AFS server process, see "AFS Server Processes and the Cache Manager" on page 8.

Not all AFS server machines must run all of the server processes. Some processes run on only a few machines because the demand for their services is low. Other processes run on only one machine in order to act as a synchronization site. See "The Four Roles for File Server Machines" on page 68.

Client Machines

The other class of machines are the *client machines*, which generally work directly for users, providing computational power and other general purpose tools. Clients also provide users with access to the files stored on the file server machines. Clients do not run any special processes per se, but do use a modified kernel that enables them to communicate with the AFS server processes running on the file server machines and to cache files. This collection of kernel modifications is referred to as the Cache Manager; see "The Cache Manager" on page 13. There are usually many more client machines in a cell than file server machines.

Client and Server Configuration

In the most typical AFS configuration, both file server machines and client machines are high-function workstations with disk drives. While this configuration is not required, it does have some advantages.

There are several advantages to using personal workstations as file server machines. One is that it is easy to expand the network by adding another file server machine. It is also easy to increase storage space by adding disks to existing machines. Using workstations rather than more powerful mainframes makes it more economical to use multiple file server machines rather than one. Multiple file server machines provide an increase in system availability and reliability if popular files are available on more than one machine.

The advantage of using workstations as clients is that caching on the local disk speeds the delivery of files to application programs. (For an explanation of caching, see "Caching and Callbacks" on page 7.) Diskless machines can access AFS if they are running NFS(R) and the NFS/AFS Translator, an optional component of the AFS distribution.

Cells

A *cell* is an independently administered site running AFS. In terms of hardware, it consists of a collection of file server machines and client machines defined as belonging to the cell; a machine can only belong to one cell at a time. Users also belong to a cell in the sense of having an account in it, but unlike machines can belong to (have an account in) multiple cells. To say that a cell is administratively independent means that its administrators determine many details of its configuration without having to consult administrators in other cells or a central authority. For example, a cell administrator determines how many machines of different types to run, where to put files in the local tree, how to associate volumes and directories, and how much space to allocate to each user.

The terms *local cell* and *home cell* are equivalent, and refer to the cell in which a user has initially authenticated during a session, by logging onto a machine that belongs to that cell. All other cells are referred to as *foreign* from the user's perspective. In other words, throughout a login session, a user is accessing the filespace through a single Cache Manager--the one on the machine to which he or she initially logged in--whose cell membership defines the local cell. All other cells are considered foreign during that login session, even if the user authenticates in additional cells or uses the **cd** command to change directories into their file trees.

It is possible to maintain more than one cell at a single geographical location. For instance, separate departments on a university campus or in a corporation can choose to administer their own cells. It is also possible to have machines at geographically distant sites belong to the same cell; only limits on the speed of network communication determine how practical this is.

Despite their independence, AFS cells generally agree to make their local filespace visible to other AFS cells, so that users in different cells can share files if they choose. If your cell is to participate in the "global" AFS namespace, it must comply with a few basic conventions governing how the local filespace is configured and how the addresses of certain file server machines are advertised to the outside world.

The Uniform Namespace and Transparent Access

One of the features that makes AFS easy to use is that it provides transparent access to the files in a cell's filespace. Users do not have to know which file server machine stores a file in order to access it; they simply provide the file's pathname, which AFS automatically translates into a machine location.

In addition to transparent access, AFS also creates a *uniform namespace*--a file's pathname is identical regardless of which client machine the user is working on. The cell's file tree looks the same when viewed from any client because the cell's file server machines store all the files centrally and present them in an identical manner to all clients.

To enable the transparent access and the uniform namespace features, the system administrator must follow a few simple conventions in configuring client machines and file trees. For details, see "Making Other Cells Visible in Your Cell" on page 23.

Volumes

A *volume* is a conceptual container for a set of related files that keeps them all together on one file server machine partition. Volumes can vary in size, but are (by definition) smaller than a partition. Volumes are the main administrative unit in AFS, and have several characteristics that make administrative tasks easier and help improve overall system performance.

- The relatively small size of volumes makes them easy to move from one partition to another, or even between machines.
- You can maintain maximum system efficiency by moving volumes to keep the load balanced evenly among the different machines. If a partition becomes full, the small size of individual volumes makes it easy to find enough room on other machines for them.
- Each volume corresponds logically to a directory in the file tree and keeps together, on a single partition, all the data that makes up the files in the directory. By maintaining (for example) a separate volume for each user's home directory, you keep all of the user's files together, but separate from those of other users. This is an administrative convenience that is impossible if the partition is the smallest unit of storage.
- The directory/volume correspondence also makes transparent file access possible, because it simplifies the process of file location. All files in a directory reside together in one volume and in order to find a file, a file server process need only know the name of the file's parent directory, information which is included in the file's pathname. AFS knows how to translate the directory name into a volume name, and automatically tracks every volume's location, even when a volume is moved from machine to machine. For more about the directory/volume correspondence, see "Mount Points" on page 6.
- Volumes increase file availability through replication and backup.
- Replication (placing copies of a volume on more than one file server machine) makes the contents more reliably available; for details, see "Replication" on page 7. Entire sets of volumes can be backed up to tape and restored to the file system; see "Configuring the AFS Backup System" on page 195 and "Backing Up and Restoring AFS Data" on page 241. In AFS, backup also refers to recording the state of a volume at a certain time and then storing it (either on tape or elsewhere in the file system) for recovery in the event files in it are accidentally deleted or changed. See "Creating Backup Volumes" on page 144.
- Volumes are the unit of resource management. A space quota associated with each volume sets a limit on the maximum volume size. See "Setting and Displaying Volume Quota and Current Size" on page 176.

Mount Points

The previous section discussed how each volume corresponds logically to a directory in the file system: the volume keeps together on one partition all the data in the files residing in the directory. The directory that corresponds to a volume is called its *root directory*, and the mechanism that associates the directory and volume is called a *mount point*. A mount point is similar to a symbolic link in the file tree that specifies which volume contains the files kept in a directory. A mount point is not an actual symbolic link; its internal structure is different.

> **Note:** You must not create a symbolic link to a file whose name begins with the number sign (#) or the percent sign (%), because the Cache Manager interprets such a link as a mount point to a regular or read/write volume, respectively.

The use of mount points means that many of the elements in an AFS file tree that look and function just like standard UNIX file system directories are actually mount points. In form, a mount point is a one-line file that names the volume containing the data for files in the directory. When the Cache Manager (see "The Cache Manager" on page 13) encounters a mount point--for example, in the course of interpreting a pathname--it looks in the volume named in the mount point. In the volume the Cache Manager finds an actual UNIX-style directory element--the volume's root directory--that lists the files contained in the directory/volume. The next element in the pathname appears in that list.

A volume is said to be *mounted* at the point in the file tree where there is a mount point pointing to the volume. A volume's contents are not visible or accessible unless it is mounted.

Replication

Replication refers to making a copy, or *clone*, of a source read/write volume and then placing the copy on one or more additional file server machines in a cell. One benefit of replicating a volume is that it increases the availability of the contents. If one file server machine housing the volume fails, users can still access the volume on a different machine. No one machine need become overburdened with requests for a popular file, either, because the file is available from several machines.

Replication is not necessarily appropriate for cells with limited disk space, nor are all types of volumes equally suitable for replication (replication is most appropriate for volumes that contain popular files that do not change very often). For more details, see "When to Replicate Volumes" on page 29.

Caching and Callbacks

Just as replication increases system availability, *caching* increases the speed and efficiency of file access in AFS. Each AFS client machine dedicates a portion of its local disk or memory to a cache where it stores data temporarily. Whenever an application program (such as a text editor) running on a client machine requests data from an AFS file, the request passes through the Cache Manager. The Cache Manager is a portion of the client machine's kernel that translates file requests from local application programs into cross-network requests to the *File Server process* running on the file server machine storing the file. When the Cache Manager receives the requested data from the File Server, it stores it in the cache and then passes it on to the application program.

Caching improves the speed of data delivery to application programs in the following ways:

- When the application program repeatedly asks for data from the same file, it is already on the local disk. The application does not have to wait for the Cache Manager to request and receive the data from the File Server.
- Caching data eliminates the need for repeated request and transfer of the same data, so network traffic is reduced. Thus, initial requests and other traffic can get through more quickly.

While caching provides many advantages, it also creates the problem of maintaining consistency among the many cached copies of a file and the source version of a file. This problem is solved using a mechanism referred to as a *callback*.

A callback is a promise by a File Server to a Cache Manager to inform the latter when a change is made to any of the data delivered by the File Server. Callbacks are used differently based on the type of file delivered by the File Server:

- When a File Server delivers a writable copy of a file (from a read/write volume) to the Cache Manager, the File Server sends along a callback with that file. If the source version of the file is changed by another user, the File Server breaks the callback associated with the cached version of that file--indicating to the Cache Manager that it needs to update the cached copy.
- When a File Server delivers a file from a read-only volume to the Cache Manager, the File Server sends along a callback associated with the entire volume (so it does not need to send any more callbacks when it delivers additional files from the volume). Only a single callback is required per accessed read-only volume because files in a read-only volume can change only when a new version of the complete volume is released. All callbacks associated with the old version of the volume are broken at release time.

The callback mechanism ensures that the Cache Manager always requests the most up-to-date version of a file. However, it does not ensure that the user necessarily notices the most current version as soon as the Cache Manager has it. That depends on how often the application program requests additional data from the File System or how often it checks with the Cache Manager.

AFS Server Processes and the Cache Manager

As mentioned in "Servers and Clients" on page 4, AFS file server machines run a number of processes, each with a specialized function. One of the main responsibilities of a system administrator is to make sure that processes are running correctly as much of the time as possible, using the administrative services that the server processes provide.

The following list briefly describes the function of each server process and the Cache Manager; the following sections then discuss the important features in more detail.

The *File Server*, the most fundamental of the servers, delivers data files from the file server machine to local workstations as requested, and stores the files again when the user saves any changes to the files.

The *Basic OverSeer Server (BOS Server)* ensures that the other server processes on its server machine are running correctly as much of the time as possible, since a server is useful only if it is available. The BOS Server relieves system administrators of much of the responsibility for overseeing system operations.

The *Authentication Server* helps ensure that communications on the network are secure. It verifies user identities at login and provides the facilities through which participants in transactions prove their identities to one another (mutually authenticate). It maintains the Authentication Database.

The Protection Server helps users control who has access to their files and directories. Users can grant access to several other users at once by putting them all in a group entry in the Protection Database maintained by the Protection Server.

The *Volume Server* performs all types of volume manipulation. It helps the administrator move volumes from one server machine to another to balance the workload among the various machines.

The *Volume Location Server (VL Server)* maintains the Volume Location Database (VLDB), in which it records the location of volumes as they move from file server machine to file server machine. This service is the key to transparent file access for users.

The *Update Server* distributes new versions of AFS server process software and configuration information to all file server machines. It is crucial to stable system performance that all server machines run the same software.

The *Backup Server* maintains the Backup Database, in which it stores information related to the Backup System. It enables the administrator to back up data from volumes to tape. The data can then be restored from tape in the event that it is lost from the file system.

The *Salvager* is not a server in the sense that others are. It runs only after the File Server or Volume Server fails; it repairs any inconsistencies caused by the failure. The system administrator can invoke it directly if necessary.

The *Network Time Protocol Daemon (NTPD)* is not an AFS server process per se, but plays a vital role nonetheless. It synchronizes the internal clock on a file server machine with those on other machines. Synchronized clocks are particularly important for correct functioning of the AFS distributed database technology (known as Ubik); see "Configuring the Cell for Proper Ubik Operation" on page 74. The NTPD is controlled by the **runntp** process.

The *Cache Manager* is the one component in this list that resides on AFS client rather than file server machines. It not a process per se, but rather a part of the kernel on AFS client machines that communicates with AFS server processes. Its main responsibilities are to retrieve files for application programs running on the client and to maintain the files in the cache.

The File Server

The *File Server* is the most fundamental of the AFS server processes and runs on each file server machine. It provides the same services across the network that the UNIX file system provides on the local disk:

- Delivering programs and data files to client workstations as requested and storing them again when the client workstation finishes with them.
- Maintaining the hierarchical directory structure that users create to organize their files.
- Handling requests for copying, moving, creating, and deleting files and directories.
- Keeping track of status information about each file and directory (including its size and latest modification time).

- Making sure that users are authorized to perform the actions they request on particular files or directories.
- Creating symbolic and hard links between files.
- Granting advisory locks (corresponding to UNIX locks) on request.

The Basic OverSeer Server

The *Basic OverSeer Server (BOS Server)* reduces the demands on system administrators by constantly monitoring the processes running on its file server machine. It can restart failed processes automatically and provides a convenient interface for administrative tasks.

The BOS Server runs on every file server machine. Its primary function is to minimize system outages. It also

- Constantly monitors the other server processes (on the local machine) to make sure they are running correctly.
- Automatically restarts failed processes, without contacting a human operator. When restarting multiple server processes simultaneously, the BOS server takes interdependencies into account and initiates restarts in the correct order.
- Accepts requests from the system administrator. Common reasons to contact BOS are to verify the status of server processes on file server machines, install and start new processes, stop processes either temporarily or permanently, and restart dead processes manually.
- Helps system administrators to manage system configuration information. The BOS server automates the process of adding and changing *server encryption keys*, which are important in mutual authentication. The BOS Server also provides a simple interface for modifying two files that contain information about privileged users and certain special file server machines. For more details about these configuration files, see "Common Configuration Files in the /usr/afs/etc Directory" on page 62.

The Authentication Server

The *Authentication Server* performs two main functions related to network security:

- Verifying the identity of users as they log into the system by requiring that they provide a password. The Authentication Server grants the user a token as proof to AFS server processes that the user has authenticated. For more on tokens, see "Complex Mutual Authentication" on page 52.
- Providing the means through which server and client processes prove their identities to each other (mutually authenticate). This helps to create a secure environment in which to send cross-network messages.

In fulfilling these duties, the Authentication Server utilizes algorithms and other procedures known as *Kerberos* (which is why many commands used to contact the Authentication Server begin with the letter

k). This technology was originally developed by the Massachusetts Institute of Technology's Project Athena.

The Authentication Server also maintains the *Authentication Database*, in which it stores user passwords converted into encryption key form as well as the AFS server encryption key. To learn more about the procedures AFS uses to verify user identity and during mutual authentication, see "A More Detailed Look at Mutual Authentication" on page 51.

The Protection Server

The *Protection Server* is the key to AFS's refinement of the normal UNIX methods for protecting files and directories from unauthorized use. The refinements include the following:

- Defining seven access permissions rather than the standard UNIX file system's three. In conjunction with the UNIX mode bits associated with each file and directory element, AFS associates an *access control list (ACL)* with each directory. The ACL specifies which users have which of the seven specific permissions for the directory and all the files it contains. For a definition of AFS's seven access permissions and how users can set them on access control lists, see "Managing Access Control Lists" on page 513.
- Enabling users to grant permissions to numerous individual users--a different combination to each individual if desired. UNIX protection distinguishes only between three user or groups: the owner of the file, members of a single specified group, and everyone who can access the local file system.
- Enabling users to define their own groups of users, recorded in the *Protection Database* maintained by the Protection Server. The groups then appear on directories' access control lists as though they were individuals, which enables the granting of permissions to many users simultaneously.
- Enabling system administrators to create groups containing client machine IP addresses to permit access when it originates from the specified client machines. These types of groups are useful when it is necessary to adhere to machine-based licensing restrictions.

The Protection Server's main duty is to help the File Server determine if a user is authorized to access a file in the requested manner. The Protection Server creates a list of all the groups to which the user belongs. The File Server then compares this list to the ACL associated with the file's parent directory. A user thus acquires access both as an individual and as a member of any groups.

The Protection Server also maps usernames (the name typed at the login prompt) to *AFS user ID* numbers (*AFS UIDs*). These UIDs are functionally equivalent to UNIX UIDs, but operate in the domain of AFS rather than in the UNIX file system on a machine's local disk. This conversion service is essential because the tokens that the Authentication Server grants to authenticated users are stamped with usernames (to comply with Kerberos standards). The AFS server processes identify users by AFS UID, not by username. Before they can understand whom the token represents, they need the Protection Server to translate the username into an AFS UID. For further discussion of tokens, see "A More Detailed Look at Mutual Authentication" on page 51.

The Volume Server

The *Volume Server* provides the interface through which you create, delete, move, and replicate volumes, as well as prepare them for archiving to tape or other media (backing up). "Volumes" on page 6 explained the advantages gained by storing files in volumes. Creating and deleting volumes are necessary when adding and removing users from the system; volume moves are done for load balancing; and replication enables volume placement on multiple file server machines (for more on replication, see "Replication" on page 7).

The Volume Location (VL) Server

The *VL Server* maintains a complete list of volume locations in the *Volume Location Database (VLDB)*. When the Cache Manager (see "The Cache Manager" on page 13) begins to fill a file request from an application program, it first contacts the VL Server in order to learn which file server machine currently houses the volume containing the file. The Cache Manager then requests the file from the File Server process running on that file server machine.

The VLDB and VL Server make it possible for AFS to take advantage of the increased system availability gained by using multiple file server machines, because the Cache Manager knows where to find a particular file. Indeed, in a certain sense the VL Server is the keystone of the entire file system--when the information in the VLDB is inaccessible, the Cache Manager cannot retrieve files, even if the File Server processes are working properly. A list of the information stored in the VLDB about each volume is provided in "Volume Information in the VLDB" on page 133.

The Update Server

The *Update Server* helps guarantee that all file server machines are running the same version of a server process. System performance can be inconsistent if some machines are running one version of the BOS Server (for example) and other machines were running another version.

To ensure that all machines run the same version of a process, install new software on a single file server machine of each system type, called the *binary distribution machine* for that type. The binary distribution machine runs the server portion of the Update Server, whereas all the other machines of that type run the client portion of the Update Server. The client portions check frequently with the *server portion* to see if they are running the right version of every process; if not, the *client portion* retrieves the right version from the binary distribution machine and installs it locally. The system administrator does not need to remember to install new software individually on all the file server machines: the Update Server does it automatically. For more on binary distribution machines, see "Binary Distribution Machines" on page 70.

In cells that run the United States edition of AFS, the Update Server also distributes configuration files that all file server machines need to store on their local disks (for a description of the contents and purpose of these files, see "Common Configuration Files in the /usr/afs/etc Directory" on page 62). As with server process software, the need for consistent system performance demands that all the machines have the same version of these files. With the United States edition, the system administrator needs to make changes to these files on one machine only, the cell's *system control machine*, which runs a server portion of the Update Server. All other machines in the cell run a client portion that accesses the correct versions of these configuration files from the system control machine. Cells running the international edition of AFS do not use a system control machine to distribute configuration files. For more information, see "The System Control Machine" on page 70.

The Backup Server

The *Backup Server* maintains the information in the *Backup Database*. The Backup Server and the Backup Database enable administrators to back up data from AFS volumes to tape and restore it from tape to the file system if necessary. The server and database together are referred to as the Backup System.

Administrators initially configure the Backup System by defining sets of volumes to be dumped together and the schedule by which the sets are to be dumped. They also install the system's tape drives and define the drives' *Tape Coordinators*, which are the processes that control the tape drives.

Once the Backup System is configured, user and system data can be dumped from volumes to tape. In the event that data is ever lost from the system (for example, if a system or disk failure causes data to be lost), administrators can restore the data from tape. If tapes are periodically archived, or saved, data can also be restored to its state at a specific time. Additionally, because Backup System data is difficult to reproduce, the Backup Database itself can be backed up to tape and restored if it ever becomes corrupted. For more information on configuring and using the Backup System, see "Configuring the AFS Backup System" on page 195 and "Backing Up and Restoring AFS Data" on page 241.

The Salvager

The *Salvager* differs from other AFS Servers in that it runs only at selected times. The BOS Server invokes the Salvager when the File Server, Volume Server, or both fail. The Salvager attempts to repair disk corruption that can result from a failure.

As a system administrator, you can also invoke the Salvager as necessary, even if the File Server or Volume Server has not failed. See "Salvaging Volumes" on page 172.

The Network Time Protocol Daemon

The *Network Time Protocol Daemon (NTPD)* is not an AFS server process per se, but plays an important role. It helps guarantee that all of the file server machines agree on the time. The NTPD on one file server machine acts as a synchronization site, generally learning the correct time from a source outside the cell. The NTPDs on the other file server machines refer to the synchronization site to set the internal clocks on their machines.

Keeping clocks synchronized is particularly important to the correct operation of AFS's distributed database technology, which coordinates the copies of the Authentication, Backup, Protection, and Volume Location Databases; see "Replicating the AFS Administrative Databases" on page 31. Client machines also refer to these clocks for the correct time; therefore, it is less confusing if all file server machines have the same time. For more technical detail about the NTPD, see "The runntp Process" on page 109.

The Cache Manager

As already mentioned in "Caching and Callbacks" on page 7, the *Cache Manager* is the one component in this section that resides on client machines rather than on file server machines. It is not technically a stand-alone process, but rather a set of extensions or modifications in the client machine's kernel that enable communication with the server processes running on server machines. Its main duty is to translate file requests (made by application programs on client machines) into *remote procedure calls (RPCs)* to the File Server. (The Cache Manager first contacts the VL Server to find out which File Server currently houses the volume that contains a requested file, as mentioned in "The Volume Location (VL) Server" on page 12). When the Cache Manager receives the requested file, it caches it before passing data on to the application program.

The Cache Manager also tracks the state of files in its cache compared to the version at the File Server by storing the callbacks sent by the File Server. When the File Server breaks a callback, indicating that a file or volume changed, the Cache Manager requests a copy of the new version before providing more data to application programs.

Chapter 2. Issues in Cell Configuration and Administration

This chapter discusses many of the issues to consider when configuring and administering a cell, and directs you to detailed related information available elsewhere in this guide. It is assumed you are already familiar with the material in "An Overview of AFS Administration" on page 1.

It is best to read this chapter before installing your cell's first file server machine or performing any other administrative task.

Differences between AFS and UNIX: A Summary

AFS behaves like a standard UNIX file system in most respects, while also making file sharing easy within and between cells. This section describes some differences between AFS and the UNIX file system, referring you to more detailed information as appropriate.

Differences in File and Directory Protection

AFS augments the standard UNIX file protection mechanism in two ways: it associates an *access control list (ACL)* with each directory, and it enables users to define a large number of their own groups, which can be placed on ACLs.

AFS uses ACLs to protect files and directories, rather than relying exclusively on the mode bits. This has several implications, which are discussed further in the indicated sections:

- AFS ACLs use seven access permissions rather than the three UNIX mode bits. See "The AFS ACL Permissions" on page 515.
- For directories, AFS ignores the UNIX mode bits. For files, AFS uses only the first set of mode bits (the **owner** bits) , and their meaning interacts with permissions on the directory's ACL. See "How AFS Interprets the UNIX Mode Bits" on page 529.
- A directory's ACL protects all of the files in a directory in the same manner. To apply a more restrictive set of AFS permissions to certain file, place it in directory with a different ACL.
- Moving a file to a different directory changes its protection. See "Differences Between UFS and AFS Data Protection" on page 513.
- An ACL can include about 20 entries granting different combinations of permissions to different users or groups, rather than only the three UNIX entities represented by the three sets of mode bits. See "Differences Between UFS and AFS Data Protection" on page 513.
- You can designate an AFS file as write-only as in the UNIX file system, by setting only the **w** (**write**) mode bit. You cannot designate an AFS directory as write-only, because AFS ignores the mode bits on a directory. See "How AFS Interprets the UNIX Mode Bits" on page 529.

AFS enables users to define the groups of other users. Placing these groups on ACLs extends the same permissions to a number of exactly specified users at the same time, which is much more convenient than placing the individuals on the ACLs directly. See "Administering the Protection Database" on page 487.

There are also system-defined groups, **system:anyuser** and **system:authuser**, whose presence on an ACL extends access to a wide range of users at once. See "The System Groups" on page 488 and "Using Groups on ACLs" on page 518.

Differences in Authentication

Just as the AFS filespace is distinct from each machine's local file system, AFS authentication is separate from local login. This has two practical implications, which are discussed further in "Using an AFS-modified login Utility" on page 43.

- To access AFS files, users must both log into the local machine's UNIX file system and authenticate with the AFS authentication service. (Logging into the local UNIX file system is necessary because the AFS filespace is accessed through the Cache Manager, which resides in the local machine's kernel.)

  AFS provides a modified login utility for each system type that accomplishes both local login and AFS authentication in one step, based on a single password. If you choose not to use the AFS-modified login utility, your users must login and authenticate in separate steps, as detailed in the *IBM AFS User Guide*.

- Passwords are stored in two separate places: the Authentication Database for AFS and each machine's local password file (**/etc/passwd** or equivalent) for the UNIX file system. A user's passwords in the two places can differ if desired, though the resulting behavior depends on whether and how the cell is using an AFS-modified login utility.

Differences in the Semantics of Standard UNIX Commands

This section summarizes how AFS modifies the functionality of some UNIX commands.

The chmod command

Only members of the **system:administrators** group can use this command to turn on the setuid, setgid or sticky mode bits on AFS files. For more information, see "Determining if a Client Can Run Setuid Programs" on page 369.

The chown command

Only members of the **system:administrators** group can issue this command on AFS files.

The chgrp command

Only members of the **system:administrators** can issue this command on AFS files and directories.

The ftpd daemon

The AFS-modified version of this daemon attempts to authenticate remote issuers of the **ftp** command with the local AFS authentication service. See "Using UNIX Remote Services in the AFS Environment" on page 54.

The groups command

If the user’s AFS tokens are associated with a process authentication group (PAG), the output of this command sometimes includes two large numbers. To learn about PAGs, see "Identifying AFS Tokens by PAG" on page 42.

The inetd daemon

The AFS-modified version of this daemon authenticates remote issuers of the AFS-modified **rcp** and **rsh** commands with the local AFS authentication service. See "Using UNIX Remote Services in the AFS Environment" on page 54.

The login utility

AFS-modified login utilities both log the issuer into the local file system and authenticate the user with the AFS authentication service. See "Using an AFS-modified login Utility" on page 43.

The ln command

This command cannot create hard links between files in different AFS directories. See "Creating Hard Links" on page 18.

The rcp command

The AFS-modified version of this command enables the issuer to access files on the remote machine as an authenticated AFS user. See "Using UNIX Remote Services in the AFS Environment" on page 54.

The rlogind daemon

The AFS-modified version of this daemon authenticates remote issuers of the **rlogin** command with the local AFS authentication service. See "Using UNIX Remote Services in the AFS Environment" on page 54.

The AFS distribution for some system types possibly does not include a modified **rlogind** program. See the *IBM AFS Release Notes*.

The remsh or rsh command

The AFS-modified version of this command enables the issuer to execute commands on the remote machine as an authenticated AFS user. See "Using UNIX Remote Services in the AFS Environment" on page 54.

The AFS version of the fsck Command

Never run the standard UNIX **fsck** command on an AFS file server machine. It does not understand how the File Server organizes volume data on disk, and so moves all AFS data into the **lost+found** directory on the partition.

Instead, use the version of the **fsck** program that is included in the AFS distribution. The *IBM AFS Quick Beginnings* explains how to replace the vendor-supplied **fsck** program with the AFS version as you install each server machine.

The AFS version functions like the standard **fsck** program on data stored on both UFS and AFS partitions. The appearance of a banner like the following as the **fsck** program initializes confirms that you are running the correct one:

```
   --- AFS (R) version fsck---
```

where *version* is the AFS version. For correct results, it must match the AFS version of the server binaries in use on the machine.

If you ever accidentally run the standard version of the program, contact AFS Product Support immediately. It is sometimes possible to recover volume data from the **lost+found** directory.

Creating Hard Links

AFS does not allow hard links (created with the UNIX **ln** command) between files that reside in different directories, because in that case it is unclear which of the directory's ACLs to associate with the link.

AFS also does not allow hard links to directories, in order to keep the file system organized as a tree.

It is possible to create symbolic links (with the UNIX **ln -s** command) between elements in two different AFS directories, or even between an element in AFS and one in a machine's local UNIX file system. Do not create a symbolic link to a file whose name begins with either a number sign (**#**) or a percent sign (**%**), however. The Cache Manager interprets such links as a mount point to a regular or read/write volume, respectively.

AFS Implements Save on Close

When an application issues the UNIX **close** system call on a file, the Cache Manager performs a synchronous write of the data to the File Server that maintains the central copy of the file. It does not return control to the application until the File Server has acknowledged receipt of the data. For the **fsync** system call, control does not return to the application until the File Server indicates that it has written the data to non-volatile storage on the file server machine.

When an application issues the UNIX **write** system call, the Cache Manager writes modifications to the local AFS client cache only. If the local machine crashes or an application program exits without issuing the **close** system call, it is possible that the modifications are not recorded in the central copy of the file maintained by the File Server. The Cache Manager does sometimes write this type of modified data from the cache to the File Server without receiving the **close** or **fsync** system call, for example if it needs to free cache chunks for new data. However, it is not generally possible to predict when the Cache Manager transfers modified data to the File Server in this way.

The implication is that if an application's **Save** option invokes the **write** system call rather than **close** or **fsync**, the changes are not necessarily stored permanently on the File Server machine. Most application programs issue the **close** system call for save operations, as well as when they finish handling a file and when they exit.

Setuid Programs

Set the UNIX setuid bit only for the local superuser **root**; this does not present an automatic security risk: the local superuser has no special privilege in AFS, but only in the local machine's UNIX file system and kernel.

Any file can be marked with the setuid bit, but only members of the **system:administrators** group can issue the **chown** system call or the **/etc/chown** command.

The **fs setcell** command determines whether setuid programs that originate in a foreign cell can run on a given client machine. See "Determining if a Client Can Run Setuid Programs" on page 369.

Choosing a Cell Name

This section explains how to choose a cell name and explains why choosing an appropriate cell name is important.

Your cell name must distinguish your cell from all others in the AFS global namespace. By conventions, the cell name is the second element in any AFS pathname; therefore, a unique cell name guarantees that every AFS pathname uniquely identifies a file, even if cells use the same directory names at lower levels in their local AFS filespace. For example, both the ABC Corporation cell and the State University cell can have a home directory for the user **pat**, because the pathnames are distinct: **/afs/abc.com/usr/pat** and **/afs/stateu.edu/usr/pat**.

By convention, cell names follow the ARPA Internet Domain System conventions for site names. If you are already an Internet site, then it is simplest to choose your Internet domain name as the cellname.

If you are not an Internet site, it is best to choose a unique Internet-style name, particularly if you plan to connect to the Internet in the future. AFS Product Support is available for help in selecting an appropriate name. There are a few constraints on AFS cell names:

- It can contain as many as 64 characters, but shorter names are better because the cell name frequently is part of machine and file names. If your cell name is long, you can reduce pathname length by creating a symbolic link to the complete cell name, at the second level in your file tree. See "The Second (Cellname) Level" on page 24.
- To guarantee it is suitable for different operating system types, the cell name can contain only lowercase characters, numbers, underscores, dashes, and periods. Do not include command shell metacharacters.
- It can include any number of fields, which are conventionally separated by periods (see the examples below).
- It must end in a suffix that indicates the type of institution it is, or the country in which it is situated. The following are some of the standard suffixes:

  .com

  For businesses and other commercial organizations. Example: **abc.com** for the ABC Corporation cell.

.edu

For educational institutions such as universities. Example: **stateu.edu** for the State University cell.

.gov

For United States government institutions.

.mil

For United States military installations.

Other suffixes are available if none of these are appropriate. You can learn about suffixes by calling the Defense Data Network [Internet] Network Information Center in the United States at (800) 235-3155. The NIC can also provide you with the forms necessary for registering your cell name as an Internet domain name. Registering your name prevents another Internet site from adopting the name later.

How to Set the Cell Name

The cell name is recorded in two files on the local disk of each file server and client machine. Among other functions, these files define the machine's cell membership and so affect how programs and processes run on the machine; see "Why Choosing the Appropriate Cell Name is Important" on page 21. The procedure for setting the cell name is different for the two types of machines.

For file server machines, the two files that record the cell name are the **/usr/afs/etc/ThisCell** and **/usr/afs/etc/CellServDB** files. As described more explicitly in the *IBM AFS Quick Beginnings*, you set the cell name in both by issuing the **bos setcellname** command on the first file server machine you install in your cell. It is not usually necessary to issue the command again. If you run the United States edition of AFS and use the Update Server, it distributes its copy of the **ThisCell** and **CellServDB** files to additional server machines that you install. If you use the international edition of AFS, the *IBM AFS Quick Beginnings* explains how to copy the files manually.

For client machines, the two files that record the cell name are the **/usr/vice/etc/ThisCell** and **/usr/vice/etc/CellServDB** files. You create these files on a per-client basis, either with a text editor or by copying them onto the machine from a central source in AFS. See "Maintaining Knowledge of Database Server Machines" on page 364 for details.

Change the cell name in these files only when you want to transfer the machine to a different cell (it can only belong to one cell at a time). If the machine is a file server, follow the complete set of instructions in the *IBM AFS Quick Beginnings* for configuring a new cell. If the machine is a client, all you need to do is change the files appropriately and reboot the machine. The next section explains further the negative consequences of changing the name of an existing cell.

To set the default cell name used by most AFS commands without changing the local **/usr/vice/etc/ThisCell** file, set the AFSCELL environment variable in the command shell. It is worth setting this variable if you need to complete significant administrative work in a foreign cell.

Note: The **fs checkservers** and **fs mkmount** commands do not use the AFSCELL variable. The **fs checkservers** command always defaults to the cell named in the **ThisCell** file, unless the **-cell**

argument is used. The **fs mkmount** command defaults to the cell in which the parent directory of the new mount point resides.

Why Choosing the Appropriate Cell Name is Important

Take care to select a cell name that is suitable for long-term use. Changing a cell name later is complicated. An appropriate cell name is important because it is the second element in the pathname of all files in a cell's file tree. Because each cell name is unique, its presence in an AFS pathname makes the pathname unique in the AFS global namespace, even if multiple cells use similar filespace organization at lower levels. For instance, it means that every cell can have a home directory called **/afs/**`cellname`**/usr/pat** without causing a conflict. The presence of the cell name in pathnames also means that users in every cell use the same pathname to access a file, whether the file resides in their local cell or in a foreign cell.

Another reason to choose the correct cell name early in the process of installing your cell is that the cell membership defined in each machine's **ThisCell** file affects the performance of many programs and processes running on the machine. For instance, AFS commands (**fs**, **kas**, **pts** and **vos** commands) by default execute in the cell of the machine on which they are issued. The command interpreters check the **ThisCell** file on the local disk and then contact the database server machines listed in the **CellServDB** file for the indicated cell (the **bos** commands work differently because the issuer always has to name of the machine on which to run the command).

The **ThisCell** file also determines the cell for which a user receives an AFS token when he or she logs in to a machine. The cell name also plays a role in security. As it converts a user password into an encryption key for storage in the Authentication Database, the Authentication Server combines the password with the cell name found in the **ThisCell** file. AFS-modified login utilities use the same algorithm to convert the user's password into an encryption key before contacting the Authentication Server to obtain a token for the user. (For a description of how AFS's security system uses encryption keys, see "A More Detailed Look at Mutual Authentication" on page 51.)

This method of converting passwords into encryption keys means that the same password results in different keys in different cells. Even if a user uses the same password in multiple cells, obtaining a user's token from one cell does not enable unauthorized access to the user's account in another cell.

If you change the cell name, you must change the **ThisCell** and **CellServDB** files on every server and client machine. Failure to change them all can prevent login, because the encryption keys produced by the login utility do not match the keys stored in the Authentication Database. In addition, many commands from the AFS suites do not work as expected.

Participating in the AFS Global Namespace

Participating in the AFS global namespace makes your cell's local file tree visible to AFS users in foreign cells and makes other cells' file trees visible to your local users. It makes file sharing across cells just as easy as sharing within a cell. This section outlines the procedures necessary for participating in the global namespace.

- Participation in the global namespace is not mandatory. Some cells use AFS primarily to facilitate file sharing within the cell, and are not interested in providing their users with access to foreign cells.
- Making your file tree visible does not mean making it vulnerable. You control how foreign users access your cell using the same protection mechanisms that control local users' access. See "Granting and Denying Foreign Users Access to Your Cell" on page 24.
- The two aspects of participation are independent. A cell can make its file tree visible without allowing its users to see foreign cells' file trees, or can enable its users to see other file trees without advertising its own.
- You make your cell visible to others by advertising your database server machines. See "Making Your Cell Visible to Others" on page 22.
- You control access to foreign cells on a per-client machine basis. In other words, it is possible to make a foreign cell accessible from one client machine in your cell but not another. See "Making Other Cells Visible in Your Cell" on page 23.

What the Global Namespace Looks Like

The AFS global namespace appears the same to all AFS cells that participate in it, because they all agree to follow a small set of conventions in constructing pathnames.

The first convention is that all AFS pathnames begin with the string **/afs** to indicate that they belong to the AFS global namespace.

The second convention is that the cell name is the second element in an AFS pathname; it indicates where the file resides (that is, the cell in which a file server machine houses the file). As noted, the presence of a cell name in pathnames makes the global namespace possible, because it guarantees that all AFS pathnames are unique even if cells use the same directory names at lower levels in their AFS filespace.

What appears at the third and lower levels in an AFS pathname depends on how a cell has chosen to arrange its filespace. There are some suggested conventional directories at the third level; see "The Third Level" on page 25.

Making Your Cell Visible to Others

You make your cell visible to others by advertising your cell name and database server machines. Just like client machines in the local cell, the Cache Manager on machines in foreign cells use the information to reach your cell's Volume Location (VL) Servers when they need volume and file location information. Similarly, client-side authentication programs running in foreign cells use the information to contact your cell's authentication service.

There are two places you can make this information available:

- In the global **CellServDB** file maintained by the AFS Product Support group. This file lists the name and database server machines of every cell that has agreed to make this information available to other cells.

To add or change your cell's listing in this file, have the official support contact at your site call or write to AFS Product Support. Changes to the file are frequent enough that AFS Product Support does not announce each one. It is a good policy to check the file for changes on a regular schedule.

- A file called **CellServDB.local** in the **/afs/**`cellname`**/service/etc** directory of your cell's filespace. List only your cell's database server machines.

Update the files whenever you change the identity of your cell's database server machines. Also update the copies of the **CellServDB** files on all of your server machines (in the **/usr/afs/etc** directory) and client machines (in the **/usr/vice/etc** directory). For instructions, see "Maintaining the Server CellServDB File" on page 88 and "Maintaining Knowledge of Database Server Machines" on page 364.

Once you have advertised your database server machines, it can be difficult to make your cell invisible again. You can remove the **CellServDB.local** file and ask AFS Product Support to remove your entry from the global **CellServDB** file, but other cells probably have an entry for your cell in their local **CellServDB** files already. To make those entries invalid, you must change the names or IP addresses of your database server machines.

Your cell does not have to be invisible to be inaccessible, however. To make your cell completely inaccessible to foreign users, remove the **system:anyuser** group from all ACLs at the top three levels of your filespace; see "Granting and Denying Foreign Users Access to Your Cell" on page 24.

Making Other Cells Visible in Your Cell

To make a foreign cell's filespace visible on a client machine in your cell, perform the following three steps:

1. Mount the cell's **root.cell** volume at the second level in your cell's filespace just below the **/afs** directory. Use the **fs mkmount** command with the **-cell** argument as instructed in "To create a cellular mount point" on page 154.
2. Mount AFS at the **/afs** directory on the client machine. The **afsd** program, which initializes the Cache Manager, performs the mount automatically at the directory named in the first field of the local **/usr/vice/etc/cacheinfo** file or by the command's **-mountdir** argument. Mounting AFS at an alternate location makes it impossible to reach the filespace of any cell that mounts its **root.afs** and **root.cell** volumes at the conventional locations. See "Displaying and Setting the Cache Size and Location" on page 357.
3. Create an entry for the cell in the list of database server machines which the Cache Manager maintains in kernel memory.

   The **/usr/vice/etc/CellServDB** file on every client machine's local disk lists the database server machines for the local and foreign cells. The **afsd** program reads the contents of the **CellServDB** file into kernel memory as it initializes the Cache Manager. You can also use the **fs newcell** command to add or alter entries in kernel memory directly between reboots of the machine. See "Maintaining Knowledge of Database Server Machines" on page 364.

Note that making a foreign cell visible to client machines does not guarantee that your users can access its filespace. The ACLs in the foreign cell must also grant them the necessary permissions.

Granting and Denying Foreign Users Access to Your Cell

Making your cell visible in the AFS global namespace does not take away your control over the way in which users from foreign cells access your file tree.

By default, foreign users access your cell as the user **anonymous**, which means they have only the permissions granted to the **system:anyuser** group on each directory's ACL. Normally these permissions are limited to the **l** (**lookup**) and **r** (**read**) permissions.

There are two ways to grant wider access to foreign users:

- Grant additional permissions to the **system:anyuser** group on certain ACLs. Keep in mind, however, that all users can then access that directory in the indicated way (not just specific foreign users you have in mind).
- Create a local authentication account for specific foreign users, by creating entries in the Protection and Authentication Databases and local password file. It is not possible to place foreign usernames on ACLs, nor to authenticate in a foreign cell without having an account in it.

Configuring Your AFS Filespace

This section summarizes the issues to consider when configuring your AFS filespace. For a discussion of creating volumes that correspond most efficiently to the filespace's directory structure, see "Creating Volumes to Simplify Administration" on page 26.

Note: For Windows users: Windows uses a backslash (\) rather than a forward slash (/) to separate the elements in a pathname. The hierarchical organization of the filespace is however the same as on a UNIX machine.

AFS pathnames must follow a few conventions so the AFS global namespace looks the same from any AFS client machine. There are corresponding conventions to follow in building your file tree, not just because pathnames reflect the structure of a file tree, but also because the AFS Cache Manager expects a certain configuration.

The Top /afs Level

The first convention is that the top level in your file tree be called the **/afs** directory. If you name it something else, then you must use the **-mountdir** argument with the **afsd** program to get Cache Managers to mount AFS properly. You cannot participate in the AFS global namespace in that case.

The Second (Cellname) Level

The second convention is that just below the **/afs** directory you place directories corresponding to each cell whose file tree is visible and accessible from the local cell. Minimally, there must be a directory for the local cell. Each such directory is a mount point to the indicated cell's **root.cell** volume. For example, in the ABC Corporation cell, **/afs/abc.com** is a mount point for the cell's own **root.cell** volume and **stateu.edu** is a mount point for the State University cell's **root.cell** volume. The **fs lsmount** command displays the mount points.

```
% fs lsmount /afs/abc.com
'/afs/abc.com' is a mount point for volume '#root.cell'
% fs lsmount /afs/stateu.edu
'/afs/stateu.edu' is a mount point for volume '#stateu.edu:root.cell'
```

To reduce the amount of typing necessary in pathnames, you can create a symbolic link with an abbreviated name to the mount point of each cell your users frequently access (particularly the home cell). In the ABC Corporation cell, for instance, **/afs/abc** is a symbolic link to the **/afs/abc.com** mount point, as the **fs lsmount** command reveals.

```
% fs lsmount /afs/abc
'/afs/abc' is a symbolic link, leading to a mount point for volume '#root.cell'
```

The Third Level

You can organize the third level of your cell's file tree any way you wish. The following list describes directories that appear at this level in the conventional configuration:

common

This directory contains programs and files needed by users working on machines of all system types, such as text editors, online documentation files, and so on. Its **/etc** subdirectory is a logical place to keep the central update sources for files used on all of your cell's client machines, such as the **ThisCell** and **CellServDB** files.

public

A directory accessible to anyone who can access your filespace, because its ACL grants the **l** (**lookup**) and **r** (**read**) permissions to the **system:anyuser** group. It is useful if you want to enable your users to make selected information available to everyone, but do not want to grant foreign users access to the contents of the **usr** directory which houses user home directories (and is also at this level). It is conventional to create a subdirectory for each of your cell's users.

service

This directory contains files and subdirectories that help cells coordinate resource sharing. For a list of the proposed standard files and subdirectories to create, call or write to AFS Product Support.

As an example, files that other cells expect to find in this directory's **etc** subdirectory can include the following:

- **CellServDB.export**, a list of database server machines for many cells

- **CellServDB.local**, a list of the cell's own database server machines
- **passwd**, a copy of the local password file (**/etc/passwd** or equivalent) kept on the local disk of the cell's client machines
- **group**, a copy of the local groups file (**/etc/group** or equivalent) kept on the local disk of the cell's client machines

sys_type

A separate directory for storing the server and client binaries for each system type you use in the cell. Configuration is simplest if you use the system type names assigned in the AFS distribution, particularly if you wish to use the **@sys** variable in pathnames (see "Using the @sys Variable in Pathnames" on page 34). The *IBM AFS Release Notes* lists the conventional name for each supported system type.

Within each such directory, create directories named **bin**, **etc**, **usr**, and so on, to store the programs normally kept in the **/bin**, **/etc** and **/usr** directories on a local disk. Then create symbolic links from the local directories on client machines into AFS; see "Configuring the Local Disk" on page 33. Even if you do not choose to use symbolic links in this way, it can be convenient to have central copies of system binaries in AFS. If binaries are accidentally removed from a machine, you can recopy them onto the local disk from AFS rather than having to recover them from tape

usr

This directory contains home directories for your local users. As discussed in the previous entry for the **public** directory, it is often practical to protect this directory so that only locally authenticated users can access it. This keeps the contents of your user's home directories as secure as possible.

If your cell is quite large, directory lookup can be slowed if you put all home directories in a single **usr** directory. For suggestions on distributing user home directories among multiple grouping directories, see "Grouping Home Directories" on page 38.

wsadmin

This directory contains prototype, configuration and library files for use with the **package** program. See "Configuring Client Machines with the package Program" on page 389.

Creating Volumes to Simplify Administration

This section discusses how to create volumes in ways that make administering your system easier.

At the top levels of your file tree (at least through the third level), each directory generally corresponds to a separate volume. Some cells also configure the subdirectories of some third level directories as separate volumes. Common examples are the **/afs/**`cellname`**/common** and **/afs/**`cellname`**/usr** directories.

You do not have to create a separate volume for every directory level in a tree, but the advantage is that each volume tends to be smaller and easier to move for load balancing. The overhead for a mount point is no greater than for a standard directory, nor does the volume structure itself require much disk space. Most cells find that below the fourth level in the tree, using a separate volume for each directory is no longer efficient. For instance, while each user's home directory (at the fourth level in the tree) corresponds to a separate volume, all of the subdirectories in the home directory normally reside in the same volume.

Keep in mind that only one volume can be mounted at a given directory location in the tree. In contrast, a volume can be mounted at several locations, though this is not recommended because it distorts the hierarchical nature of the file tree, potentially causing confusion.

Assigning Volume Names

You can name your volumes anything you choose, subject to a few restrictions:

- Read/write volume names can be up to 22 characters in length. The maximum length for volume names is 31 characters, and there must be room to add the **.readonly** extension on read-only volumes.
- Do not add the **.readonly** and **.backup** extensions to volume names yourself, even if they are appropriate. The Volume Server adds them automatically as it creates a read-only or backup version of a volume.
- There must be volumes named **root.afs** and **root.cell**, mounted respectively at the top (**/afs**) level in the filespace and just below that level, at the cell's name (for example, at **/afs/abc.com** in the ABC Corporation cell).

  Deviating from these names only creates confusion and extra work. Changing the name of the **root.afs** volume, for instance, means that you must use the **-rootvol** argument to the **afsd** program on every client machine, to name the alternate volume.

  Similarly, changing the **root.cell** volume name prevents users in foreign cells from accessing your filespace, if the mount point for your cell in their filespace refers to the conventional **root.cell** name. Of course, this is one way to make your cell invisible to other cells.

It is best to assign volume names that indicate the type of data they contain, and to use similar names for volumes with similar contents. It is also helpful if the volume name is similar to (or at least has elements in common with) the name of the directory at which it is mounted. Understanding the pattern then enables you accurately to guess what a volume contains and where it is mounted.

Many cells find that the most effective volume naming scheme puts a common prefix on the names of all related volumes. "Table 1" on page 27 describes the recommended prefixing scheme.

Prefix	Contents	Example Name	Example Mount Point
common.	popular programs and files	**common.etc**	**/afs/**`cellname`**/common/etc**
src.	source code	**src.afs**	**/afs/**`cellname`**/src/afs**
proj.	project data	**proj.portafs**	**/afs/**`cellname`**/proj/portafs**

Prefix	Contents	Example Name	Example Mount Point
test.	testing or other temporary data	**test.smith**	**/afs/**`cellname`**/usr/smith/test**
user.	user home directory data	**user.terry**	**/afs/**`cellname`**/usr/terry**
sys_type**.**	programs compiled for an operating system type	**rs_aix42.bin**	**/afs/**`cellname`**/rs_aix42/bin**

Table 1. Suggested volume prefixes

"Table 2" on page 28 is a more specific example for a cell's **rs_aix42** system volumes and directories:

Example Name	Example Mount Point
rs_aix42.bin	**/afs/**`cellname`**/rs_aix42/bin**, **/afs/**`cellname`**/rs_aix42/bin**
rs_aix42.etc	**/afs/**`cellname`**/rs_aix42/etc**
rs_aix42.usr	**/afs/**`cellname`**/rs_aix42/usr**
rs_aix42.usr.afsws	**/afs/**`cellname`**/rs_aix42/usr/afsws**
rs_aix42.usr.lib	**/afs/**`cellname`**/rs_aix42/usr/lib**
rs_aix42.usr.bin	**/afs/**`cellname`**/rs_aix42/usr/bin**
rs_aix42.usr.etc	**/afs/**`cellname`**/rs_aix42/usr/etc**
rs_aix42.usr.inc	**/afs/**`cellname`**/rs_aix42/usr/inc**
rs_aix42.usr.man	**/afs/**`cellname`**/rs_aix42/usr/man**
rs_aix42.usr.sys	**/afs/**`cellname`**/rs_aix42/usr/sys**
rs_aix42.usr.local	**/afs/**`cellname`**/rs_aix42/usr/local**

Table 2. Example volume-prefixing scheme

There are several advantages to this scheme:

- The volume name is similar to the mount point name in the filespace. In all of the entries in "Table 2" on page 28, for example, the only difference between the volume and mount point name is that the former uses periods as separators and the latter uses slashes. Another advantage is that the volume name indicates the contents, or at least suggests the directory on which to issue the **ls** command to learn the contents.
- It makes it easy to manipulate groups of related volumes at one time. In particular, the **vos backupsys** command's **-prefix** argument enables you to create a backup version of every volume whose name starts with the same string of characters. Making a backup version of each volume is one of the first steps in backing up a volume with the AFS Backup System, and doing it for many volumes with one command saves you a good deal of typing. For instructions for creating backup volumes, see "Creating Backup Volumes" on page 144, For information on the AFS Backup System, see "Configuring the AFS Backup System" on page 195 and "Backing Up and Restoring AFS Data" on page 241.
- It makes it easy to group related volumes together on a partition. Grouping related volumes together has several advantages of its own, discussed in "Grouping Related Volumes on a Partition" on page 28.

Grouping Related Volumes on a Partition

If your cell is large enough to make it practical, consider grouping related volumes together on a partition. In general, you need at least three file server machines for volume grouping to be effective. Grouping has several advantages, which are most obvious when the file server machine becomes inaccessible:

- If you keep a hardcopy record of the volumes on a partition, you know which volumes are unavailable. You can keep such a record without grouping related volumes, but a list composed of unrelated volumes is much harder to maintain. Note that the record must be on paper, because the outage can prevent you from accessing an online copy or from issuing the **vos listvol** command, which gives you the same information.
- The effect of an outage is more localized. For example, if all of the binaries for a given system type are on one partition, then only users of that system type are affected. If a partition houses binary volumes from several system types, then an outage can affect more people, particularly if the binaries that remain available are interdependent with those that are not available.

The advantages of grouping related volumes on a partition do not necessarily extend to the grouping of all related volumes on one file server machine. For instance, it is probably unwise in a cell with two file server machines to put all system volumes on one machine and all user volumes on the other. An outage of either machine probably affects everyone.

Admittedly, the need to move volumes for load balancing purposes can limit the practicality of grouping related volumes. You need to weigh the complementary advantages case by case.

When to Replicate Volumes

As discussed in "Replication" on page 7, replication refers to making a copy, or clone, of a read/write source volume and then placing the copy on one or more additional file server machines. Replicating a volume can increase the availability of the contents. If one file server machine housing the volume becomes inaccessible, users can still access the copy of the volume stored on a different machine. No one machine is likely to become overburdened with requests for a popular file, either, because the file is available from several machines.

However, replication is not appropriate for all cells. If a cell does not have much disk space, replication can be unduly expensive, because each clone not on the same partition as the read/write source takes up as much disk space as its source volume did at the time the clone was made. Also, if you have only one file server machine, replication uses up disk space without increasing availability.

Replication is also not appropriate for volumes that change frequently. You must issue the **vos release** command every time you need to update a read-only volume to reflect changes in its read/write source.

For both of these reasons, replication is appropriate only for popular volumes whose contents do not change very often, such as system binaries and other volumes mounted at the upper levels of your filespace. User volumes usually exist only in a read/write version since they change so often.

If you are replicating any volumes, you must replicate the **root.afs** and **root.cell** volumes, preferably at two or three sites each (even if your cell only has two or three file server machines). The Cache Manager needs to pass through the directories corresponding to the **root.afs** and **root.cell** volumes as it interprets any pathname. The unavailability of these volumes makes all other volumes unavailable too, even if the file server machines storing the other volumes are still functioning.

Another reason to replicate the **root.afs** volume is that it can lessen the load on the File Server machine. The Cache Manager has a bias to access a read-only version of the **root.afs** volume if it is replicate, which puts the Cache Manager onto the *read-only path* through the AFS filespace. While on the read-only path, the Cache Manager attempts to access a read-only copy of replicated volumes. The File Server needs to track only one callback per Cache Manager for all of the data in a read-only volume, rather than the one callback per file it must track for read/write volumes. Fewer callbacks translate into a smaller load on the File Server.

If the **root.afs** volume is not replicated, the Cache Manager follows a read/write path through the filespace, accessing the read/write version of each volume. The File Server distributes and tracks a separate callback for each file in a read/write volume, imposing a greater load on it.

For more on read/write and read-only paths, see "The Rules of Mount Point Traversal" on page 149.

It also makes sense to replicate system binary volumes in many cases, as well as the volume corresponding to the **/afs/**`cellname`**/usr** directory and the volumes corresponding to the **/afs/**`cellname`**/common** directory and its subdirectories.

It is a good idea to place a replica on the same partition as the read/write source. In this case, the read-only volume is a clone (like a backup volume): it is a copy of the source volume's vnode index, rather than a full copy of the volume contents. Only if the read/write volume moves to another partition or changes substantially does the read-only volume consume significant disk space. Read-only volumes kept on other partitions always consume the full amount of disk space that the read/write source consumed when the read-only volume was created.

The Default Quota and ACL on a New Volume

Every AFS volume has associated with it a quota that limits the amount of disk space the volume is allowed to use. To set and change quota, use the commands described in "Setting and Displaying Volume Quota and Current Size" on page 176.

By default, every new volume is assigned a space quota of 5000 KB blocks unless you include the **-maxquota** argument to the **vos create** command. Also by default, the ACL on the root directory of every new volume grants all permissions to the members of the **system:administrators** group. To learn how to change these values when creating an account with individual commands, see "To create one user account with individual commands" on page 464. When using **uss** commands to create accounts, you can specify alternate ACL and quota values in the template file's **V** instruction; see "Creating a Volume with the V Instruction" on page 429.

Configuring Server Machines

This section discusses some issues to consider when configuring server machines, which store AFS data, transfer it to client machines on request, and house the AFS administrative databases. To learn about client machines, see "Configuring Client Machines" on page 33.

If your cell has more than one AFS server machine, you can configure them to perform specialized functions. A machine can assume one or more of the roles described in the following list. For more details, see "The Four Roles for File Server Machines" on page 68.

- A *simple file server* machine runs only the processes that store and deliver AFS files to client machines. You can run as many simple file server machines as you need to satisfy your cell's performance and disk space requirements.
- A *database server machine* runs the four database server processes that maintain AFS's replicated administrative databases: the Authentication, Backup, Protection, and Volume Location (VL) Server processes.
- A *binary distribution machine* distributes the AFS server binaries for its system type to all other server machines of that system type.
- The single *system control machine* distributes common server configuration files to all other server machines in the cell, in a cell that runs the United States edition of AFS (cells that use the international edition of AFS must not use the system control machine for this purpose). The machine conventionally also serves as the time synchronization source for the cell, adjusting its clock according to a time source outside the cell.

The *IBM AFS Quick Beginnings* explains how to configure your cell's first file server machine to assume all four roles. The *IBM AFS Quick Beginnings* chapter on installing additional server machines also explains how to configure them to perform one or more roles.

Replicating the AFS Administrative Databases

The AFS administrative databases are housed on database server machines and store information that is crucial for correct cell functioning. Both server processes and Cache Managers access the information frequently:

- Every time a Cache Manager fetches a file from a directory that it has not previously accessed, it must look up the file's location in the Volume Location Database (VLDB).
- Every time a user obtains an AFS token from the Authentication Server, the server looks up the user's password in the Authentication Database.
- The first time that a user accesses a volume housed on a specific file server machine, the File Server contacts the Protection Server for a list of the user's group memberships as recorded in the Protection Database.
- Every time you back up a volume using the AFS Backup System, the Backup Server creates records for it in the Backup Database.

Maintaining your cell is simplest if the first machine has the lowest IP address of any machine you plan to use as a database server machine. If you later decide to use a machine with a lower IP address as a database server machine, you must update the **CellServDB** file on all clients before introducing the new machine.

If your cell has more than one server machine, it is best to run more than one as a database server machine (but more than three are rarely necessary). Replicating the administrative databases in this way yields the same benefits as replicating volumes: increased availability and reliability. If one database server machine or process stops functioning, the information in the database is still available from others. The load of requests for database information is spread across multiple machines, preventing any one from becoming overloaded.

Unlike replicated volumes, however, replicated databases do change frequently. Consistent system performance demands that all copies of the database always be identical, so it is not acceptable to record changes in only some of them. To synchronize the copies of a database, the database server processes use AFS's distributed database technology, Ubik. See "Replicating the AFS Administrative Databases" on page 74.

If your cell has only one file server machine, it must also serve as a database server machine. If you cell has two file server machines, it is not always advantageous to run both as database server machines. If a server, process, or network failure interrupts communications between the database server processes on the two machines, it can become impossible to update the information in the database because neither of them can alone elect itself as the synchronization site.

AFS Files on the Local Disk

It is generally simplest to store the binaries for all AFS server processes in the **/usr/afs/bin** directory on every file server machine, even if some processes do not actively run on the machine. This makes it easier to reconfigure a machine to fill a new role.

For security reasons, the **/usr/afs** directory on a file server machine and all of its subdirectories and files must be owned by the local superuser **root** and have only the first **w** (**write**) mode bit turned on. Some files even have only the first **r** (**read**) mode bit turned on (for example, the **/usr/afs/etc/KeyFile** file, which lists the AFS server encryption keys). Each time the BOS Server starts, it checks that the mode bits on certain files and directories match the expected values. For a list, see the *IBM AFS Quick Beginnings* section about protecting sensitive AFS directories, or the discussion of the output from the **bos status** command in "To display the status of server processes and their BosConfig entries" on page 113.

For a description of the contents of all AFS directories on a file server machine's local disk, see "Administering Server Machines" on page 59.

Configuring Partitions to Store AFS Data

The partitions that house AFS volumes on a file server machine must be mounted at directories named

/vicep*index*

where *index* is one or two lowercase letters. By convention, the first AFS partition created is mounted at the **/vicepa** directory, the second at the **/vicepb** directory, and so on through the **/vicepz** directory. The names then continue with **/vicepaa** through **/vicepaz**, **/vicepba** through **/vicepbz**, and so on, up to the maximum supported number of server partitions, which is specified in the IBM AFS Release Notes.

Each **/vicep**x directory must correspond to an entire partition or logical volume, and must be a subdirectory of the root directory (/). It is not acceptable to configure part of (for example) the **/usr** partition as an AFS server partition and mount it on a directory called **/usr/vicepa**.

Also, do not store non-AFS files on AFS server partitions. The File Server and Volume Server expect to have available all of the space on the partition. Sharing space also creates competition between AFS and the local UNIX file system for access to the partition, particularly if the UNIX files are frequently used.

Monitoring, Rebooting and Automatic Process Restarts

AFS provides several tools for monitoring the File Server, including the **scout** and **afsmonitor** programs. You can configure them to alert you when certain threshold values are exceeded, for example when a server partition is more than 95% full. See "Monitoring and Auditing AFS Performance" on page 295.

Rebooting a file server machine requires shutting down the AFS processes and so inevitably causes a service outage. Reboot file server machines as infrequently as possible. For instructions, see "Rebooting a Server Machine" on page 103.

By default, the BOS Server on each file server machine stops and immediately restarts all AFS server processes on the machine (including itself) once a week, at 4:00 a.m. on Sunday. This reduces the potential for the core leaks that can develop as any process runs for an extended time.

The BOS Server also checks each morning at 5:00 a.m. for any newly installed binary files in the **/usr/afs/bin** directory. It compares the timestamp on each binary file to the time at which the corresponding process last restarted. If the timestamp on the binary is later, the BOS Server restarts the corresponding process to start using it.

The default times are in the early morning hours when the outage that results from restarting a process is likely to disturb the fewest number of people. You can display the restart times for each machine with the **bos getrestart** command, and set them with the **bos setrestart** command. The latter command enables you to disable automatic restarts entirely, by setting the time to **never**. See "Setting the BOS Server's Restart Times" on page 126.

Configuring Client Machines

This section summarizes issues to consider as you install and configure client machines in your cell.

Configuring the Local Disk

You can often free up significant amounts of local disk space on AFS client machines by storing standard UNIX files in AFS and creating symbolic links to them from the local disk. The **@sys** pathname variable can be useful in links to system-specific files; see "Using the @sys Variable in Pathnames" on page 34.

There are two types of files that must actually reside on the local disk: boot sequence files needed before the **afsd** program is invoked, and files that can be helpful during file server machine outages.

During a reboot, AFS is inaccessible until the **afsd** program executes and initializes the Cache Manager. (In the conventional configuration, the AFS initialization file is included in the machine's initialization sequence and invokes the **afsd** program.) Files needed during reboot prior to that point must reside on the local disk. They include the following, but this list is not necessarily exhaustive.

- Standard UNIX utilities including the following or their equivalents:

  - Machine initialization files (stored in the **/etc** or **/sbin** directory on many system types)
  - The **fstab** file
  - The **mount** command binary
  - The **umount** command binary

- All subdirectories and files in the **/usr/vice** directory, including the following:
  - The **/usr/vice/cache** directory
  - The **/usr/vice/etc/afsd** command binary
  - The **/usr/vice/etc/cacheinfo** file
  - The **/usr/vice/etc/CellServDB** file
  - The **/usr/vice/etc/ThisCell** file

  For more information on these files, see "Configuration and Cache-Related Files on the Local Disk" on page 353.

The other type of files and programs to retain on the local disk are those you need when diagnosing and fixing problems caused by a file server outage, because the outage can make inaccessible the copies stored in AFS. Examples include the binaries for a text editor (such as **ed** or **vi**) and for the **fs** and **bos** commands. Store copies of AFS command binaries in the **/usr/vice/etc** directory as well as including them in the **/usr/afsws** directory, which is normally a link into AFS. Then place the **/usr/afsws** directory before the **/usr/vice/etc** directory in users' PATH environment variable definition. When AFS is functioning normally, users access the copy in the **/usr/afsws** directory, which is more likely to be current than a local copy.

You can automate the configuration of client machine local disks by using the **package** program, which updates the contents of the local disk to match a configuration file. See "Configuring Client Machines with the package Program" on page 389.

Enabling Access to Foreign Cells

As detailed in "Making Other Cells Visible in Your Cell" on page 23, you enable the Cache Manager to access a cell's AFS filespace by storing a list of the cell's database server machines in the local **/usr/vice/etc/CellServDB** file. The Cache Manager reads the list into kernel memory at reboot for faster retrieval. You can change the list in kernel memory between reboots by using the **fs newcell** command. It is often practical to store a central version of the **CellServDB** file in AFS and use the **package** program periodically to update each client's version with the source copy. See "Maintaining Knowledge of Database Server Machines" on page 364.

Because each client machine maintains its own copy of the **CellServDB** file, you can in theory enable access to different foreign cells on different client machines. This is not usually practical, however, especially if users do not always work on the same machine.

Using the @sys Variable in Pathnames

When creating symbolic links into AFS on the local disk, it is often practical to use the @sys variable in pathnames. The Cache Manager automatically substitutes the local machine's AFS system name (CPU/operating system type) for the @sys variable. This means you can place the same links on machines of various system types and still have each machine access the binaries for its system type. For example, the Cache Manager on a machine running AIX 4.2 converts **/afs/abc.com/@sys** to **/afs/abc.com/rs_aix42**, whereas a machine running Solaris 7 converts it to **/afs/abc.com/sun4x_57**.

If you want to use the @sys variable, it is simplest to use the conventional AFS system type names as specified in the IBM AFS Release Notes. The Cache Manager records the local machine's system type name in kernel memory during initialization. If you do not use the conventional names, you must use the **fs sysname** command to change the value in kernel memory from its default just after Cache Manager initialization, on every client machine of the relevant system type. The **fs sysname** command also displays the current value; see "Displaying and Setting the System Type Name" on page 383.

In pathnames in the AFS filespace itself, use the @sys variable carefully and sparingly, because it can lead to unexpected results. It is generally best to restrict its use to only one level in the filespace. The third level is a common choice, because that is where many cells store the binaries for different machine types.

Multiple instances of the @sys variable in a pathname are especially dangerous to people who must explicitly change directories (with the **cd** command, for example) into directories that store binaries for system types other than the machine on which they are working, such as administrators or developers who maintain those directories. After changing directories, it is recommended that such people verify they are in the desired directory.

Setting Server Preferences

The Cache Manager stores a table of preferences for file server machines in kernel memory. A preference rank pairs a file server machine interface's IP address with an integer in the range from 1 to 65,534. When it needs to access a file, the Cache Manager compares the ranks for the interfaces of all machines that house the file, and first attempts to access the file via the interface with the best rank. As it initializes, the Cache Manager sets default ranks that bias it to access files via interfaces that are close to it in terms of network topology. You can adjust the preference ranks to improve performance if you wish.

The Cache Manager also uses similar preferences for Volume Location (VL) Server machines. Use the **fs getserverprefs** command to display preference ranks and the **fs setserverprefs** command to set them. See "Maintaining Server Preference Ranks" on page 375.

Configuring AFS User Accounts

This section discusses some of the issues to consider when configuring AFS user accounts. Because AFS is separate from the UNIX file system, a user's AFS account is separate from her UNIX account.

The preferred method for creating a user account is with the **uss** suite of commands. With a single command, you can create all the components of one or many accounts, after you have prepared a template file that guides the account creation. See "Creating and Deleting User Accounts with the uss Command Suite" on page 413.

Alternatively, you can issue the individual commands that create each component of an account. For instructions, along with instructions for removing user accounts and changing user passwords, user volume quotas and usernames, see "Administering User Accounts" on page 459.

When users leave your system, it is often good policy to remove their accounts. Instructions appear in "Deleting Individual Accounts with the uss delete Command" on page 448 and "Removing a User Account" on page 481.

An AFS user account consists of the following components, which are described in greater detail in "The Components of an AFS User Account" on page 459.

- A Protection Database entry
- An Authentication Database entry
- A volume
- A home directory at which the volume is mounted
- Ownership of the home directory and full permissions on its ACL
- An entry in the local password file (**/etc/passwd** or equivalent) of each machine the user needs to log into
- Optionally, standard files and subdirectories that make the account more useful

By creating some components but not others, you can create accounts at different levels of functionality, using either **uss** commands as described in "Creating and Deleting User Accounts with the uss Command Suite" on page 413 or individual commands as described in "Administering User Accounts" on page 459. The levels of functionality include the following

- An authentication-only account enables the user to obtain AFS tokens and so to access protected AFS data and to issue privileged commands. It consists only of entries in the Authentication and Protection Database. This type of account is suitable for administrative accounts and for users from foreign cells who need to access protected data. Local users generally also need a volume and home directory.
- A basic user account includes a volume for the user, in addition to Authentication and Protection Database entries. The volume is mounted in the AFS filespace as the user's home directory, and provides a repository for the user's personal files.
- A full account adds configuration files for basic functions such as logging in, printing, and mail delivery to a basic account, making it more convenient and useful. For a discussion of some useful types of configuration files, see "Creating Standard Files in New AFS Accounts" on page 39.

If your users have UNIX user accounts that predate the introduction of AFS in the cell, you possibly want to convert them into AFS accounts. There are three main issues to consider:

- Making UNIX and AFS UIDs match
- Setting the password field in the local password file appropriately
- Moving files from the UNIX file system into AFS

For further discussion, see "Converting Existing UNIX Accounts with uss" on page 419 or "Converting Existing UNIX Accounts" on page 462.

Choosing Usernames and Naming Other Account Components

This section suggests schemes for choosing usernames, AFS UIDs, user volume names and mount point names, and also outlines some restrictions on your choices.

Usernames

AFS imposes very few restrictions on the form of usernames. It is best to keep usernames short, both because many utilities and applications can handle usernames of no more than eight characters and because by convention many components of and AFS account incorporate the name. These include the entries in the Protection and Authentication Databases, the volume, and the mount point. Depending on your electronic mail delivery system, the username can become part of the user's mailing address. The username is also the string that the user types when logging in to a client machine.

Some common choices for usernames are last names, first names, initials, or a combination, with numbers sometimes added. It is also best to avoid using the following characters, many of which have special meanings to the command shell.

- The comma (**,**)
- The colon (**:**), because AFS reserves it as a field separator in protection group names; see "The Two Types of User-Defined Groups" on page 41
- The semicolon (**;**)
- The "at-sign" (**@**); this character is reserved for Internet mailing addresses
- Spaces
- The newline character
- The period (**.**); it is conventional to use this character only in the special username that an administrator adopts while performing privileged tasks, such as **pat.admin**

AFS UIDs and UNIX UIDs

AFS associates a unique identification number, the AFS UID, with every username, recording the mapping in the user's Protection Database entry. The AFS UID functions within AFS much as the UNIX UID does in the local file system: the AFS server processes and the Cache Manager use it internally to identify a user, rather than the username.

Every AFS user also must have a UNIX UID recorded in the local password file (**/etc/passwd** or equivalent) of each client machine they log onto. Both administration and a user's AFS access are simplest if the AFS UID and UNIX UID match. One important consequence of matching UIDs is that the owner reported by the **ls -l** command matches the AFS username.

It is usually best to allow the Protection Server to allocate the AFS UID as it creates the Protection Database entry. However, both the **pts createuser** command and the **uss** commands that create user accounts enable you to assign AFS UIDs explicitly. This is appropriate in two cases:

- You wish to group together the AFS UIDs of related users
- You are converting an existing UNIX account into an AFS account and want to make the AFS UID match the existing UNIX UID

After the Protection Server initializes for the first time on a cell's first file server machine, it starts assigning AFS UIDs at a default value. To change the default before creating any user accounts, or at any time, use the **pts setmax** command to reset the `max user id counter`. To display the counter, use the **pts listmax** command. See "Displaying and Setting the AFS UID and GID Counters" on page 509.

AFS reserves one AFS UID, 32766, for the user **anonymous**. The AFS server processes assign this identity and AFS UID to any user who does not possess a token for the local cell. Do not assign this AFS UID to any other user or hardcode its current value into any programs or a file's owner field, because it is subject to change in future releases.

User Volume Names

Like any volume name, a user volume's base (read/write) name cannot exceed 22 characters in length or include the **.readonly** or **.backup** extension. See "Creating Volumes to Simplify Administration" on page 26. By convention, user volume names have the format **user.**username. Using the **user.** prefix not only makes it easy to identify the volume's contents, but also to create a backup version of all user volumes by issuing a single **vos backupsys** command.

Mount Point Names

By convention, the mount point for a user's volume is named after the username. Many cells follow the convention of mounting user volumes in the **/afs/***`cellname`***/usr** directory, as discussed in "The Third Level" on page 25. Very large cells sometimes find that mounting all user volumes in the same directory slows directory lookup, however; for suggested alternatives, see the following section.

Grouping Home Directories

Mounting user volumes in the **/afs/***`cellname`***/usr** directory is an AFS-appropriate variation on the standard UNIX practice of putting user home directories under the **/usr** subdirectory. However, cells with more than a few hundred users sometimes find that mounting all user volumes in a single directory results in slow directory lookup. The solution is to distribute user volume mount points into several directories; there are a number of alternative methods to accomplish this.

- Distribute user home directories into multiple directories that reflect organizational divisions, such as academic or corporate departments. For example, a company can create group directories called **usr/marketing**, **usr/research**, **usr/finance**. A good feature of this scheme is that knowing a user's department is enough to find the user's home directory. Also, it makes it easy to set the ACL to limit access to members of the department only. A potential drawback arises if departments are of sufficiently unequal size that users in large departments experience slower lookup than users in small departments. This scheme is also not appropriate in cells where users frequently change between divisions.
- Distribute home directories into alphabetic subdirectories of the **usr** directory (the **usr/a** subdirectory, the **usr/b** subdirectory, and so on), based on the first letter of the username. If the cell is very large,

create subdirectories under each letter that correspond to the second letter in the user name. This scheme has the same advantages and disadvantages of a department-based scheme. Anyone who knows the user's username can find the user's home directory, but users with names that begin with popular letters sometimes experience slower lookup.

- Distribute home directories randomly but evenly into more than one grouping directory. One cell that uses this scheme has over twenty such directories called the **usr1** directory, the **usr2** directory, and so on. This scheme is especially appropriate in cells where the other two schemes do not seem feasible. It eliminates the potential problem of differences in lookup speed, because all directories are about the same size. Its disadvantage is that there is no way to guess which directory a given user's volume is mounted in, but a solution is to create a symbolic link in the regular **usr** directory that references the actual mount point. For example, if user **smith**'s volume is mounted at the **/afs/bigcell.com/usr17/smith** directory, then the **/afs/bigcell.com/usr/smith** directory is a symbolic link to the **../usr17/smith** directory. This way, if someone does not know which directory the user **smith** is in, he or she can access it through the link called **usr/smith**; people who do know the appropriate directory save lookup time by specifying it.

For instructions on how to implement the various schemes when using the **uss** program to create user accounts, see "Evenly Distributing User Home Directories with the G Instruction" on page 428 and "Creating a Volume with the V Instruction" on page 429.

Making a Backup Version of User Volumes Available

Mounting the backup version of a user's volume is a simple way to enable users themselves to restore data they have accidentally removed or deleted. It is conventional to mount the backup version at a subdirectory of the user's home directory (called perhaps the **OldFiles** subdirectory), but other schemes are possible. Once per day you create a new backup version to capture the changes made that day, overwriting the previous day's backup version with the new one. Users can always retrieve the previous day's copy of a file without your assistance, freeing you to deal with more pressing tasks.

Users sometimes want to delete the mount point to their backup volume, because they erroneously believe that the backup volume's contents count against their quota. Remind them that the backup volume is separate, so the only space it uses in the user volume is the amount needed for the mount point.

For further discussion of backup volumes, see "Backing Up AFS Data" on page 54 and "Creating Backup Volumes" on page 144.

Creating Standard Files in New AFS Accounts

From your experience as a UNIX administrator, you are probably familiar with the use of login and shell initialization files (such as the **.login** and **.cshrc** files) to make an account easier to use.

It is often practical to add some AFS-specific directories to the definition of the user's PATH environment variable, including the following:

- The path to a **bin** subdirectory in the user's home directory for binaries the user has created (that is, **/afs/**`cellname`**/usr/**`username`**/bin**)

- The **/usr/afsws/bin** path, which conventionally includes programs like **fs**, **klog**, **kpasswd**, **pts**, **tokens**, and **unlog**
- The **/usr/afsws/etc** path, if the user is an administrator; it usually houses the AFS command suites that require privilege (the **backup**, **butc**, **kas**, **uss**, **vos** commands), the **package** program, and others

If you are not using an AFS-modified login utility, it can be helpful to users to invoke the **klog** command in their **.login** file so that they obtain AFS tokens as part of logging in. In the following example command sequence, the first line echoes the string `klog` to the standard output stream, so that the user understands the purpose of the `Password:` prompt that appears when the second line is executed. The **-setpag** flag associates the new tokens with a process authentication group (PAG), which is discussed further in "Identifying AFS Tokens by PAG" on page 42.

```
echo -n "klog "
klog -setpag
```

The following sequence of commands has a similar effect, except that the **pagsh** command forks a new shell with which the PAG and tokens are associated.

```
pagsh
echo -n "klog "
klog
```

If you use an AFS-modified login utility, this sequence is not necessary, because such utilities both log a user in locally and obtain AFS tokens.

Using AFS Protection Groups

AFS enables users to define their own groups of other users or machines. The groups are placed on ACLs to grant the same permissions to many users without listing each user individually. For group creation instructions, see "Administering the Protection Database" on page 487.

Groups have AFS ID numbers, just as users do, but an AFS group ID (GID) is a negative integer whereas a user’s AFS UID is a positive integer. By default, the Protection Server allocates a new group’s AFS GID automatically, but members of the **system:administrators** group can assign a GID when issuing the **pts creategroup** command. Before explicitly assigning a GID, it is best to verify that it is not already in use.

A group cannot belong to another group, but it can own another group or even itself as long as it (the owning group) has at least one member. The current owner of a group can transfer ownership of the group to another user or group, even without the new owner’s permission. At that point the former owner loses administrative control over the group.

By default, each user can create 20 groups. A system administrator can increase or decrease this group creation quota with the **pts setfields** command.

Each Protection Database entry (group or user) is protected by a set of five privacy flagswhich limit who can administer the entry and what they can do. The default privacy flags are fairly restrictive, especially for user entries. See "Setting the Privacy Flags on Database Entries" on page 507.

The Three System Groups

As the Protection Server initializes for the first time on a cell's first database server machine, it automatically creates three group entries: the **system:anyuser**, **system:authuser**, and **system:administrators** groups.

The first two system groups are unlike any other groups in the Protection Database in that they do not have a stable membership:

- The **system:anyuser** group includes everyone who can access a cell's AFS filespace: users who have tokens for the local cell, users who have logged in on a local AFS client machine but not obtained tokens (such as the local superuser **root**), and users who have connected to a local machine from outside the cell. Placing the **system:anyuser** group on an ACL grants access to the widest possible range of users. It is the only way to extend access to users from foreign AFS cells that do not have local accounts.
- The **system:authuser** group includes everyone who has a valid token obtained from the cell's AFS authentication service.

Because the groups do not have a stable membership, the **pts membership** command produces no output for them. Similarly, they do not appear in the list of groups to which a user belongs.

The **system:administrators** group does have a stable membership, consisting of the cell's privileged administrators. Members of this group can issue any **pts** command, and are the only ones who can issue several other restricted commands (such as the **chown** command on AFS files). By default, they also implicitly have the **a** (**administer**) and **l** (**lookup**) permissions on every ACL in the filespace. For information about changing this default, see "Administering the system:administrators Group" on page 532.

For a discussion of how to use system groups effectively on ACLs, see "Using Groups on ACLs" on page 518.

The Two Types of User-Defined Groups

All users can create regular groups. A regular group name has two fields separated by a colon, the first of which must indicate the group's ownership. The Protection Server refuses to create or change the name of a group if the result does not accurately indicate the ownership.

Members of the **system:administrators** group can create prefix-less groups whose names do not have the first field that indicates ownership. For suggestions on using the two types of groups effectively, see "Using Groups Effectively" on page 498.

Login and Authentication in AFS

As explained in "Differences in Authentication" on page 16, AFS authentication is separate from UNIX authentication because the two file systems are separate. The separation has two practical implications:

- To access AFS files, users must both log into the local file system and authenticate with the AFS authentication service. (Logging into the local file system is necessary because the only way to access the AFS filespace is through a Cache Manager, which resides in the local machine's kernel.)

- Passwords are stored in two separate places: in the Authentication Database for AFS and in the each machine's local password file (the **/etc/passwd** file or equivalent) for the local file system.

When a user successfully authenticates, the AFS authentication service passes a token to the user's Cache Manager. The token is a small collection of data that certifies that the user has correctly provided the password associated with a particular AFS identity. The Cache Manager presents the token to AFS server processes along with service requests, as proof that the user is genuine. To learn about the mutual authentication procedure they use to establish identity, see "A More Detailed Look at Mutual Authentication" on page 51.

The Cache Manager stores tokens in the user's credential structure in kernel memory. To distinguish one user's credential structure from another's, the Cache Manager identifies each one either by the user's UNIX UID or by a process authentication group (PAG), which is an identification number guaranteed to be unique in the cell. For further discussion, see "Identifying AFS Tokens by PAG" on page 42.

A user can have only one token per cell in each separately identified credential structure. To obtain a second token for the same cell, the user must either log into a different machine or obtain another credential structure with a different identifier than any existing credential structure, which is most easily accomplished by issuing the **pagsh** command (see "Identifying AFS Tokens by PAG" on page 42). In a single credential structure, a user can have one token for each of many cells at the same time. As this implies, authentication status on one machine or PAG is independent of authentication status on another machine or PAG, which can be very useful to a user or system administrator.

The AFS distribution includes library files that enable each system type's login utility to authenticate users with AFS and log them into the local file system in one step. If you do not configure an AFS-modified login utility on a client machine, its users must issue the **klog** command to authenticate with AFS after logging in.

Note: The AFS-modified libraries do not necessarily support all features available in an operating system's proprietary login utility. In some cases, it is not possible to support a utility at all. For more information about the supported utilities in each AFS version, see the IBM AFS Release Notes.

Identifying AFS Tokens by PAG

As noted, the Cache Manager identifies user credential structures either by UNIX UID or by PAG. Using a PAG is preferable because it guaranteed to be unique: the Cache Manager allocates it based on a counter that increments with each use. In contrast, multiple users on a machine can share or assume the same UNIX UID, which creates potential security problems. The following are two common such situations:

- The local superuser **root** can always assume any other user's UNIX UID simply by issuing the **su** command, without providing the user's password. If the credential structure is associated with the user's UNIX UID, then assuming the UID means inheriting the AFS tokens.

- Two users working on different NFS client machines can have the same UNIX UID in their respective local file systems. If they both access the same NFS/AFS Translator machine, and the Cache Manager there identifies them by their UNIX UID, they become indistinguishable. To eliminate this problem, the Cache Manager on a translator machine automatically generates a PAG for each user and uses it, rather than the UNIX UID, to tell users apart.

Yet another advantage of PAGs over UIDs is that processes spawned by the user inherit the PAG and so share the token; thus they gain access to AFS as the authenticated user. In many environments, for example, printer and other daemons run under identities (such as the local superuser **root**) that the AFS server processes recognize only as the **anonymous** user. Unless PAGs are used, such daemons cannot access files for which the **system:anyuser** group does not have the necessary ACL permissions.

Once a user has a PAG, any new tokens the user obtains are associated with the PAG. The PAG expires two hours after any associated tokens expire or are discarded. If the user issues the **klog** command before the PAG expires, the new token is associated with the existing PAG (the PAG is said to be recycled in this case).

AFS-modified login utilities automatically generate a PAG, as described in the following section. If you use a standard login utility, your users must issue the **pagsh** command before the **klog** command, or include the latter command's **-setpag** flag. For instructions, see "Using Two-Step Login and Authentication" on page 44.

Users can also use either command at any time to create a new PAG. The difference between the two commands is that the **klog** command replaces the PAG associated with the current command shell and tokens. The **pagsh** command initializes a new command shell before creating a new PAG. If the user already had a PAG, any running processes or jobs continue to use the tokens associated with the old PAG whereas any new jobs or processes use the new PAG and its associated tokens. When you exit the new shell (by pressing **<Ctrl-d>**, for example), you return to the original PAG and shell. By default, the **pagsh** command initializes a Bourne shell, but you can include the **-c** argument to initialize a C shell (the **/bin/csh** program on many system types) or Korn shell (the **/bin/ksh** program) instead.

Using an AFS-modified login Utility

As previously mentioned, an AFS-modified login utility simultaneously obtains an AFS token and logs the user into the local file system. This section outlines the login and authentication process and its interaction with the value in the password field of the local password file.

An AFS-modified login utility performs a sequence of steps similar to the following; details can vary for different operating systems:

1. It checks the user's entry in the local password file (the **/etc/passwd** file or equivalent).
2. If no entry exists, or if an asterisk (*) appears in the entry's password field, the login attempt fails. If the entry exists, the attempt proceeds to the next step.
3. The utility obtains a PAG.
4. The utility converts the password provided by the user into an encryption key and encrypts a packet of data with the key. It sends the packet to the AFS authentication service (the AFS Authentication Server in the conventional configuration).

5. The authentication service decrypts the packet and, depending on the success of the decryption, judges the password to be correct or incorrect. (For more details, see "A More Detailed Look at Mutual Authentication" on page 51.)
   - If the authentication service judges the password incorrect, the user does not receive an AFS token. The PAG is retained, ready to be associated with any tokens obtained later. The attempt proceeds to Step "6" on page 44.
   - If the authentication service judges the password correct, it issues a token to the user as proof of AFS authentication. The login utility logs the user into the local UNIX file system. Some login utilities echo the following banner to the screen to alert the user to authentication with AFS. Step "6" on page 44 is skipped.

     ```
     AFS(R) version Login
     ```

6. If no AFS token was granted in Step "4" on page 43, the login utility attempts to log the user into the local file system, by comparing the password provided to the local password file.
   - If the password is incorrect or any value other than an encrypted 13-character string appears in the password field, the login attempt fails.
   - If the password is correct, the user is logged into the local file system only.

As indicated, when you use an AFS-modified login utility, the password field in the local password file is no longer the primary gate for access to your system. If the user provides the correct AFS password, then the program never consults the local password file. However, you can still use the password field to control access, in the following way:

- To prevent both local login and AFS authentication, place an asterisk (*****) in the field. This is useful mainly in emergencies, when you want to prevent a certain user from logging into the machine.
- To prevent login to the local file system if the user does not provide the correct AFS password, place a character string of any length other than the standard thirteen characters in the field. This is appropriate if you want to permit only people with local AFS accounts to login on your machines. A single **X** or other character is the most easily recognizable way to do this.
- To enable a user to log into the local file system even after providing an incorrect AFS password, record a standard UNIX encrypted password in the field by issuing the standard UNIX password-setting command (**passwd** or equivalent).

Systems that use a Pluggable Authentication Module (PAM) for login and AFS authentication do not necessarily consult the local password file at all, in which case they do not use the password field to control authentication and login attempts. Instead, instructions in the PAM configuration file (on many system types, **/etc/pam.conf**) fill the same function. See the instructions in the IBM AFS Quick Beginnings for installing AFS-modified login utilities.

Using Two-Step Login and Authentication

In cells that do not use an AFS-modified login utility, users must issue separate commands to login and authenticate, as detailed in the IBM AFS User Guide:

1. They use the standard **login** program to login to the local file system, providing the password listed in the local password file (the **/etc/passwd** file or equivalent).
2. They must issue the **klog** command to authenticate with the AFS authentication service, including its **-setpag** flag to associate the new tokens with a process authentication group (PAG).

As mentioned in "Creating Standard Files in New AFS Accounts" on page 39, you can invoke the **klog -setpag** command in a user's **.login** file (or equivalent) so that the user does not have to remember to issue the command after logging in. The user still must type a password twice, once at the prompt generated by the login utility and once at the **klog** command's prompt. This implies that the two passwords can differ, but it is less confusing if they do not.

Another effect of not using an AFS-modified login utility is that the AFS servers recognize the standard **login** program as the **anonymous** user. If the **login** program needs to access any AFS files (such as the **.login** file in a user's home directory), then the ACL that protects the file must include an entry granting the **l** (**lookup**) and **r** (**read**) permissions to the **system:anyuser** group.

When you do not use an AFS-modified login utility, an actual (scrambled) password must appear in the local password file for each user. Use the **/bin/passwd** file to insert or change these passwords. It is simpler if the password in the local password file matches the AFS password, but it is not required.

Obtaining, Displaying, and Discarding Tokens

Once logged in, a user can obtain a token at any time with the **klog** command. If a valid token already exists, the new one overwrites it. If a PAG already exists, the new token is associated with it.

By default, the **klog** command authenticates the issuer using the identity currently logged in to the local file system. To authenticate as a different identity, use the **-principal** argument. To obtain a token for a foreign cell, use the **-cell** argument (it can be combined with the **-principal** argument). See the IBM AFS User Guide and the entry for the **klog** command in the IBM AFS Administration Reference.

To discard either all tokens or the token for a particular cell, issue the **unlog** command. The command affects only the tokens associated with the current command shell. See the IBM AFS User Guideand the entry for the **unlog** command in the IBM AFS Administration Reference.

To display the tokens associated with the current command shell, issue the **tokens** command. The following examples illustrate its output in various situations.

If the issuer is not authenticated in any cell:

```
% tokens
Tokens held by the Cache Manager:
       --End of list--
```

The following shows the output for a user with AFS UID 1000 in the ABC Corporation cell:

```
% tokens
```

```
Tokens held by the Cache Manager:
User’s (AFS ID 1000) tokens for afs@abc.com  [Expires Jun  2 10:00]
    --End of list--
```

The following shows the output for a user who is authenticated in ABC Corporation cell, the State University cell and the DEF Company cell. The user has different AFS UIDs in the three cells. Tokens for the last cell are expired:

```
% tokens
Tokens held by the Cache Manager:
User’s (AFS ID 1000) tokens for afs@abc.com  [Expires Jun  2 10:00]
User’s (AFS ID 4286) tokens for afs@stateu.edu  [Expires Jun  3 1:34]
User’s (AFS ID 22) tokens for afs@def.com  [>>Expired<<]
    --End of list--
```

The Kerberos version of the **tokens** command (the **tokens.krb** command), also reports information on the ticket-granting ticket, including the ticket’s owner, the ticket-granting service, and the expiration date, as in the following example. Also see "Support for Kerberos Authentication" on page 47.

```
% tokens.krb
Tokens held by the Cache Manager:
User’s (AFS ID 1000) tokens for afs@abc.com [Expires Jun  2 10:00]
User smith’s tokens for krbtgt.ABC.COM@abc.com [Expires Jun  2 10:00]
  --End of list--
```

Setting Default Token Lifetimes for Users

The maximum lifetime of a user token is the smallest of the ticket lifetimes recorded in the following three Authentication Database entries. The **kas examine** command reports the lifetime as `Max ticket lifetime`. Administrators who have the `ADMIN` flag on their Authentication Database entry can use the **-lifetime** argument to the **kas setfields** command to set an entry’s ticket lifetime.

- The **afs** entry, which corresponds to the AFS server processes. The default is 100 hours.
- The **krbtgt**.cellname entry, which corresponds to the ticket-granting ticket used internally in generating the token. The default is 720 hours (30 days).
- The entry for the user of the AFS-modified login utility or issuer of the **klog** command. The default is 25 hours for user entries created using the AFS 3.1 or later version of the Authentication Server, and 100 hours for user entries created using the AFS 3.0 version of the Authentication Server. A user can use the **kas examine** command to display his or her own Authentication Database entry.

Note: An AFS-modified login utility always grants a token with a lifetime calculated from the previously described three values. When issuing the **klog** command, a user can request a lifetime shorter than the default by using the **-lifetime** argument. For further information, see the IBM AFS User Guide and the **klog** reference page in the IBM AFS Administration Reference.

Changing Passwords

Regular AFS users can change their own passwords by using either the **kpasswd** or **kas setpassword** command. The commands prompt for the current password and then twice for the new password, to screen out typing errors.

Administrators who have the `ADMIN` flag on their Authentication Database entries can change any user's password, either by using the **kpasswd** command (which requires knowing the current password) or the **kas setpassword** command.

If your cell does not use an AFS-modified login utility, remember also to change the local password, using the operating system's password-changing command. For more instructions on changing passwords, see "Changing AFS Passwords" on page 476.

Imposing Restrictions on Passwords and Authentication Attempts

You can help to make your cell more secure by imposing restrictions on user passwords and authentication attempts. To impose the restrictions as you create an account, use the **A** instruction in the **uss** template file as described in "Increasing Account Security with the A Instruction" on page 439. To set or change the values on an existing account, use the **kas setfields** command as described in "Improving Password and Authentication Security" on page 470.

By default, AFS passwords never expire. Limiting password lifetime can help improve security by decreasing the time the password is subject to cracking attempts. You can choose an lifetime from 1 to 254 days after the password was last changed. It automatically applies to each new password as it is set. When the user changes passwords, you can also insist that the new password is not similar to any of the 20 passwords previously used.

Unscrupulous users can try to gain access to your AFS cell by guessing an authorized user's password. To protect against this type of attack, you can limit the number of times that a user can consecutively fail to provide the correct password. When the limit is exceeded, the authentication service refuses further authentication attempts for a specified period of time (the lockout time). To reenable authentication attempts before the lockout time expires, an administrator must issue the **kas unlock** command.

In addition to settings on user's authentication accounts, you can improve security by automatically checking the quality of new user passwords. The **kpasswd** and **kas setpassword** commands pass the proposed password to a program or script called **kpwvalid**, if it exists. The **kpwvalid** performs quality checks and returns a code to indicate whether the password is acceptable. You can create your own program or modified the sample program included in the AFS distribution. See the **kpwvalid** reference page in the IBM AFS Administration Reference.

There are several types of quality checks that can improve password quality.

- The password is a minimum length
- The password is not a word
- The password contains both numbers and letters

Support for Kerberos Authentication

If your site is using standard Kerberos authentication rather than the AFS Authentication Server, use the modified versions of the **klog**, **pagsh**, and **tokens** commands that support Kerberos authentication. The binaries for the modified version of these commands have the same name as the standard binaries with the addition of a **.krb** extension.

Use either the Kerberos version or the standard command throughout the cell; do not mix the two versions. AFS Product Support can provide instructions on installing the Kerberos version of these four commands. For information on the differences between the two versions of these commands, see the IBM AFS Administration Reference.

Security and Authorization in AFS

AFS incorporates several features to ensure that only authorized users gain access to data. This section summarizes the most important of them and suggests methods for improving security in your cell.

Some Important Security Features

ACLs on Directories

Files in AFS are protected by the access control list (ACL) associated with their parent directory. The ACL defines which users or groups can access the data in the directory, and in what way. See "Managing Access Control Lists" on page 513.

Mutual Authentication Between Client and Server

When an AFS client and server process communicate, each requires the other to prove its identity during mutual authentication, which involves the exchange of encrypted information that only valid parties can decrypt and respond to. For a detailed description of the mutual authentication process, see "A More Detailed Look at Mutual Authentication" on page 51.

AFS server processes mutually authenticate both with one another and with processes that represent human users. After mutual authentication is complete, the server and client have established an authenticated connection, across which they can communicate repeatedly without having to authenticate again until the connection expires or one of the parties closes it. Authenticated connections have varying lifetimes.

Tokens

In order to access AFS files, users must prove their identities to the AFS authentication service by providing the correct AFS password. If the password is correct, the authentication service sends the user a token as evidence of authenticated status. See "Login and Authentication in AFS" on page 41.

Servers assign the user identity **anonymous** to users and processes that do not have a valid token. The **anonymous** identity has only the access granted to the **system:anyuser** group on ACLs.

Authorization Checking

Mutual authentication establishes that two parties communicating with one another are actually who they claim to be. For many functions, AFS server processes also check that the client whose identity they have verified is also authorized to make the request. Different requests require different kinds of privilege. See "Three Types of Privilege" on page 49.

Encrypted Network Communications

The AFS server processes encrypt particularly sensitive information before sending it back to clients. Even if an unauthorized party is able to eavesdrop on an authenticated connection, they cannot decipher encrypted data without the proper key.

The following AFS commands encrypt data because they involve server encryption keys and passwords:

- The **bos addkey** command, which adds a server encryption key to the **/usr/afs/etc/KeyFile** file
- The **bos listkeys** command, which lists the server encryption keys from the **/usr/afs/etc/KeyFile** file
- The **kpasswd** command, which changes a password in the Authentication Database
- Most commands in the **kas** command suite

In addition, the United States edition of the Update Server encrypts sensitive information (such as the contents of **KeyFile**) when distributing it. Other commands in the **bos** suite and the commands in the **fs**, **pts** and **vos** suites do not encrypt data before transmitting it.

Three Types of Privilege

AFS uses three separate types of privilege for the reasons discussed in "The Reason for Separate Privileges" on page 531.

- Membership in the **system:administrators** group. Members are entitled to issue any **pts** command and those **fs** commands that set volume quota. By default, they also implicitly have the **a** (**administer**) and **l** (**lookup**) permissions on every ACL in the file tree even if the ACL does not include an entry for them.
- The `ADMIN` flag on the Authentication Database entry. An administrator with this flag can issue any **kas** command.
- Inclusion in the **/usr/afs/etc/UserList** file. An administrator whose username appears in this file can issue any **bos**, **vos**, or **backup** command (although some **backup** commands require additional privilege as described in "Granting Administrative Privilege to Backup Operators" on page 204).

Authorization Checking versus Authentication

AFS distinguishes between authentication and authorization checking. Authentication refers to the process of proving identity. Authorization checking refers to the process of verifying that an authenticated identity is allowed to perform a certain action.

AFS implements authentication at the level of connections. Each time two parties establish a new connection, they mutually authenticate. In general, each issue of an AFS command establishes a new connection between AFS server process and client.

AFS implements authorization checking at the level of server machines. If authorization checking is enabled on a server machine, then all of the server processes running on it provide services only to authorized users. If authorization checking is disabled on a server machine, then all of the server processes perform any action for anyone. Obviously, disabling authorization checking is an extreme security exposure. For more information, see "Managing Authentication and Authorization Requirements" on page 93.

Improving Security in Your Cell

You can improve the level of security in your cell by configuring user accounts, server machines, and system administrator accounts in the indicated way.

User Accounts

- Use an AFS-modified login utility, or include the **-setpag** flag to the **klog** command, to associate the credential structure that houses tokens with a PAG rather than a UNIX UID. This prevents users from inheriting someone else's tokens by assuming their UNIX identity. For further discussion, see "Identifying AFS Tokens by PAG" on page 42.
- Encourage users to issue the **unlog** command to destroy their tokens before logging out. This forestalls attempts to access tokens left behind kernel memory. Consider including the **unlog** command in every user's **.logout** file or equivalent.

Server Machines

- Disable authorization checking only in emergencies or for very brief periods of time. It is best to work at the console of the affected machine during this time, to prevent anyone else from accessing the machine through the keyboard.
- Change the AFS server encryption key on a frequent and regular schedule. Make it difficult to guess (a long string including nonalphabetic characters, for instance). Unlike user passwords, the password from which the AFS key is derived can be longer than eight characters, because it is never used during login. The **kas setpassword** command accepts a password hundreds of characters long. For instructions, see "Managing Server Encryption Keys" on page 333.
- As much as possible, limit the number of people who can login at a server machine's console or remotely. Imposing this limit is an extra security precaution rather than a necessity. The machine cannot serve as an AFS client in this case.
- Particularly limit access to the local superuser **root** account on a server machine. The local superuser **root** has free access to important administrative subdirectories of the **/usr/afs** directory, as described in "AFS Files on the Local Disk" on page 32.
- As in any computing environment, server machines must be located in a secured area. Any other security measures are effectively worthless if unauthorized people can access the computer hardware.

System Administrators

- Limit the number of system administrators in your cell. Limit the use of system administrator accounts on publicly accessible workstations. Such machines are not secure, so unscrupulous users can install programs that try to steal tokens or passwords. If administrators must use publicly accessible workstations at times, they must issue the **unlog** command before leaving the machine.
- Create an administrative account for each administrator separate from the personal account, and assign AFS privileges only to the administrative account. The administrators must authenticate to the administrative accounts to perform duties that require privilege, which provides a useful audit trail as well.
- Administrators must not leave a machine unattended while they have valid tokens. Issue the **unlog** command before leaving.
- Use the **-lifetime** argument to the **kas setfields** command to set the token lifetime for administrative accounts to a fairly short amount of time. The default lifetime for AFS tokens is 25 hours, but 30 or 60 minutes is possibly a more reasonable lifetime for administrative tokens. The tokens for administrators who initiate AFS Backup System operations must last somewhat longer, because it can take several hours to complete some dump operations, depending on the speed of the tape device and the network connecting it to the file server machines that house the volumes is it accessing.
- Limit administrators' use of the **telnet** program. It sends unencrypted passwords across the network. Similarly, limit use of other remote programs such as **rsh** and **rcp**, which send unencrypted tokens across the network.

A More Detailed Look at Mutual Authentication

As in any file system, security is a prime concern in AFS. A file system that makes file sharing easy is not useful if it makes file sharing mandatory, so AFS incorporates several features that prevent unauthorized users from accessing data. Security in a networked environment is difficult because almost all procedures require transmission of information across wires that almost anyone can tap into. Also, many machines on networks are powerful enough that unscrupulous users can monitor transactions or even intercept transmissions and fake the identity of one of the participants.

The most effective precaution against eavesdropping and information theft or fakery is for servers and clients to accept the claimed identity of the other party only with sufficient proof. In other words, the nature of the network forces all parties on the network to assume that the other party in a transaction is not genuine until proven so. Mutual authentication is the means through which parties prove their genuineness.

Because the measures needed to prevent fakery must be quite sophisticated, the implementation of mutual authentication procedures is complex. The underlying concept is simple, however: parties prove their identities by demonstrating knowledge of a shared secret. A shared secret is a piece of information known only to the parties who are mutually authenticating (they can sometimes learn it in the first place from a trusted third party or some other source). The party who originates the transaction presents the shared secret and refuses to accept the other party as valid until it shows that it knows the secret too.

The most common form of shared secret in AFS transactions is the encryption key, also referred to simply as a key. The two parties use their shared key to encrypt the packets of information they send and to decrypt the ones they receive. Encryption using keys actually serves two related purposes. First, it protects messages as they cross the network, preventing anyone who does not know the key from eavesdropping. Second, ability to encrypt and decrypt messages successfully indicates that the parties are using the key (it is their shared secret). If they are using different keys, messages remain scrambled and unintelligible after decryption.

The following sections describe AFS's mutual authentication procedures in more detail. Feel free to skip these sections if you are not interested in the mutual authentication process.

Simple Mutual Authentication

Simple mutual authentication involves only one encryption key and two parties, generally a client and server. The client contacts the server by sending a challenge message encrypted with a key known only to the two of them. The server decrypts the message using its key, which is the same as the client's if they really do share the same secret. The server responds to the challenge and uses its key to encrypt its response. The client uses its key to decrypt the server's response, and if it is correct, then the client can be sure that the server is genuine: only someone who knows the same key as the client can decrypt the challenge and answer it correctly. On its side, the server concludes that the client is genuine because the challenge message made sense when the server decrypted it.

AFS uses simple mutual authentication to verify user identities during the first part of the login procedure. In that case, the key is based on the user's password.

Complex Mutual Authentication

Complex mutual authentication involves three encryption keys and three parties. All secure AFS transactions (except the first part of the login process) employ complex mutual authentication.

When a client wishes to communicate with a server, it first contacts a third party called a ticket-granter. The ticket-granter and the client mutually authenticate using the simple procedure. When they finish, the ticket-granter gives the client a server ticket (or simply ticket) as proof that it (the ticket-granter) has preverified the identity of the client. The ticket-granter encrypts the ticket with the first of the three keys, called the server encryption key because it is known only to the ticket-granter and the server the client wants to contact. The client does not know this key.

The ticket-granter sends several other pieces of information along with the ticket. They enable the client to use the ticket effectively despite being unable to decrypt the ticket itself. Along with the ticket, the items constitute a token:

- A session key, which is the second encryption key involved in mutual authentication. The ticket-granter invents the session key at random as the shared secret between client and server. For reasons explained further below, the ticket-granter also puts a copy of the session key inside the ticket. The client and server use the session key to encrypt messages they send to one another during their transactions. The ticket-granter invents a different session key for each connection between a client and a server (there can be several transactions during a single connection).
- The name of the server for which the ticket is valid (and so which server encryption key encrypts the ticket itself).

- A ticket lifetime indicator. The default lifetime of AFS server tickets is 100 hours. If the client wants to contact the server again after the ticket expires, it must contact the ticket-granter to get a new ticket.

The ticket-granter seals the entire token with the third key involved in complex mutual authentication--the key known only to it (the ticket-granter) and the client. In some cases, this third key is derived from the password of the human user whom the client represents.

Now that the client has a valid server ticket, it is ready to contact the server. It sends the server two things:

- The server ticket. This is encrypted with the server encryption key.
- Its request message, encrypted with the session key. Encrypting the message protects it as it crosses the network, since only the server/client pair for whom the ticket-granter invented the session key know it.

At this point, the server does not know the session key, because the ticket-granter just created it. However, the ticket-granter put a copy of the session key inside the ticket. The server uses the server encryption key to decrypts the ticket and learns the session key. It then uses the session key to decrypt the client's request message. It generates a response and sends it to the client. It encrypts the response with the session key to protect it as it crosses the network.

This step is the heart of mutual authentication between client and server, because it proves to both parties that they know the same secret:

- The server concludes that the client is authorized to make a request because the request message makes sense when the server decrypts it using the session key. If the client uses a different session key than the one the server finds inside the ticket, then the request message remains unintelligible even after decryption. The two copies of the session key (the one inside the ticket and the one the client used) can only be the same if they both came from the ticket-granter. The client cannot fake knowledge of the session key because it cannot look inside the ticket, sealed as it is with the server encryption key known only to the server and the ticket-granter. The server trusts the ticket-granter to give tokens only to clients with whom it (the ticket-granter) has authenticated, so the server decides the client is legitimate.

  (Note that there is no direct communication between the ticket-granter and the server, even though their relationship is central to ticket-based mutual authentication. They interact only indirectly, via the client's possession of a ticket sealed with their shared secret.)

- The client concludes that the server is genuine and trusts the response it gets back from the server, because the response makes sense after the client decrypts it using the session key. This indicates that the server encrypted the response with the same session key as the client knows. The only way for the server to learn that matching session key is to decrypt the ticket first. The server can only decrypt the ticket because it shares the secret of the server encryption key with the ticket-granter. The client trusts the ticket-granter to give out tickets only for legitimate servers, so the client accepts a server that can decrypt the ticket as genuine, and accepts its response.

Backing Up AFS Data

AFS provides two related facilities that help the administrator back up AFS data: backup volumes and the AFS Backup System.

Backup Volumes

The first facility is the backup volume, which you create by cloning a read/write volume. The backup volume is read-only and so preserves the state of the read/write volume at the time the clone is made.

Backup volumes can ease administration if you mount them in the file system and make their contents available to users. For example, it often makes sense to mount the backup version of each user volume as a subdirectory of the user's home directory. A conventional name for this mount point is **OldFiles**. Create a new version of the backup volume (that is, reclone the read/write) once a day to capture any changes that were made since the previous backup. If a user accidentally removes or changes data, the user can restore it from the backup volume, rather than having to ask you to restore it.

The IBM AFS User Guide does not mention backup volumes, so regular users do not know about them if you decide not to use them. This implies that if you **do** make backup versions of user volumes, you need to tell your users about how the backup works and where you have mounted it.

Users are often concerned that the data in a backup volume counts against their volume quota and some of them even want to remove the **OldFiles** mount point. It does not, because the backup volume is a separate volume. The only amount of space it uses in the user's volume is the amount needed for the mount point, which is about the same as the amount needed for a standard directory element.

Backup volumes are discussed in detail in "Creating Backup Volumes" on page 144.

The AFS Backup System

Backup volumes can reduce restoration requests, but they reside on disk and so do not protect data from loss due to hardware failure. Like any file system, AFS is vulnerable to this sort of data loss.

To protect your cell's users from permanent loss of data, you are strongly urged to back up your file system to tape on a regular and frequent schedule. The AFS Backup System is available to ease the administration and performance of backups. For detailed information about the AFS Backup System, see "Configuring the AFS Backup System" on page 195 and "Backing Up and Restoring AFS Data" on page 241.

Using UNIX Remote Services in the AFS Environment

The AFS distribution includes modified versions of several standard UNIX commands, daemons and programs that provide remote services, including the following:

- The **ftpd** program
- The **inetd** daemon
- The **rcp** program

- The **rlogind** daemon
- The **rsh** command

These modifications enable the commands to handle AFS authentication information (tokens). This enables issuers to be recognized on the remote machine as an authenticated AFS user.

Replacing the standard versions of these programs in your file tree with the AFS-modified versions is optional. It is likely that AFS's transparent access reduces the need for some of the programs anyway, especially those involved in transferring files from machine to machine, like the **ftpd** and **rcp** programs.

If you decide to use the AFS versions of these commands, be aware that several of them are interdependent. For example, the passing of AFS authentication information works correctly with the **rcp** command only if you are using the AFS version of both the **rcp** and **inetd** commands.

The conventional installation location for the modified remote commands are the **/usr/afsws/bin** and **/usr/afsws/etc** directories. To learn more about commands' functionality, see their reference pages in the IBM AFS Administration Reference.

Accessing AFS through NFS

Users of NFS client machines can access the AFS filespace by mounting the **/afs** directory of an AFS client machine that is running the NFS/AFS Translator. This is a particular advantage in cells already running NFS who want to access AFS using client machines for which AFS is not available. See "Appendix A, Managing the NFS/AFS Translator" on page 539.

II. Managing File Server Machines

Chapter 3. Administering Server Machines

This chapter describes how to administer an AFS server machine. It describes the following configuration information and administrative tasks:

- The binary and configuration files that must reside in the subdirectories of the **/usr/afs** directory on every server machine's local disk; see "Local Disk Files on a Server Machine" on page 60.
- The various *roles* or functions that an AFS server machine can perform, and how to determine which machines are taking a role; see "The Four Roles for File Server Machines" on page 68.
- How to maintain database server machines; see "Administering Database Server Machines" on page 74.
- How to maintain the list of database server machines in the **/usr/afs/etc/CellServDB** file; see "Maintaining the Server CellServDB File" on page 88.
- How to control authorization checking on a server machine; see "Managing Authentication and Authorization Requirements" on page 93.
- How to install new disks or partitions on a file server machine; see "Adding or Removing Disks and Partitions" on page 95.
- How to change a server machine's IP addresses and manager VLDB server entries; see "Managing Server IP Addresses and VLDB Server Entries" on page 99.
- How to reboot a file server machine; see "Rebooting a Server Machine" on page 103.

To learn how to install and configure a new server machine, see the *IBM AFS Quick Beginnings*.

To learn how to administer the server processes themselves, see "Monitoring and Controlling Server Processes" on page 105.

To learn how to administer volumes, see "Managing Volumes" on page 131.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Install new binaries	**bos install**
Examine binary check-and-restart time	**bos getrestart**
Set binary check-and-restart time	**bos setrestart**
Examine compilation dates on binary files	**bos getdate**
Restart a process to use new binaries	**bos restart**
Revert to old version of binaries	**bos uninstall**
Remove obsolete **.BAK** and **.OLD** versions	**bos prune**
List partitions on a file server machine	**vos listpart**
Shutdown AFS server processes	**bos shutdown**
List volumes on a partition	**vos listvldb**
Move read/write volumes	**vos move**

List a cell's database server machines	**bos listhosts**
Add a database server machine to server **CellServDB** file	**bos addhost**
Remove a database server machine from server **CellServDB** file	**bos removehost**
Set authorization checking requirements	**bos setauth**
Prevent authentication for **bos**, **pts**, and **vos** commands	Include **-noauth** flag
Prevent authentication for kas commands	Include **-noauth** flag on some commands or issue **noauthentication** while in interactive mode
Display all VLDB server entries	**vos listaddrs**
Remove a VLDB server entry	**vos changeaddr**
Reboot a server machine remotely	**bos exec** *reboot_command*

Local Disk Files on a Server Machine

Several types of files must reside in the subdirectories of the **/usr/afs** directory on an AFS server machine's local disk. They include binaries, configuration files, the administrative database files (on database server machines), log files, and volume header files.

Note for Windows users: Some files described in this document possibly do not exist on machines that run a Windows operating system. Also, Windows uses a backslash (****) rather than a forward slash (**/**) to separate the elements in a pathname.

Binaries in the /usr/afs/bin Directory

The **/usr/afs/bin** directory stores the AFS server process and command suite binaries appropriate for the machine's system (CPU and operating system) type. If a process has both a server portion and a client portion (as with the Update Server) or if it has separate components (as with the **fs** process), each component resides in a separate file.

To ensure predictable system performance, all file server machines must run the same AFS build version of a given process. To maintain consistency easily, use the Update Server process to distribute binaries from a binary distribution machine of each system type, as described further in "Binary Distribution Machines" on page 70.

It is best to keep the binaries for all processes in the **/usr/afs/bin** directory, even if you do not run the process actively on the machine. It simplifies the process of reconfiguring machines (for example, adding database server functionality to an existing file server machine). Similarly, it is best to keep the command suite binaries in the directory, even if you do not often issue commands while working on the server machine. It enables you to issue commands during recovery from server and machine outages.

The following lists the binary files in the **/usr/afs/bin** directory that are directly related to the AFS server processes or command suites. Other binaries (for example, for the **klog** command) sometimes appear in this directory on a particular file server machine's disk or in an AFS distribution.

backup

The command suite for the AFS Backup System (the binary for the Backup Server is **buserver**).

bos

The command suite for communicating with the Basic OverSeer (BOS) Server (the binary for the BOS Server is **bosserver**).

bosserver

The binary for the Basic OverSeer (BOS) Server process.

buserver

The binary for the Backup Server process.

fileserver

The binary for the File Server component of the **fs** process.

kas

The command suite for communicating with the Authentication Server (the binary for the Authentication Server is **kaserver**).

kaserver

The binary for the Authentication Server process.

ntpd

The binary for the Network Time Protocol Daemon (NTPD). AFS redistributes this binary and uses the **runntp** program to configure and initialize the NTPD process.

ntpdc

A debugging utility furnished with the **ntpd** program.

pts

The command suite for communicating with the Protection Server process (the binary for the Protection Server is **ptserver**).

ptserver

The binary for the Protection Server process.

runntp

The binary for the program used to configure NTPD most appropriately for use with AFS.

salvager

The binary for the Salvager component of the **fs** process.

udebug

The binary for a program that reports the status of AFS's distributed database technology, Ubik.

upclient

The binary for the client portion of the Update Server process.

upserver

The binary for the server portion of the Update Server process.

vlserver

The binary for the Volume Location (VL) Server process.

volserver

The binary for the Volume Server component of the **fs** process.

vos

The command suite for communicating with the Volume and VL Server processes (the binaries for the servers are **volserver** and **vlserver**, respectively).

Common Configuration Files in the /usr/afs/etc Directory

The directory **/usr/afs/etc** on every file server machine's local disk contains configuration files in ASCII and machine-independent binary format. For predictable AFS performance throughout a cell, all server machines must have the same version of each configuration file:

- Cells that run the United States edition of AFS conventionally use the Update Server to distribute a common version of each file from the cell's system control machine to other server machines (for more on the system control machine, see "The System Control Machine" on page 70). Run the Update Server's server portion on the system control machine, and the client portion on all other server machines. Update the files on the system control machine only, except as directed by instructions for dealing with emergencies.
- Cells that run the international edition of AFS must not use the Update Server to distribute the contents of the **/usr/afs/etc** directory. Due to United States government regulations, the data encryption routines that AFS uses to protect the files in this directory as they cross the network are not available to the Update Server in the international edition of AFS. You must instead update the files on each server machine individually, taking extra care to issue exactly the same **bos** command for each machine. The necessary data encryption routines are available to the **bos** commands, so information is safe as it crosses the network from the machine where the **bos** command is issued to the server machines.

Never directly edit any of the files in the **/usr/afs/etc** directory, except as directed by instructions for dealing with emergencies. In normal circumstances, use the appropriate **bos** commands to change the files. The following list includes pointers to instructions.

The files in this directory include:

CellServDB

An ASCII file that names the cell's database server machines, which run the Authentication, Backup, Protection, and VL Server processes. You create the initial version of this file by issuing the **bos setcellname** command while installing your cell's first server machine. It is very important to update this file when you change the identity of your cell's database server machines.

The server **CellServDB** file is not the same as the **CellServDB** file stored in the **/usr/vice/etc** directory on client machines. The client version lists the database server machines for every AFS cell that you choose to make accessible from the client machine. The server **CellServDB** file lists only the local cell's database server machines, because server processes never contact processes in other cells.

For instructions on maintaining this file, see "Maintaining the Server CellServDB File" on page 88.

KeyFile

A machine-independent, binary-format file that lists the server encryption keys the AFS server processes use to encrypt and decrypt tickets. The information in this file is the basis for secure communication in the cell, and so is extremely sensitive. The file is specially protected so that only privileged users can read or change it.

For instructions on maintaining this file, see "Managing Server Encryption Keys" on page 333.

ThisCell

An ASCII file that consists of a single line defining the complete Internet domain-style name of the cell (such as `abc.com`). You create this file with the **bos setcellname** command during the installation of your cell's first file server machine, as instructed in the *IBM AFS Quick Beginnings*.

Note that changing this file is only one step in changing your cell's name. For discussion, see "Choosing a Cell Name" on page 19.

UserList

An ASCII file that lists the usernames of the system administrators authorized to issue privileged **bos**, **vos**, and **backup** commands. For instructions on maintaining the file, see "Administering the UserList File" on page 535.

Local Configuration Files in the /usr/afs/local Directory

The directory **/usr/afs/local** contains configuration files that are different for each file server machine in a cell. Thus, they are not updated automatically from a central source like the files in **/usr/afs/bin** and **/usr/afs/etc** directories. The most important file is the **BosConfig** file; it defines which server processes are to run on that machine.

As with the common configuration files in **/usr/afs/etc**, you must not edit these files directly. Use commands from the **bos** command suite where appropriate; some files never need to be altered.

The files in this directory include the following:

BosConfig

This file lists the server processes to run on the server machine, by defining which processes the BOS Server monitors and what it does if the process fails. It also defines the times at which the BOS Server automatically restarts processes for maintenance purposes.

As you create server processes during a file server machine's installation, their entries are defined in this file automatically. The *IBM AFS Quick Beginnings* outlines the **bos** commands to use. For a more complete description of the file, and instructions for controlling process status by editing the file with commands from the **bos** suite, see "Monitoring and Controlling Server Processes" on page 105.

NetInfo

This optional ASCII file lists one or more of the network interface addresses on the server machine. If it exists when the File Server initializes, the File Server uses it as the basis for the list of interfaces that it registers in its Volume Location Database (VLDB) server entry. See "Managing Server IP Addresses and VLDB Server Entries" on page 99.

NetRestrict

This optional ASCII file lists one or more network interface addresses. If it exists when the File Server initializes, the File Server removes the specified addresses from the list of interfaces that it registers in its VLDB server entry. See "Managing Server IP Addresses and VLDB Server Entries" on page 99.

NoAuth

This zero-length file instructs all AFS server processes running on the machine not to perform authorization checking. Thus, they perform any action for any user, even **anonymous**. This very insecure state is useful only in rare instances, mainly during the installation of the machine.

The file is created automatically when you start the initial **bosserver** process with the **-noauth** flag, or issue the **bos setauth** command to turn off authentication requirements. When you use the **bos setauth** command to turn on authentication, the BOS Server removes this file. For more information, see "Managing Authentication and Authorization Requirements" on page 93.

SALVAGE.fs

This zero-length file controls how the BOS Server handles a crash of the File Server component of the **fs** process. The BOS Server creates this file each time it starts or restarts the **fs** process. If the file is present when the File Server crashes, then the BOS Server runs the Salvager before restarting the File Server and Volume Server again. When the File Server exits normally, the BOS Server removes the file so that the Salvager does not run.

Do not create or remove this file yourself; the BOS Server does so automatically. If necessary, you can salvage a volume or partition by using the **bos salvage** command; see "Salvaging Volumes" on page 172.

salvage.lock

This file guarantees that only one Salvager process runs on a file server machine at a time (the single process can fork multiple subprocesses to salvage multiple partitions in parallel). As the Salvager initiates (when invoked by the BOS Server or by issue of the **bos salvage** command), it creates this zero-length file and issues the **flock** system call on it. It removes the file when it completes the salvage operation. Because the Salvager must lock the file in order to run, only one Salvager can run at a time.

sysid

This file records the network interface addresses that the File Server (**fileserver** process) registers in its VLDB server entry. When the Cache Manager requests volume location information, the Volume Location (VL) Server provides all of the interfaces registered for each server machine that houses the volume. This enables the Cache Manager to make use of multiple addresses when accessing AFS data stored on a multihomed file server machine. For further information, see "Managing Server IP Addresses and VLDB Server Entries" on page 99.

Replicated Database Files in the /usr/afs/db Directory

The directory **/usr/afs/db** contains two types of files pertaining to the four replicated databases in the cell--the Authentication Database, Backup Database, Protection Database, and Volume Location Database (VLDB):

- A file that contains each database, with a **.DB0** extension.
- A log file for each database, with a **.DBSYS1** extension. The database server process logs each database operation in this file before performing it. If the operation is interrupted, the process consults this file to learn how to finish it.

Each database server process (Authentication, Backup, Protection, or VL Server) maintains its own database and log files. The database files are in binary format, so you must always access or alter them using commands from the **kas** suite (for the Authentication Database), **backup** suite (for the Backup Database), **pts** suite (for the Protection Database), or **vos** suite (for the VLDB).

If a cell runs more than one database server machine, each database server process keeps its own copy of its database on its machine's hard disk. However, it is important that all the copies of a given database are the same. To synchronize them, the database server processes call on AFS's distributed database technology, Ubik, as described in "Replicating the AFS Administrative Databases" on page 74.

The files listed here appear in this directory only on database server machines. On non-database server machines, this directory is empty.

bdb.DB0

The Backup Database file.

bdb.DBSYS1

The Backup Database log file.

kaserver.DB0

The Authentication Database file.

kaserver.DBSYS1

The Authentication Database log file.

prdb.DB0

The Protection Database file.

prdb.DBSYS1

The Protection Database log file.

vldb.DB0

The Volume Location Database file.

vldb.DBSYS1

The Volume Location Database log file.

Log Files in the /usr/afs/logs Directory

The **/usr/afs/logs** directory contains log files from various server processes. The files detail interesting events that occur during normal operations. For instance, the Volume Server can record volume moves in the **VolserLog** file. Events are recorded at completion, so the server processes do not use these files to reconstruct failed operations unlike the ones in the **/usr/afs/db** directory.

The information in log files can be very useful as you evaluate process failures and other problems. For instance, if you receive a timeout message when you try to access a volume, checking the **FileLog** file possibly provides an explanation, showing that the File Server was unable to attach the volume. To examine a log file remotely, use the **bos getlog** command as described in "Displaying Server Process Log Files" on page 128.

This directory also contains the core image files generated if a process being monitored by the BOS Server crashes. The BOS Server attempts to add an extension to the standard **core** name to indicate which process generated the core file (for example, naming a core file generated by the Protection Server **core.ptserver**). The BOS Server cannot always assign the correct extension if two processes fail at about the same time, so it is not guaranteed to be correct.

The directory contains the following files:

AuthLog

The Authentication Server's log file.

BackupLog

The Backup Server's log file.

BosLog

The BOS Server's log file.

FileLog

The File Server's log file.

SalvageLog

The Salvager's log file.

VLLog

The Volume Location (VL) Server's log file.

VolserLog

The Volume Server's log file.

core.process

If present, a core image file produced as an AFS server process on the machine crashed (probably the process named by process).

Note: To prevent log files from growing unmanageably large, restart the server processes periodically, particularly the database server processes. To avoid restarting the processes, use the UNIX **rm** command to remove the file as the process runs; it re-creates it automatically.

Volume Headers on Server Partitions

A partition that houses AFS volumes must be mounted at a subdirectory of the machine's root (/) directory (not, for instance under the **/usr** directory). The file server machine's file system registry file (**/etc/fstab** or equivalent) must correctly map the directory name and the partition's device name. The directory name is of the form **/vicep**index, where each index is one or two lowercase letters. By convention, the first AFS partition on a machine is mounted at **/vicepa**, the second at **/vicepb**, and so on. If there are more than 26 partitions, continue with **/vicepaa**, **/vicepab** and so on. The *IBM AFS Release Notes* specifies the number of supported partitions per server machine.

Do not store non-AFS files on AFS partitions. The File Server and Volume Server expect to have available all of the space on the partition.

The **/vicep** directories contain two types of files:

Vvol_ID.vol

Each such file is a volume header. The vol_ID corresponds to the volume ID number displayed in the output from the **vos examine**, **vos listvldb**, and **vos listvol** commands.

FORCESALVAGE

This zero-length file triggers the Salvager to salvage the entire partition. The AFS-modified version of the **fsck** program creates this file if it discovers corruption.

Note: For most system types, it is important never to run the standard **fsck** program provided with the operating system on an AFS file server machine. It removes all AFS volume data from server partitions because it does not recognize their format.

The Four Roles for File Server Machines

In cells that have more than one server machine, not all server machines have to perform exactly the same functions. The are four possible *roles* a machine can assume, determined by which server processes it is running. A machine can assume more than one role by running all of the relevant processes. The following list summarizes the four roles, which are described more completely in subsequent sections.

- A *simple file server* machine runs only the processes that store and deliver AFS files to client machines. You can run as many simple file server machines as you need to satisfy your cell's performance and disk space requirements.
- A *database server machine* runs the four database server processes that maintain AFS's replicated administrative databases: the Authentication, Backup, Protection, and Volume Location (VL) Server processes.
- A *binary distribution machine* distributes the AFS server binaries for its system type to all other server machines of that system type.
- The single *system control machine* distributes common server configuration files to all other server machines in the cell, in a cell that runs the United States edition of AFS (cells that use the international edition of AFS must not use the system control machine for this purpose). The machine conventionally also serves as the time synchronization source for the cell, adjusting its clock according to a time source outside the cell.

If a cell has a single server machine, it assumes the simple file server and database server roles. The instructions in the *IBM AFS Quick Beginnings* also have you configure it as the system control machine and binary distribution machine for its system type, but it does not actually perform those functions until you install another server machine.

It is best to keep the binaries for all of the AFS server processes in the **/usr/afs/bin** directory, even if not all processes are running. You can then change which roles a machine assumes simply by starting or stopping the processes that define the role.

Simple File Server Machines

A *simple file server machine* runs only the server processes that store and deliver AFS files to client machines, monitor process status, and pick up binaries and configuration files from the cell's binary distribution and system control machines.

In general, only cells with more than three server machines need to run simple file server machines. In cells with three or fewer machines, all of them are usually database server machines (to benefit from replicating the administrative databases); see "Database Server Machines" on page 69.

The following processes run on a simple file server machine:

- The BOS Server (**bosserver** process)
- The **fs** process, which combines the File Server, Volume Server, and Salvager processes so that they can coordinate their operations on the data in volumes and avoid the inconsistencies that can result from multiple simultaneous operations on the same data
- The NTP coordinator (**runntp** process), which helps keep the machine's clock synchronized with the clocks on the other server machines in the cell
- A client portion of the Update Server that picks up binary files from the binary distribution machine of its AFS system type (the **upclientbin** process)
- A client portion of the Update Server that picks up common configuration files from the system control machine, in cells running the United States edition of AFS (the **upclientetc** process)

Database Server Machines

A *database server machine* runs the four processes that maintain the AFS replicated administrative databases: the Authentication Server, Backup Server, Protection Server, and Volume Location (VL) Server, which maintain the Authentication Database, Backup Database, Protection Database, and Volume Location Database (VLDB), respectively. To review the functions of these server processes and their databases, see "AFS Server Processes and the Cache Manager" on page 8.

If a cell has more than one server machine, it is best to run more than one database server machine, but more than three are rarely necessary. Replicating the databases in this way yields the same benefits as replicating volumes: increased availability and reliability of information. If one database server machine or process goes down, the information in the database is still available from others. The load of requests for database information is spread across multiple machines, preventing any one from becoming overloaded.

Unlike replicated volumes, however, replicated databases do change frequently. Consistent system performance demands that all copies of the database always be identical, so it is not possible to record changes in only some of them. To synchronize the copies of a database, the database server processes use AFS's distributed database technology, Ubik. See "Replicating the AFS Administrative Databases" on page 74.

It is critical that the AFS server processes on every server machine in a cell know which machines are the database server machines. The database server processes in particular must maintain constant contact with their peers in order to coordinate the copies of the database. The other server processes often need information from the databases. Every file server machine keeps a list of its cell's database server machines in its local **/usr/afs/etc/CellServDB** file. Cells that use the States edition of AFS can use the system control machine to distribute this file (see "The System Control Machine" on page 70).

The following processes define a database server machine:

- The Authentication Server (**kaserver** process)
- The Backup Server (**buserver** process)
- The Protection Server (**ptserver** process)
- The VL Server (**vlserver** process)

Database server machines can also run the processes that define a simple file server machine, as listed in "Simple File Server Machines" on page 68. One database server machine can act as the cell's system control machine, and any database server machine can serve as the binary distribution machine for its system type; see "The System Control Machine" on page 70 and "Binary Distribution Machines" on page 70.

Binary Distribution Machines

A *binary distribution machine* stores and distributes the binary files for the AFS processes and command suites to all other server machines of its system type. Each file server machine keeps its own copy of AFS server process binaries on its local disk, by convention in the **/usr/afs/bin** directory. For consistent system performance, however, all server machines must run the same version (build level) of a process. For instructions for checking a binary's build level, see "Displaying A Binary File's Build Level" on page 87. The easiest way to keep the binaries consistent is to have a binary distribution machine of each system type distribute them to its system-type peers.

The process that defines a binary distribution machine is the server portion of the Update Server (**upserver** process). The client portion of the Update Server (**upclientbin** process) runs on the other server machines of that system type and references the binary distribution machine.

Binary distribution machines usually also run the processes that define a simple file server machine, as listed in "Simple File Server Machines" on page 68. One binary distribution machine can act as the cell's system control machine, and any binary distribution machine can serve as a database server machine; see "The System Control Machine" on page 70 and "Database Server Machines" on page 69.

The System Control Machine

In cells that run the United States edition of AFS, the *system control machine* stores and distributes system configuration files shared by all of the server machines in the cell. Each file server machine keeps its own copy of the configuration files on its local disk, by convention in the **/usr/afs/etc** directory. For consistent system performance, however, all server machines must use the same files. The easiest way to keep the files consistent is to have the system control machine distribute them. You make changes only to the copy stored on the system control machine, as directed by the instructions in this document. The United States edition of AFS is available to cells in the United States and Canada and to selected institutions in other countries, as determined by United States government regulations.

Cells that run the international version of AFS do not use the system control machine to distribute system configuration files. Some of the files contain information that is too sensitive to cross the network unencrypted, and United States government regulations forbid the export of the necessary encryption routines in the form that the Update Server uses. You must instead update the configuration files on each file server machine individually. The **bos** commands that you use to update the files encrypt the information using an exportable form of the encryption routines.

For a list of the configuration files stored in the **/usr/afs/etc** directory, see "Common Configuration Files in the /usr/afs/etc Directory" on page 62.

The *IBM AFS Quick Beginnings* configures a cell's first server machine as the system control machine. If you wish, you can reassign the role to a different machine that you install later, but you must then change the client portion of the Update Server (**upclientetc**) process running on all other server machines to refer to the new system control machine.

The following processes define the system control machine:

- The server portion of the Update Server (**upserver**) process, in cells using the United States edition of AFS. The client portion of the Update Server (**upclientetc** process) runs on the other server machines and references the system control machine.
- The NTP coordinator (**runntp** process) which points to a time source outside the cell, if the cell uses NTPD to synchronize its clocks. The **runntp** process on other machines reference the system control machine as their main time source.

The system control machine can also run the processes that define a simple file server machine, as listed in "Simple File Server Machines" on page 68. It can also server as a database server machine, and by convention acts as the binary distribution machine for its system type. A single **upserver** process can distribute both configuration files and binaries. See "Database Server Machines" on page 69 and "Binary Distribution Machines" on page 70.

To locate database server machines

1. Issue the **bos listhosts** command.

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

   The machines listed in the output are the cell's database server machines. For complete instructions and example output, see "To display a cell's database server machines" on page 89.

2. **(Optional)** Issue the **bos status** command to verify that a machine listed in the output of the **bos listhosts** command is actually running the processes that define it as a database server machine. For complete instructions, see "Displaying Process Status and Information from the BosConfig File" on page 113.

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

   If the specified machine is a database server machine, the output from the **bos status** command includes the following lines:

```
Instance buserver, currently running normally.
Instance kaserver, currently running normally.
Instance ptserver, currently running normally.
Instance vlserver, currently running normally.
```

To locate the system control machine

1. Issue the **bos status** command for any server machine. Complete instructions appear in "Displaying Process Status and Information from the BosConfig File" on page 113.

   ```
   % bos status <machine name> upserver upclientbin upclientetc -long
   ```

   The output you see depends on the machine you have contacted: a simple file server machine, the system control machine, or a binary distribution machine. See "Interpreting the Output from the bos status Command" on page 72.

To locate the binary distribution machine for a system type

1. Issue the **bos status** command for a file server machine of the system type you are checking (to determine a machine's system type, issue the **fs sysname** or **sys** command as described in "Displaying and Setting the System Type Name" on page 383. Complete instructions for the **bos status** command appear in "Displaying Process Status and Information from the BosConfig File" on page 113.

   ```
   % bos status <machine name> upserver upclientbin upclientetc -long
   ```

   The output you see depends on the machine you have contacted: a simple file server machine, the system control machine, or a binary distribution machine. See "Interpreting the Output from the bos status Command" on page 72.

Interpreting the Output from the bos status Command

Interpreting the output of the **bos status** command is most straightforward for a simple file server machine. There is no **upserver** process, so the output includes the following message:

```
bos: failed to get instance info for 'upserver' (no such entity)
```

A simple file server machine runs the **upclientbin** process, so the output includes a message like the following. It indicates that **fs7.abc.com** is the binary distribution machine for this system type.

```
Instance upclientbin, (type is simple) currently running normally.
Process last started at Wed Mar 10  23:37:09 1999 (1 proc start)
Command 1 is '/usr/afs/bin/upclient fs7.abc.com -t 60 /usr/afs/bin'
```

If you run the United States edition of AFS, a simple file server machine also runs the **upclientetc** process, so the output includes a message like the following. It indicates that **fs1.abc.com** is the system control machine.

```
Instance upclientetc, (type is simple) currently running normally.
Process last started at Mon Mar 22  05:23:49 1999 (1 proc start)
Command 1 is '/usr/afs/bin/upclient fs1.abc.com -t 60 /usr/afs/etc'
```

The Output on the System Control Machine

If you run the United States edition of AFS and have issued the **bos status** command for the system control machine, the output includes an entry for the **upserver** process similar to the following:

```
Instance upserver, (type is simple) currently running normally.
Process last started at Mon Mar 22 05:23:54 1999 (1 proc start)
Command 1 is '/usr/afs/bin/upserver'
```

If you are using the default configuration recommended in the *IBM AFS Quick Beginnings*, the system control machine is also the binary distribution machine for its system type, and a single **upserver** process distributes both kinds of updates. In that case, the output includes the following messages:

```
bos: failed to get instance info for 'upclientbin' (no such entity)
bos: failed to get instance info for 'upclientetc' (no such entity)
```

If the system control machine is not a binary distribution machine, the output includes an error message for the **upclientetc** process, but a complete a listing for the **upclientbin** process (in this case it refers to the machine **fs5.abc.com** as the binary distribution machine):

```
Instance upclientbin, (type is simple) currently running normally.
Process last started at Mon Mar 22  05:23:49 1999 (1 proc start)
Command 1 is '/usr/afs/bin/upclient fs5.abc.com -t 60 /usr/afs/bin'
bos: failed to get instance info for 'upclientetc' (no such entity)
```

The Output on a Binary Distribution Machine

If you have issued the **bos status** command for a binary distribution machine, the output includes an entry for the **upserver** process similar to the following and error message for the **upclientbin** process:

```
Instance upserver, (type is simple) currently running normally.
Process last started at Mon Apr 5 05:23:54 1999 (1 proc start)
Command 1 is '/usr/afs/bin/upserver'
bos: failed to get instance info for 'upclientbin' (no such entity)
```

Unless this machine also happens to be the system control machine, a message like the following references the system control machine (in this case, **fs3.abc.com**):

```
Instance upclientetc, (type is simple) currently running normally.
Process last started at Mon Apr 5 05:23:49 1999 (1 proc start)
Command 1 is '/usr/afs/bin/upclient fs3.abc.com -t 60 /usr/afs/etc'
```

Administering Database Server Machines

This section explains how to administer database server machines. For installation instructions, see the *IBM AFS Quick Beginnings*.

Replicating the AFS Administrative Databases

There are several benefits to replicating the AFS administrative databases (the Authentication, Backup, Protection, and Volume Location Databases), as discussed in "Replicating the AFS Administrative Databases" on page 31. For correct cell functioning, the copies of each database must be identical at all times. To keep the databases synchronized, AFS uses library of utilities called *Ubik*. Each database server process runs an associated lightweight Ubik process, and client-side programs call Ubik's client-side subroutines when they submit requests to read and change the databases.

Ubik is designed to work with minimal administrator intervention, but there are several configuration requirements, as detailed in "Configuring the Cell for Proper Ubik Operation" on page 74. The following brief overview of Ubik's operation is helpful for understanding the requirements. For more details, see "How Ubik Operates Automatically" on page 75.

Ubik is designed to distribute changes made in an AFS administrative database to all copies as quickly as possible. Only one copy of the database, the *synchronization site*, accepts change requests from clients; the lightweight Ubik process running there is the *Ubik coordinator*. To maintain maximum availability, there is a separate Ubik coordinator for each database, and the synchronization site for each of the four databases can be on a different machine. The synchronization site for a database can also move from machine to machine in response to process, machine, or network outages.

The other copies of a database, and the Ubik processes that maintain them, are termed *secondary*. The secondary sites do not accept database changes directly from client-side programs, but only from the synchronization site.

After the Ubik coordinator records a change in its copy of a database, it immediately sends the change to the secondary sites. During the brief distribution period, clients cannot access any of the copies of the database, even for reading. If the coordinator cannot reach a majority of the secondary sites, it halts the distribution and informs the client that the attempted change failed.

To avoid distribution failures, the Ubik processes maintain constant contact by exchanging time-stamped messages. As long as a majority of the secondary sites respond to the coordinator's messages, there is a *quorum* of sites that are synchronized with the coordinator. If a process, machine, or network outage breaks the quorum, the Ubik processes attempt to elect a new coordinator in order to establish a new quorum among the highest possible number of sites. See "A Flexible Coordinator Boosts Availability" on page 77.

Configuring the Cell for Proper Ubik Operation

This section describes how to configure your cell to maintain proper Ubik operation.

- Run all four database server processes--Authentication Server, Backup Server, Protection Server, and VL Server--on all database server machines.

  Both the client and server portions of Ubik expect that all the database server machines listed in the **CellServDB** file are running all of the database server processes. There is no mechanism for indicating that only some database server processes are running on a machine.

- Maintain correct information in the **/usr/afs/etc/CellServDB** file at all times.

  Ubik consults the **/usr/afs/etc/CellServDB** file to determine the sites with which to establish and maintain a quorum. Incorrect information can result in unsynchronized databases or election of a coordinator in each of several subgroups of machines, because the Ubik processes on various machines do not agree on which machines need to participate in the quorum.

  If you run the United States version of AFS and use the Update Server, it is simplest to maintain the **/usr/afs/etc/CellServDB** file on the system control machine, which distributes its copy to all other server machines. The *IBM AFS Quick Beginnings* explains how to configure the Update Server. If you run the international version of AFS, you must update the file on each machine individually.

  The only reason to alter the file is when configuring or decommissioning a database server machine. Use the appropriate **bos** commands rather than editing the file by hand. For instructions, see "Maintaining the Server CellServDB File" on page 88. The instructions in "Monitoring and Controlling Server Processes" on page 105 for stopping and starting processes remind you to alter the **CellServDB** file when appropriate, as do the instructions in the *IBM AFS Quick Beginnings* for installing or decommissioning a database server machine.

  (Client processes and the server processes that do not maintain databases also rely on correct information in the **CellServDB** file for proper operation, but their use of the information does not affect Ubik's operation. See "Maintaining the Server CellServDB File" on page 88 and "Maintaining Knowledge of Database Server Machines" on page 364.)

- Keep the clocks synchronized on all machines in the cell, especially the database server machines.

  In the conventional configuration specified in the *IBM AFS Quick Beginnings*, you run the **runntp** process to supervise the local Network Time Protocol Daemon (NTPD) on every AFS server machine. The NTPD on the system control machine synchronizes its clock with a reliable source outside the cell and broadcasts the time to the NTPDs on the other server machines. You can choose to run a different time synchronization protocol if you wish.

  Keeping clocks synchronized is important because the Ubik processes at a database's sites timestamp the messages which they exchange to maintain constant contact. Timestamping the messages is necessary because in a networked environment it is not safe to assume that a message reaches its destination instantly. Ubik compares the timestamp on an incoming message with the current time. If the difference is too great, it is possible that an outage is preventing reliable communication between the Ubik sites, which can possibly result in unsynchronized databases. Ubik considers the message invalid, which can prompt it to attempt election of a different coordinator.

  Electing a new coordinator is appropriate if a timestamped message is expired due to actual interruption of communication, but not if a message appears expired only because the sender and recipient do not share the same time. For detailed examples of how unsynchronized clocks can destabilize Ubik operation, see "How Ubik Uses Timestamped Messages" on page 76.

How Ubik Operates Automatically

The following Ubik features help keep its maintenance requirements to a minimum:

- Ubik's server and client portions operate automatically.

  Each database server process runs a lightweight process to call on the server portion of the Ubik library. It is common to refer to this lightweight process itself as Ubik. Because it is lightweight, the Ubik process does not appear in process listings such as those generated by the UNIX **ps** command. Client-side programs that need to read and change the databases directly call the subroutines in the Ubik library's client portion, rather than running a separate lightweight process. Examples of such programs are the **klog** command and the commands in the **pts** suite.

- Ubik tracks database version numbers.

  As the coordinator records a change to a database, it increments the database's version number. The version number makes it easy for the coordinator to determine if a site has the most recent version or not. The version number speeds the return to normal functioning after election of a new coordinator or when communication is restored after an outage, because it makes it easy to determine which site has the most current database and which need to be updated.

- Ubik's use of timestamped messages guarantees that database copies are always synchronized during normal operation.

  Replicating a database to increase data availability is pointless if all copies of the database are not the same. Inconsistent performance can result if clients receive different information depending on which copy of the database they access. As previously noted, Ubik sites constantly track the status of their peers by exchanging timestamped messages. For a detailed description, see "How Ubik Uses Timestamped Messages" on page 76.

- The ability to move the coordinator maximizes database availability.

  Suppose, for example, that in a cell with three database server machines a network partition separates the two secondary sites from the coordinator. The coordinator retires because it is no longer in contact with a majority of the sites listed in the **CellServDB** file. The two sites on the other side of the partition can elect a new coordinator among themselves, and it can then accept database changes from clients. If the coordinator cannot move in this way, the database has to be read-only until the network partition is repaired. For a detailed description of Ubik's election procedure, see "A Flexible Coordinator Boosts Availability" on page 77.

How Ubik Uses Timestamped Messages

Ubik synchronizes the copies of a database by maintaining constant contact between the synchronization site and the secondary sites. The Ubik coordinator frequently sends a time-stamped *guarantee* message to each of the secondary sites. When the secondary site receives the message, it concludes that it is in contact with the coordinator. It considers its copy of the database to be valid until time *T*, which is usually 60 seconds from the time the coordinator sent the message. In response, the secondary site returns a *vote* message that acknowledges the coordinator as valid until a certain time X, which is usually 120 seconds in the future.

The coordinator sends guarantee messages more frequently than every *T* seconds, so that the expiration periods overlap. There is no danger of expiration unless a network partition or other outage actually interrupts communication. If the guarantee expires, the secondary site's copy of the database it not necessarily current. Nonetheless, the database server continues to service client requests. It is considered better for overall cell functioning that a secondary site remains accessible even if the information it is distributing is possibly out of date. Most of the AFS administrative databases do not change that frequently, in any case, and making a database inaccessible causes a timeout for clients that happen to access that copy.

As previously mentioned, Ubik's use of timestamped messages makes it vital to synchronize the clocks on database server machines. There are two ways that skewed clocks can interrupt normal Ubik functioning, depending on which clock is ahead of the others.

Suppose, for example, that the Ubik coordinator's clock is ahead of the secondary sites: the coordinator's clock says 9:35:30, but the secondary clocks say 9:31:30. The secondary sites send votes messages that acknowledge the coordinator as valid until 9:33:30. This is two minutes in the future according to the secondary clocks, but is already in the past from the coordinator's perspective. The coordinator concludes that it no longer has enough support to remain coordinator and forces election of a new coordinator. Election takes about three minutes, during which time no copy of the database accepts changes.

The opposite possibility is that a secondary site's clock (14:50:00) is ahead of the coordinator's (14:46:30). When the coordinator sends a guarantee message good until 14:47:30), it has already expired according to the secondary clock. Believing that it is out of contact with the coordinator, the secondary site stops sending votes for the coordinator and tries get itself elected as coordinator. This is appropriate if the coordinator has actually failed, but is inappropriate when there is no actual outage.

The attempt of a single secondary site to get elected as the new coordinator usually does not affect the performance of the other sites. As long as their clocks agree with the coordinator's, they ignore the other secondary site's request for votes and continue voting for the current coordinator. However, if enough of the secondary sites's clocks get ahead of the coordinator's, they can force election of a new coordinator even though the current one is actually working fine.

A Flexible Coordinator Boosts Availability

Ubik uses timestamped messages to determine when coordinator election is necessary, just as it does to keep the database copies synchronized. As long as the coordinator receives vote messages from a majority of the sites (it implicitly votes for itself), it is appropriate for it to continue as coordinator because it is successfully distributing database changes. A majority is defined as more than 50% of all database sites when there are an odd number of sites; with an even number of sites, the site with the lowest Internet address has an extra vote for breaking ties as necessary.If the coordinator is not receiving sufficient votes, it retires and the Ubik sites elect a new coordinator. This does not happen spontaneously, but only when the coordinator really fails or stops receiving a majority of the votes. The secondary sites have a built-in bias to continue voting for an existing coordinator, which prevents undue elections.

The election of the new coordinator is by majority vote. The Ubik subprocesses have a bias to vote for the site with the lowest Internet address, which helps it gather the necessary majority quicker than if all the sites were competing to receive votes themselves. During the election (which normally lasts less than three minutes), clients can read information from the database, but cannot make any changes.

Ubik's election procedure makes it possible for each database server process's coordinator to be on a different machine. For example, if the Ubik coordinators for all four processes start out on machine A and the Protection Server on machine A fails for some reason, then a different site (say machine B) must be elected as the new Protection Database Ubik coordinator. Machine B remains the coordinator for the Protection Database even after the Protection Server on machine A is working again. The failure of the Protection Server has no effect on the Authentication, Backup, or VL Servers, so their coordinators remain on machine A.

Backing Up and Restoring the Administrative Databases

The AFS administrative databases store information that is critical for AFS operation in your cell. If a database becomes corrupted due to a hardware failure or other problem on a database server machine, it likely to be difficult and time-consuming to recreate all of the information from scratch. To protect yourself against loss of data, back up the administrative databases to a permanent media, such as tape, on a regular basis. The recommended method is to use a standard local disk backup utility such as the UNIX **tar** command.

When deciding how often to back up a database, consider the amount of data that you are willing to recreate by hand if it becomes necessary to restore the database from a backup copy. In most cells, the databases differ quite a bit in how often and how much they change. Changes to the Authentication Database are probably the least frequent, and consist mostly of changed user passwords. Protection Database and VLDB changes are probably more frequent, as users add or delete groups and change group memberships, and as you and other administrators create or move volumes. The number and frequency of changes is probably greatest in the Backup Database, particularly if you perform backups every day.

The ease with which you can recapture lost changes also differs for the different databases:

- If regular users make a large proportion of the changes to the Authentication Database and Protection Database in your cell, then recovering them possibly requires a large amount of detective work and interviewing of users, assuming that they can even remember what changes they made at what time.
- Recovering lost changes to the VLDB is more straightforward, because you can use the **vos syncserv** and **vos syncvldb** commands to correct any discrepancies between the VLDB and the actual state of volumes on server machines. Running these commands can be time-consuming, however.
- The configuration information in the Backup Database (Tape Coordinator port offsets, volume sets and entries, the dump hierarchy, and so on) probably does not change that often, in which case it is not that hard to recover a few recent changes. In contrast, there are likely to be a large number of new dump records resulting from dump operations. You can recover these records by using the **-dbadd** argument to the **backup scantape** command, reading in information from the backup tapes themselves. This can take a long time and require numerous tape changes, however, depending on how much data you back up in your cell and how you append dumps. Furthermore, the **backup scantape** command is subject to several restrictions. The most basic is that it halts if it finds that an existing dump record in the database has the same dump ID number as a dump on the tape it is scanning. If you want to continue with the scanning operation, you must locate and remove the existing record from the database. For

further discussion, see the **backup scantape** command's reference page in the *IBM AFS Administration Reference*.

These differences between the databases possibly suggest backing up the database at different frequencies, ranging from every few days or weekly for the Backup Database to every few weeks for the Authentication Database. On the other hand, it is probably simpler from a logistical standpoint to back them all up at the same time (and frequently), particularly if tape consumption is not a major concern. Also, it is not generally necessary to keep backup copies of the databases for a long time, so you can recycle the tapes fairly frequently.

To back up the administrative databases

1. Log in as the local superuser **root** on a database server machine that is not the synchronization site. The machine with the highest IP address is normally the best choice, since it is least likely to become the synchronization site in an election.
2. Issue the **bos shutdown** command to shut down the relevant server process on the local machine. For a complete description of the command, see "To stop processes temporarily" on page 122.

   For the **-instance** argument, specify one or more database server process names (**buserver** for the Backup Server, **kaserver** for the Authentication Server, **ptserver** for the Protection Server, or **vlserver** for the Volume Location Server. Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens.

   ```
   # bos shutdown <machine name> -instance <instances>+ -localauth [-wait]
   ```

3. Use a local disk backup utility, such as the UNIX **tar** command, to transfer one or more database files to tape. If the local database server machine does not have a tape device attached, use a remote copy command to transfer the file to a machine with a tape device, then use the **tar** command there.

   The following command sequence backs up the complete contents of the **/usr/afs/db** directory

   ```
   # cd /usr/afs/db
   # tar cvf  tape_device .
   ```

   To back up individual database files, substitute their names for the period in the preceding **tar** command:

   - **bdb.DB0** for the Backup Database
   - **kaserver.DB0** for the Authentication Database
   - **prdb.DB0** for the Protection Database
   - **vldb.DB0** for the VLDB

4. Issue the **bos start** command to restart the server processes on the local machine. For a complete description of the command, see "To start processes by changing their status flags to Run" on page

121. Provide the same values for the **-instance** argument as in Step "2" on page 79, and the **-localauth** flag for the same reason.

```
# bos start <machine name> -instance <server process name>+ -localauth
```

To restore an administrative database

1. Log in as the local superuser **root** on each database server machine in the cell.
2. Working on one of the machines, issue the **bos shutdown** command once for each database server machine, to shut down the relevant server process on all of them. For a complete description of the command, see "To stop processes temporarily" on page 122.

   For the **-instance** argument, specify one or more database server process names (**buserver** for the Backup Server, **kaserver** for the Authentication Server, **ptserver** for the Protection Server, or **vlserver** for the Volume Location Server. Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens.

   ```
   # bos shutdown <machine name> -instance <instances>+ -localauth [-wait]
   ```

3. Remove the database from each database server machine, by issuing the following commands on each one.

   ```
   # cd /usr/afs/db
   ```

   For the Backup Database:

   ```
   # rm bdb.DB0
   # rm bdb.DBSYS1
   ```

   For the Authentication Database:

   ```
   # rm kaserver.DB0
   # rm kaserver.DBSYS1
   ```

   For the Protection Database:

   ```
   # rm prdb.DB0
   # rm prdb.DBSYS1
   ```

   For the VLDB:

   ```
   # rm vldb.DB0
   # rm vldb.DBSYS1
   ```

4. Using the local disk backup utility that you used to back up the database, copy the most recently backed-up version of it to the appropriate file on the database server machine with the lowest IP address. The following is an appropriate **tar** command if the synchronization site has a tape device attached:

   ```
   # cd /usr/afs/db
   # tar xvf tape_device  database_file
   ```

   where *database_file* is one of the following:

- **bdb.DB0** for the Backup Database
- **kaserver.DB0** for the Authentication Database
- **prdb.DB0** for the Protection Database
- **vldb.DB0** for the VLDB

5. Working on one of the machines, issue the **bos start** command to restart the server process on each of the database server machines in turn. Start with the machine with the lowest IP address, which becomes the synchronization site for the Backup Database. Wait for it to establish itself as the synchronization site before repeating the command to restart the process on the other database server machines. For a complete description of the command, see "To start processes by changing their status flags to Run" on page 121. Provide the same values for the **-instance** argument as in Step "2" on page 80, and the **-localauth** flag for the same reason.

   ```
   # bos start <machine name> -instance  <server process name>+  -localauth
   ```

6. If the database has changed since you last backed it up, issue the appropriate commands from the instructions in the indicated sections to recreate the information in the restored database. If issuing **pts** commands, you must first obtain administrative tokens. The **backup** and **vos** commands accept the **-localauth** flag if you are logged in as the local superuser **root**, so you do not need administrative tokens. The Authentication Server always performs a separate authentication anyway, so you only need to include the **-admin** argument if issuing **kas** commands.

   - To define or remove volume sets and volume entries in the Backup Database, see "Defining and Displaying Volume Sets and Volume Entries" on page 209.
   - To edit the dump hierarchy in the Backup Database, see "Defining and Displaying the Dump Hierarchy" on page 215.
   - To define or remove Tape Coordinator port offset entries in the Backup Database, see "Configuring Tape Coordinator Machines and Tape Devices" on page 205.
   - To restore dump records in the Backup Database, see "To scan the contents of a tape" on page 272.
   - To recreate Authentication Database entries or password changes for users, see the appropriate section of "Administering User Accounts" on page 459.
   - To recreate Protection Database entries or group membership information, see the appropriate section of "Administering the Protection Database" on page 487.
   - To synchronize the VLDB with volume headers, see "Synchronizing the VLDB and Volume Headers" on page 168.

Installing Server Process Software

This section explains how to install new server process binaries on file server machines, how to revert to a previous version if the current version is not working properly, and how to install new disks to house AFS volumes on a file server machine.

The most frequent reason to replace a server process's binaries is to upgrade AFS to a new version. In general, installation instructions accompany the updated software, but this chapter provides an additional reference.

Each AFS server machine must store the server process binaries in a local disk directory, called **/usr/afs/bin** by convention. For predictable system performance, it is best that all server machines run the same build level, or at least the same version, of the server software. For instructions on checking AFS build level, see "Displaying A Binary File's Build Level" on page 87.

The Update Server makes it easy to distribute a consistent version of software to all server machines. You designate one server machine of each system type as the *binary distribution machine* by running the server portion of the Update Server (**upserver** process) on it. All other server machines of that system type run the client portion of the Update Server (**upclientbin** process) to retrieve updated software from the binary distribution machine. The *IBM AFS Quick Beginnings* explains how to install the appropriate processes. For more on binary distribution machines, see "Binary Distribution Machines" on page 70.

When you use the Update Server, you install new binaries on binary distribution machines only. If you install binaries directly on a machine that is running the **upclientbin** process, they are overwritten the next time the process compares the contents of the local **/usr/afs/bin** directory to the contents on the system control machine, by default within five minutes.

The following instructions explain how to use the appropriate commands from the **bos** suite to install and uninstall server binaries.

Installing New Binaries

An AFS server process does not automatically switch to a new process binary file as soon as it is installed in the **/usr/afs/bin** directory. The process continues to use the previous version of the binary file until it (the process) next restarts. By default, the BOS Server restarts processes for which there are new binary files every day at 5:00 a.m., as specified in the **/usr/afs/local/BosConfig** file. To display or change this *binary restart time*, use the **bos getrestart** and **bos setrestart** commands, as described in "Setting the BOS Server's Restart Times" on page 126.

You can force the server machine to start using new server process binaries immediately by issuing the **bos restart** command as described in the following instructions.

You do not need to restart processes when you install new command suite binaries. The new binary is invoked automatically the next time a command from the suite is issued.

When you use the **bos install** command, the BOS Server automatically saves the current version of a binary file by adding a **.BAK** extension to its name. It renames the current **.BAK** version, if any, to the **.OLD** version, if there is no **.OLD** version already. If there is a current **.OLD** version, the current **.BAK** version must be at least seven days old to replace it.

It is best to store AFS binaries in the **/usr/afs/bin** directory, because that is the only directory the BOS Server automatically checks for new binaries. You can, however, use the **bos install** command's **-dir** argument to install non-AFS binaries into other directories on a server machine's local disk. See the command's reference page in the *IBM AFS Administration Reference* for further information.

To install new server binaries

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Verify that the binaries are available in the source directory from which you are installing them. If the machine is also an AFS client, you can retrieve the binaries from a central directory in AFS. Otherwise, you can obtain them directly from the AFS distribution media, from a local disk directory where you previously installed them, or from a remote machine using a transfer utility such as the **ftp** command.

3. Issue the **bos install** command for the binary distribution machine. (If you have forgotten which machine is performing that role, see "To locate the binary distribution machine for a system type" on page 72.)

   ```
   % bos install <machine name> <files to install>+
   ```

   where

   i

   Is the shortest acceptable abbreviation of **install**.

   machine name

   Names the binary distribution machine.

   files to install

   Names each binary file to install into the local **/usr/afs/bin** directory. Partial pathnames are interpreted relative to the current working directory. The last element in each pathname (the filename itself) matches the name of the file it is replacing, such as **bosserver** or **volserver** for server processes, **bos** or **vos** for commands.

   Each AFS server process other than the **fs** process uses a single binary file. The **fs** process uses three binary files: **fileserver**, **volserver**, and **salvager**. Installing a new version of one component does not necessarily mean that you need to replace all three.

4. Repeat Step "3" on page 83 for each binary distribution machine.

5. **(Optional)** If you want to restart processes to use the new binaries immediately, wait until the **upclientbin** process retrieves them from the binary distribution machine. You can verify the timestamps on binary files by using the **bos getdate** command as described in "Displaying Binary Version Dates" on page 85. When the binary files are available on each server machine, issue the **bos restart** command, for which complete instructions appear in "Stopping and Immediately Restarting Processes" on page 124.

If you are working on an AFS client machine, it is a wise precaution to have a copy of the **bos** command suite binaries on the local disk before restarting server processes. In the conventional configuration, the **/usr/afsws/bin** directory that houses the **bos** command binary on client machines is a symbolic link into AFS, which conserves local disk space. However, restarting certain processes (particularly the database server processes) can make the AFS filespace inaccessible, particularly if a problem arises during the restart. Having a local copy of the **bos** binary enables you to uninstall or reinstall process binaries or restart processes even in this case. Use the **cp** command to copy the **bos** command binary from the **/usr/afsws/bin** directory to a local directory such as **/tmp**.

Restarting a process causes a service outage. It is best to perform the restart at times of low system usage if possible.

```
% bos restart <machine name> <instances>+
```

Reverting to the Previous Version of Binaries

In rare cases, installing a new binary can cause problems serious enough to require reverting to the previous version. Just as with installing binaries, consistent system performance requires reverting every server machine back to the same version. Issue the **bos uninstall** command described here on each binary distribution machine.

When you use the **bos uninstall** command, the BOS Server discards the current version of a binary file and promotes the **.BAK** version of the file by removing the extension. It renames the current **.OLD** version, if any, to **.BAK**.

If there is no current **.BAK** version, the **bos uninstall** command operation fails and generates an error message. If a **.OLD** version still exists, issue the **mv** command to rename it to **.BAK** before reissuing the **bos uninstall** command.

Just as when you install new binaries, the server processes do not start using a reverted version immediately. Presumably you are reverting because the current binaries do not work, so the following instructions have you restart the relevant processes.

To revert to the previous version of binaries

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Verify that the **.BAK** version of each relevant binary is available in the **/usr/afs/bin** directory on each binary distribution machine. If necessary, you can use the **bos getdate** command as described in "Displaying Binary Version Dates" on page 85. If necessary, rename the **.OLD** version to **.BAK**

3. Issue the **bos uninstall** command for a binary distribution machine. (If you have forgotten which machine is performing that role, see "To locate the binary distribution machine for a system type" on page 72.)

   ```
   % bos uninstall <machine name> <files to uninstall>+
   ```

where

u

Is the shortest acceptable abbreviation of **uninstall**.

machine name

Names the binary distribution machine.

files to uninstall

Names each binary file in the **/usr/afs/bin** directory to replace with its **.BAK** version. The file name alone is sufficient, because the **/usr/afs/bin** directory is assumed.

4. Repeat Step "3" on page 84 for each binary distribution machine.
5. Wait until the **upclientbin** process on each server machine retrieves the reverted from the binary distribution machine. You can verify the timestamps on binary files by using the **bos getdate** command as described in "Displaying Binary Version Dates" on page 85. When the binary files are available on each server machine, issue the **bos restart** command, for which complete instructions appear in "Stopping and Immediately Restarting Processes" on page 124.

   If you are working on an AFS client machine, it is a wise precaution to have a copy of the **bos** command suite binaries on the local disk before restarting server processes. In the conventional configuration, the **/usr/afsws/bin** directory that houses the **bos** command binary on client machines is a symbolic link into AFS, which conserves local disk space. However, restarting certain processes (particularly the database server processes) can make the AFS filespace inaccessible, particularly if a problem arises during the restart. Having a local copy of the **bos** binary enables you to uninstall or reinstall process binaries or restart processes even in this case. Use the **cp** command to copy the **bos** command binary from the **/usr/afsws/bin** directory to a local directory such as **/tmp**.

   ```
   % bos restart <machine name> <instances>+
   ```

Displaying Binary Version Dates

You can check the compilation dates for all three versions of a binary file in the **/usr/afs/bin** directory--the current, **.BAK** and **.OLD** versions. This is useful for verifying that new binaries have been copied to a file server machine from its binary distribution machine before restarting a server process to use the new binaries.

To check dates on binaries in a directory other than **/usr/afs/bin**, add the **-dir** argument. See the *IBM AFS Administration Reference*.

To display binary version dates

1. Issue the **bos getdate** command.

   ```
   % bos getdate <machine name> <files to check>+
   ```

   where

   getd

   Is the shortest acceptable abbreviation of **getdate**.

   machine name

   Name the file server machine for which to display binary dates.

   files to check

   Names each binary file to display.

Removing Obsolete Binary Files

When processes with new binaries have been running without problems for a number of days, it is generally safe to remove the **.BAK** and **.OLD** versions from the **/usr/afs/bin** directory, both to reduce clutter and to free space on the file server machine's local disk.

You can use the **bos prune** command's flags to remove the following types of files:

- To remove files in the **/usr/afs/bin** directory with a **.BAK** extension, use the **-bak** flag.
- To remove files in the **/usr/afs/bin** directory with a **.OLD** extension, use the **-old** flag.
- To remove files in the **/usr/afs/logs** directory called **core**, with any extension, use the **-core** flag.
- To remove all three types of files, use the **-all** flag.

To remove obsolete binaries

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos prune** command with one or more of its flags.

   ```
   % bos prune <machine name> [-bak] [-old] [-core] [-all]
   ```

   where

p

Is the shortest acceptable abbreviation of **prune**.

machine name

Names the file server machine on which to remove obsolete files.

-bak

Removes all the files with a **.BAK** extension from the **/usr/afs/bin** directory. Do not combine this flag with the **-all** flag.

-old

Removes all the files a .**OLD** extension from the **/usr/afs/bin** directory. Do not combine this flag with the **-all** flag.

-core

Removes all core files from the **/usr/afs/logs** directory. Do not combine this flag with the **-all** flag

-all

Combines the effect of the other three flags. Do not combine it with the other three flags.

Displaying A Binary File's Build Level

For the most consistent performance on a server machine, and cell-wide, it is best for all server processes to come from the same AFS distribution. Every AFS binary includes an ASCII string that specifies its version, or *build level*. To display it, use the **strings** and **grep** commands, which are included in most UNIX distributions.

To display an AFS binary's build level

1. Change to the directory that houses the binary file . If you are not sure where the binary resides, issue the **which** command.

   ```
   % which binary_file
   /bin_dir_path/binary_file
   % cd bin_dir_path
   ```

2. Issue the **strings** command to extract all ASCII strings from the binary file. Pipe the output to the **grep** command to locate the relevant line.

   ```
   % strings ./binary_file | grep Base
   ```

   The output reports the AFS build level in a format like the following:

```
@(#)Base configuration afsversion  build_level
```

For example, the following string indicates the binary is from AFS 3.6 build 3.0:

```
@(#)Base configuration afs3.6 3.0
```

Maintaining the Server CellServDB File

Every file server machine maintains a list of its home cell's database server machines in the local disk file **/usr/afs/etc/CellServDB** on its local disk. Both database server processes and non-database server processes consult the file:

- The database server processes (the Authentication, Backup, Protection, and Volume Location Servers) maintain constant contact with their peers in order to keep their copies of the replicated administrative databases synchronized.

  As detailed in "Replicating the AFS Administrative Databases" on page 74, the database server processes use the Ubik utility to synchronize the information in the databases they maintain. The Ubik coordinator at the synchronization site for each database maintains the single read/write copy of the database and distributes changes to the secondary sites as necessary. It must maintain contact with a majority of the secondary sites to remain the coordinator, and consults the **CellServDB** file to learn how many peers it has and on which machines they are running.

  If the coordinator loses contact with the majority of its peers, they all cooperate to elect a new coordinator by majority vote. During the election, all of the Ubik processes consult the **CellServDB** file to learn where to send their votes, and what number constitutes a majority.

- The non-database server processes must know which machines are running the database server processes in order to retrieve information from the databases. For example, the first time that a user accesses an AFS file, the File Server that houses it contacts the Protection Server to obtain a list of the user's group memberships (the list is called a current protection subgroup, or CPS). The File Server uses the CPS as it determines if the access control list (ACL) protecting the file grants the required permissions to the user (for more details, see "About the Protection Database" on page 487).

The consequences of missing or incorrect information in the **CellServDB** file are as follows:

- If the file does not list a machine, then it is effectively not a database server machine even if the database server processes are running. The Ubik coordinator does not send it database updates or include it in the count that establishes a majority. It does not participate in Ubik elections, and so refuses to distribute database information to any client machines that happen to contact it (which they can do if their **/usr/vice/etc/CellServDB** file lists it). Users of the client machine must wait for a timeout before they can contact a correctly functioning database server machine.
- If the file lists a machine that is not running the database server processes, the consequences can be serious. The Ubik coordinator cannot send it database updates, but includes it in the count that establishes a majority. If valid secondary sites go down and stop sending their votes to the coordinator,

it can wrongly appear that the coordinator no longer has the majority it needs. The resulting election of a new coordinator causes a service outage during which information from the database becomes unavailable. Furthermore, the lack of a vote from the incorrectly listed site can disturb the election, if it makes the other sites believe that a majority of sites are not voting for the new coordinator.

A more minor consequence is that non-database server processes attempt to contact the database server processes on the machine. They experience a timeout delay because the processes are not running.

Note that the **/usr/afs/etc/CellServDB** file on a server machine is not the same as the **/usr/vice/etc/CellServDB** file on client machine. The client version includes entries for foreign cells as well as the local cell. However, it is important to update both versions of the file whenever you change your cell's database server machines. A server machine that is also a client needs to have both files, and you need to update them both. For more information on maintaining the client version of the **CellServDB** file, see "Maintaining Knowledge of Database Server Machines" on page 364.

Distributing the Server CellServDB File

To avoid the negative consequences of incorrect information in the **/usr/afs/etc/CellServDB** file, you must update it on all of your cell's server machines every time you add or remove a database server machine. The *IBM AFS Quick Beginnings* provides complete instructions for installing or removing a database server machine and for updating the **CellServDB** file in that context. This section explains how to distribute the file to your server machines and how to make other cells aware of the changes if you participate in the AFS global name space.

If you use the United States edition of AFS, use the Update Server to distribute the central copy of the server **CellServDB** file stored on the cell's system control machine. If you use the international edition of AFS, instead change the file on each server machine individually. For further discussion of the system control machine and why international cells must not use it for files in the **/usr/afs/etc** directory, see "The System Control Machine" on page 70. For instructions on configuring the Update Server when using the United States version of AFS, see the *IBM AFS Quick Beginnings*.

To avoid formatting errors that can cause errors, always use the **bos addhost** and **bos removehost** commands, rather than editing the file directly. You must also restart the database server processes running on the machine, to initiate a coordinator election among the new set of database server machines. This step is included in the instructions that appear in "To add a database server machine to the CellServDB file" on page 90 and "To remove a database server machine from the CellServDB file" on page 91. For instructions on displaying the contents of the file, see "To display a cell's database server machines" on page 89.

If you make your cell accessible to foreign users as part of the AFS global name space, you also need to inform other cells when you change your cell's database server machines. The AFS Support group maintains a **CellServDB** file that lists all cells that participate in the AFS global name space, and can change your cell's entry at your request. For further details, see "Making Your Cell Visible to Others" on page 22.

Another way to advertise your cell's database server machines is to maintain a copy of the file at the conventional location in your AFS filespace, **/afs/***cellname***/service/etc/CellServDB.local**. For further discussion, see "The Third Level" on page 25.

To display a cell's database server machines

1. Issue the **bos listhosts** command. If you have maintained the file properly, the output is the same on every server machine, but the *machine name* argument enables you to check various machines if you wish.

   ```
   % bos listhosts <machine name> [<cell name>]
   ```

   where

   listh

   Is the shortest acceptable abbreviation of **listhosts**.

   machine name

   Specifies the server machine from which to display the **/usr/afs/etc/CellServDB** file.

   cell name

   Specifies the complete Internet domain name of a foreign cell. You must already know the name of at least one server machine in the cell, to provide as the **machine name** argument.

The output lists the machines in the order they appear in the **CellServDB** file on the specified server machine. It assigns each one a `Host` index number, as in the following example. There is no implied relationship between the index and a machine's IP address, name, or role as Ubik coordinator or secondary site.

```
% bos listhosts fs1.abc.com
Cell name is abc.com
    Host 1 is fs1.abc.com
    Host 2 is fs7.abc.com
    Host 3 is fs4.abc.com
```

The output lists machines by name rather than IP address as long as the naming service (such as the Domain Name Service or local host table) is functioning properly. To display IP addresses, login to a server machine as the local superuser **root** and use a text editor or display command, such as the **cat** command, to view the **/usr/afs/etc/CellServDB** file.

To add a database server machine to the CellServDB file

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos addhost** command to add each new database server machine to the **CellServDB** file. If you use the United States edition of AFS, specify the system control machine as *machine name*. (If

you have forgotten which machine is the system control machine, see "The Output on the System Control Machine" on page 73.) If you use the international edition of AFS, repeat the command on each or your cell's server machines in turn by substituting its name for *machine name*.

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

where

addh

Is the shortest acceptable abbreviation of **addhost**.

machine name

Names the system control machine, if you are using the United States edition of AFS. If you are using the international edition of AFS, it names each of your server machines in turn.

host name

Specifies the fully qualified hostname of each database server machine to add to the **CellServDB** file (for example: **fs4.abc.com**). The BOS Server uses the **gethostbyname()** routine to obtain each machine's IP address and records both the name and address automatically.

3. Restart the Authentication Server, Backup Server, Protection Server, and VL Server on every database server machine, so that the new set of machines participate in the election of a new Ubik coordinator. The instruction uses the conventional names for the processes; make the appropriate substitution if you use different process names. For complete syntax, see "Stopping and Immediately Restarting Processes" on page 124.

   Important: Repeat the following command in quick succession on all of the database server machines.

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

4. Edit the **/usr/vice/etc/CellServDB** file on each of your cell's client machines. For instructions, see "Maintaining Knowledge of Database Server Machines" on page 364.
5. If you participate in the AFS global name space, please have one of your cell's designated site contacts register the changes you have made with the AFS Product Support group.

   If you maintain a central copy of your cell's server **CellServDB** file in the conventional location (**/afs/***cellname***/service/etc/CellServDB.local**), edit the file to reflect the change.

To remove a database server machine from the CellServDB file

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos removehost** command to remove each database server machine from the **CellServDB** file. If you use the United States edition of AFS, specify the system control machine as *machine name*. (If you have forgotten which machine is the system control machine, see "The Output on the System Control Machine" on page 73.) If you use the international edition of AFS, repeat the command on each or your cell's server machines in turn by substituting its name for *machine name*.

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

   where

   removeh

   Is the shortest acceptable abbreviation of **removehost**.

   machine name

   Names the system control machine, if you are using the United States edition of AFS. If you are using the international edition of AFS, it names each of your server machines in turn.

   host name

   Specifies the fully qualified hostname of each database server machine to remove from the **CellServDB** file (for example: **fs4.abc.com**).

3. Restart the Authentication Server, Backup Server, Protection Server, and VL Server on every database server machine, so that the new set of machines participate in the election of a new Ubik coordinator. The instruction uses the conventional names for the processes; make the appropriate substitution if you use different process names. For complete syntax, see "Stopping and Immediately Restarting Processes" on page 124.

   Important: Repeat the following command in quick succession on all of the database server machines.

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

4. Edit the **/usr/vice/etc/CellServDB** file on each of your cell's client machines. For instructions, see "Maintaining Knowledge of Database Server Machines" on page 364.

5. If you participate in the AFS global name space, please have one of your cell's designated site contacts register the changes you have made with the AFS Product Support group.

   If you maintain a central copy of your cell's server **CellServDB** file in the conventional location (**/afs/***cellname***/service/etc/CellServDB.local**), edit the file to reflect the change.

Managing Authentication and Authorization Requirements

This section describes how the AFS server processes guarantee that only properly authorized users perform privileged commands, by checking authorization checking and mutually authenticating with their clients. It explains how you can control authorization checking requirements on a per-machine or per-cell basis, and how to bypass mutual authentication when issuing commands.

Authentication versus Authorization

Many AFS commands are *privileged* in that the AFS server process invoked by the command performs it only for a properly authorized user. The server process performs the following two tests to determine if someone is properly authorized:

- In the *authentication* test, the server process mutually authenticates with the command interpreter, Cache Manager, or other client process that is acting on behalf of a user or application. The goal of this test is to determine who is issuing the command. The server process verifies that the issuer really is who he or she claims to be, by examining the server ticket and other components of the issuer's token. (Secondarily, it allows the client process to verify that the server process is genuine.) If the issuer has no token or otherwise fails the test, the server process assigns him or her the identity **anonymous**, a completely unprivileged user. For a more complete description of mutual authentication, see "A More Detailed Look at Mutual Authentication" on page 51.

  Many individual commands enable you to bypass the authentication test by assuming the **anonymous** identity without even attempting to mutually authenticate. Note, however, that this is futile if the command is privileged and the server process is still performing the *authorization* test, because in that case the process refuses to execute privileged commands for the **anonymous** user.

- In the authorization test, the server process determines if the issuer is authorized to use the command by consulting a list of privileged users. The goal of this test is to determine what the issuer is allowed to do. Different server processes consult different lists of users, as described in "Managing Administrative Privilege" on page 531. The server process refuses to execute any privileged command for an unauthorized issuer. If a command has no privilege requirements, the server process skips this step and executes and immediately.

  Note: Never place the **anonymous** user or the **system:anyuser** group on a privilege list; it makes authorization checking meaningless.

  You can use the **bos setauth** command to control whether the server processes on a server machine check for authorization. Other server machines are not affected. Keep in mind that turning off authorization checking is a grave security risk, because the server processes on that machine perform any action for any user.

Controlling Authorization Checking on a Server Machine

Disabling authorization checking is a serious breach of security because it means that the AFS server processes on a file server machine performs any action for any user, even the **anonymous** user.

The only time it is common to disable authorization checking is when installing a new file server machine (see the IBM AFS Quick Beginnings). It is necessary then because it is not possible to configure all of the necessary security mechanisms before performing other actions that normally make use of them. For greatest security, work at the console of the machine you are installing and enable authorization checking as soon as possible.

During normal operation, the only reason to disable authorization checking is if an error occurs with the server encryption keys, leaving the servers unable to authenticate users properly. For instructions on handling key-related emergencies, see "Handling Server Encryption Key Emergencies" on page 343.

You control authorization checking on each file server machine separately; turning it on or off on one machine does not affect the others. Because client machines generally choose a server process at random, it is hard to predict what authorization checking conditions prevail for a given command unless you make the requirement the same on all machines. To turn authorization checking on or off for the entire cell, you must repeat the appropriate command on every file server machine.

The server processes constantly monitor the directory **/usr/afs/local** on their local disks to determine if they need to check for authorization. If the file called **NoAuth** appears in that directory, then the servers do not check for authorization. When it is not present (the usual case), they perform authorization checking.

Control the presence of the **NoAuth** file through the BOS Server. When you disable authorization checking with the **bos setauth** command (or, during installation, by putting the **-noauth** flag on the command that starts up the BOS Server), the BOS Server creates the zero-length **NoAuth** file. When you reenable authorization checking, the BOS Server removes the file.

To disable authorization checking on a server machine

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos setauth** command to disable authorization checking.

   ```
   % bos setauth <machine name> off
   ```

   where

   seta

   Is the shortest acceptable abbreviation of **setauth**.

   machine name

   Specifies the file server machine on which server processes do not check for authorization.

To enable authorization checking on a server machine

1. Reenable authorization checking. (No privilege is required because the machine is not currently checking for authorization.) For detailed syntax information, see the preceding section.

   ```
   % bos setauth <machine name> on
   ```

Bypassing Mutual Authentication for an Individual Command

Several of the server processes allow any user (not just system administrators) to disable mutual authentication when issuing a command. The server process treats the issuer as the unauthenticated user **anonymous**.

The facilities for preventing mutual authentication are provided for use in emergencies (such as the key emergency discussed in "Handling Server Encryption Key Emergencies" on page 343). During normal circumstances, authorization checking is turned on, making it useless to prevent authentication: the server processes refuse to perform privileged commands for the user **anonymous**.

It can be useful to prevent authentication when authorization checking is turned off. The very act of trying to authenticate can cause problems if the server cannot understand a particular encryption key, as is likely to happen in a key emergency.

To bypass mutual authentication for bos, kas, pts, and vos commands

Provide the **-noauth** flag which is available on many of the commands in the suites. To verify that a command accepts the flag, issue the **help** command in its suite, or consult the command's reference page in the *IBM AFS Administration Reference* (the reference page also specifies the shortest acceptable abbreviation for the flag on each command). The suites' **apropos** and **help** commands do not themselves accept the flag.

You can bypass mutual authentication for all **kas** commands issued during an interactive session by including the **-noauth** flag on the **kas interactive** command. If you have already entered interactive mode with an authenticated identity, issue the **(kas) noauthentication** command to assume the **anonymous** identity.

To bypass mutual authentication for fs commands

This is not possible, except by issuing the **unlog** command to discard your tokens before issuing the **fs** command.

Adding or Removing Disks and Partitions

AFS makes it very easy to add storage space to your cell, just by adding disks to existing file server machines. This section explains how to install or remove a disk used to store AFS volumes. (Another way to add storage space is to install additional server machines, as instructed in the *IBM AFS Quick Beginnings*.)

Both adding and removing a disk cause at least a brief file system outage, because you must restart the **fs** process to have it recognize the new set of server partitions. Some operating systems require that you shut the machine off before adding or removing a disk, in which case you must shut down all of the AFS server processes first. Otherwise, the AFS-related aspects of adding or removing a disk are not complicated, so the duration of the outage depends mostly on how long it takes to install or remove the disk itself.

The following instructions for installing a new disk completely prepare it to house AFS volumes. You can then use the **vos create** command to create new volumes, or the **vos move** command to move existing ones from other partitions. For instructions, see "Creating Read/write Volumes" on page 135 and "Moving Volumes" on page 166. The instructions for removing a disk are basically the reverse of the installation instructions, but include extra steps that protect against data loss.

A server machines can house 256 AFS server partitions, each one mounted at a directory with a name of the form **/vicep***index*, where *index* is one or two lowercase letters. By convention, the first partition on a machine is mounted at **/vicepa**, the second at **/vicepb**, and so on to the twenty-sixth at **/vicepz**. Additional partitions are mounted at **/vicepaa** through **/vicepaz** and so on up to **/vicepiv**. Using the letters consecutively is not required, but is simpler.

Mount each **/vicep** directory directly under the local file system's root directory (**/**), not as a subdirectory of any other directory; for example, **/usr/vicepa** is not an acceptable location. You must also map the directory to the partition's device name in the file server machine's file systems registry file (**/etc/fstab** or equivalent).

These instructions assume that the machine's AFS initialization file includes the following command to restart the BOS Server after each reboot. The BOS Server starts the other AFS server processes listed in the local **/usr/afs/local/BosConfig** file. For information on the **bosserver** command's optional arguments, see its reference page in the *IBM AFS Administration Reference*.

```
/usr/afs/bin/bosserver &
```

To add and mount a new disk to house AFS volumes

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

   ```
   % su root
   Password: root_password
   ```

2. Decide how many AFS partitions to divide the new disk into and the names of the directories at which to mount them (the introduction to this section describes the naming conventions). To display the names of the existing server partitions on the machine, issue the **vos listpart** command. Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens.

   ```
   # vos listpart <machine name> -localauth
   ```

where

listp

Is the shortest acceptable abbreviation of **listpart**.

machine name

Names the local file server machine.

-localauth

Constructs a server ticket using a key from the local **/usr/afs/etc/KeyFile** file. The **bos** command interpreter presents it to the BOS Server during mutual authentication.

3. Create each directory at which to mount a partition.

   ```
   # mkdir /vicepx[x]
   ```

4. Using a text editor, create an entry in the machine's file systems registry file (**/etc/fstab** or equivalent) for each new disk partition, mapping its device name to the directory you created in the previous step. Refer to existing entries in the file to learn the proper format, which varies for different operating systems.
5. If the operating system requires that you shut off the machine to install a new disk, issue the **bos shutdown** command to shut down all AFS server processes other than the BOS Server (it terminates safely when you shut off the machine). Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens. For a complete description of the command, see "To stop processes temporarily" on page 122.

   ```
   # bos shutdown <machine name> -localauth [-wait]
   ```

6. If necessary, shut off the machine. Install and format the new disk according to the instructions provided by the disk and operating system vendors. If necessary, edit the disk's partition table to reflect the changes you made to the files system registry file in step "4" on page 97; consult the operating system documentation for instructions.
7. If you shut off the machine down in step "6" on page 97, turn it on. Otherwise, issue the **bos restart** command to restart the **fs** process, forcing it to recognize the new set of server partitions. Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens. For complete instructions for the **bos restart** command, see "Stopping and Immediately Restarting Processes" on page 124.

   ```
   # bos restart <machine name>  fs -localauth
   ```

8. Issue the **bos status** command to verify that all server processes are running correctly. For more detailed instructions, see "Displaying Process Status and Information from the BosConfig File" on page 113.

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

To unmount and remove a disk housing AFS volumes

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **vos listvol** command to list the volumes housed on each partition of each disk you are about to remove, in preparation for removing them or moving them to other partitions. For detailed instructions, see "Displaying Volume Headers" on page 159.

   ```
   % vos listvol <machine name> [<partition name>]
   ```

3. Move any volume you wish to retain in the file system to another partition. You can move only read/write volumes. For more detailed instructions, and for instructions on moving read-only and backup volumes, see "Moving Volumes" on page 166.

   ```
   % vos move  <volume name or ID>  \
        <machine name on source> <partition name on source>  \
        <machine name on destination> <partition name on destination>
   ```

4. **(Optional)** If there are any volumes you do not wish to retain, back them up using the **vos dump** command or the AFS Backup System. See "Dumping and Restoring Volumes" on page 183 or "Backing Up Data" on page 251, respectively.
5. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

   ```
   % su root
   Password: root_password
   ```

6. Issue the **umount** command, repeating it for each partition on the disk to be removed.

   ```
   # cd /
   # umount /dev/<partition_block_device_name>
   ```

7. Using a text editor, remove or comment out each partition's entry from the machine's file systems registry file (**/etc/fstab** or equivalent).
8. Remove the **/vicep** directory associated with each partition.

   ```
   # rmdir /vicepxx
   ```

9. If the operating system requires that you shut off the machine to remove a disk, issue the **bos shutdown** command to shut down all AFS server processes other than the BOS Server (it terminates safely when you shut off the machine). Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens. For a complete description of the command, see "To stop processes temporarily" on page 122.

   ```
   # bos shutdown <machine name> -localauth [-wait]
   ```

10. If necessary, shut off the machine. Remove the disk according to the instructions provided by the disk and operating system vendors. If necessary, edit the disk's partition table to reflect the changes you made to the files system registry file in step "7" on page 98; consult the operating system documentation for instructions.

11. If you shut off the machine down in step "10" on page 98, turn it on. Otherwise, issue the **bos restart** command to restart the **fs** process, forcing it to recognize the new set of server partitions. Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens. For complete instructions for the **bos restart** command, see "Stopping and Immediately Restarting Processes" on page 124.

    ```
    # bos restart <machine name>  fs -localauth
    ```

12. Issue the **bos status** command to verify that all server processes are running correctly. For more detailed instructions, see "Displaying Process Status and Information from the BosConfig File" on page 113.

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

Managing Server IP Addresses and VLDB Server Entries

The AFS support for multihomed file server machines is largely automatic. The File Server process records the IP addresses of its file server machine's network interfaces in the local **/usr/afs/local/sysid** file and also registers them in a *server entry* in the Volume Location Database (VLDB). The **sysid** file and server entry are identified by the same unique number, which creates an association between them.

When the Cache Manager requests volume location information, the Volume Location (VL) Server provides all of the interfaces registered for each server machine that houses the volume. This enables the Cache Manager to make use of multiple addresses when accessing AFS data stored on a multihomed file server machine.

If you wish, you can control which interfaces the File Server registers in its VLDB server entry by creating two files in the local **/usr/afs/local** directory: **NetInfo** and **NetRestrict**. Each time the File Server restarts, it builds a list of the local machine's interfaces by reading the **NetInfo** file, if it exists. If you do not create the file, the File Server uses the list of network interfaces configured with the operating system. It then removes from the list any addresses that appear in the **NetRestrict** file, if it exists. The File Server records the resulting list in the **sysid** file and registers the interfaces in the VLDB server entry that has the same unique identifier.

On database server machines, the **NetInfo** and **NetRestrict** files also determine which interfaces the Ubik database synchronization library uses when communicating with the database server processes running on other database server machines.

There is a maximum number of IP addresses in each server entry, as documented in the *IBM AFS Release Notes*. If a multihomed file server machine has more interfaces than the maximum, AFS simply ignores the excess ones. It is probably appropriate for such machines to use the **NetInfo** and **NetRestrict** files to control which interfaces are registered.

If for some reason the **sysid** file no longer exists, the File Server creates a new one with a new unique identifier. When the File Server registers the contents of the new file, the Volume Location (VL) Server normally recognizes automatically that the new file corresponds to an existing server entry, and overwrites the existing server entry with the new file contents and identifier. However, it is best not to remove the **sysid** file if that can be avoided.

Similarly, it is important not to copy the **sysid** file from one file server machine to another. If you commonly copy the contents of the **/usr/afs** directory from an existing machine as part of installing a new file server machine, be sure to remove the **sysid** file from the **/usr/afs/local** directory on the new machine before starting the File Server.

There are certain cases where the VL Server cannot determine whether it is appropriate to overwrite an existing server entry with a new **sysid** file's contents and identifier. It then refuses to allow the File Server to register the interfaces, which prevents the File Server from starting. This can happen if, for example, a new **sysid** file includes two interfaces that currently are registered by themselves in separate server entries. In such cases, error messages in the **/usr/afs/log/VLLog** file on the VL Server machine and in the **/usr/afs/log/FileLog** file on the file server machine indicate that you need to use the **vos changeaddr** command to resolve the problem. Contact the AFS Product Support group for instructions and assistance.

Except in this type of rare error case, the only appropriate use of the **vos changeaddr** command is to remove a VLDB server entry completely when you remove a file server machine from service. The VLDB can accommodate a maximum number of server entries, as specified in the *IBM AFS Release Notes*. Removing obsolete entries makes it possible to allocate server entries for new file server machines as required. See the instructions that follow.

Do not use the **vos changeaddr** command to change the list of interfaces registered in a VLDB server entry. To change a file server machine's IP addresses and server entry, see the instructions that follow.

To create or edit the server NetInfo file

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

   ```
   % su root
   Password: root_password
   ```

2. Using a text editor, open the **/usr/afs/local/NetInfo** file. Place one IP address in dotted decimal format (for example, `192.12.107.33`) on each line. The order of entries is not significant.
3. If you want the File Server to start using the revised list immediately, use the **bos restart** command to restart the **fs** process. For instructions, see "Stopping and Immediately Restarting Processes" on page 124.

To create or edit the server NetRestrict file

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

   ```
   % su root
   Password: root_password
   ```

2. Using a text editor, open the **/usr/afs/local/NetRestrict** file. Place one IP address in dotted decimal format on each line. The order of the addresses is not significant. Use the value **255** as a wildcard that represents all possible addresses in that field. For example, the entry `192.12.105.255` indicates that the Cache Manager does not register any of the addresses in the 192.12.105 subnet.
3. If you want the File Server to start using the revised list immediately, use the **bos restart** command to restart the **fs** process. For instructions, see "Stopping and Immediately Restarting Processes" on page 124.

To display all server entries from the VLDB

1. Issue the **vos listaddrs** command to display all server entries from the VLDB.

   ```
   % vos listaddrs
   ```

   where **lista** is the shortest acceptable abbreviation of **listaddrs**.

   The output displays all server entries from the VLDB, each on its own line. If a file server machine is multihomed, all of its registered addresses appear on the line. The first one is the one reported as a volume's site in the output from the **vos examine** and **vos listvldb** commands.

   VLDB server entries record IP addresses, and the command interpreter has the local name service (either a process like the Domain Name Service or a local host table) translate them to hostnames before displaying them. If an IP address appears in the output, it is not possible to translate it.

   The existence of an entry does not necessarily indicate that the machine that is still an active file server machine. To remove obsolete server entries, see the following instructions.

To remove obsolete server entries from the VLDB

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **vos changeaddr** command to remove a server entry from the VLDB.

   ```
   % vos changeaddr <original IP address> -remove
   ```

   where

   ch

   Is the shortest acceptable abbreviation of **changeaddr**.

   original IP address

   Specifies one of the IP addresses currently registered for the file server machine in the VLDB. Any of a multihomed file server machine's addresses are acceptable to identify it.

   -remove

   Removes the server entry.

To change a server machine's IP addresses

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If the machine is the system control machine or a binary distribution machine, and you are also changing its hostname, redefine all relevant **upclient** processes on other server machines to refer to the new hostname. Use the **bos delete** and **bos create** commands as instructed in "Creating and Removing Processes" on page 116.
3. If the machine is a database server machine, edit its entry in the **/usr/afs/etc/CellServDB** file on every server machine in the cell to list one of the new IP addresses. If you use the United States edition of AFS, you can edit the file on the system control machine and wait the required time (by default, five minutes) for the Update Server to distribute the changed file to all server machines.
4. If the machine is a database server machine, issue the **bos shutdown** command to stop all server processes. If the machine is also a file server, the volumes on it are inaccessible during this time. For a complete description of the command, see "To stop processes temporarily" on page 122.

   ```
   % bos shutdown <machine name>
   ```

5. Use the utilities provided with the operating system to change one or more of the machine's IP addresses.
6. If appropriate, edit the **/usr/afs/local/NetInfo** file, the **/usr/afs/local/NetRestrict** file, or both, to reflect the changed addresses. Instructions appear earlier in this section.
7. If the machine is a database server machine, issue the **bos restart** command to restart all server processes on the machine. For complete instructions for the **bos restart** command, see "Stopping and Immediately Restarting Processes" on page 124.

   ```
   % bos restart <machine name> -all
   ```

   At the same time, issue the **bos restart** command on all other database server machines in the cell to restart the database server processes only (the Authentication, Backup, Protection, and Volume Location Servers). Issue the commands in quick succession so that all of the database server processes vote in the quorum election.

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

   If you are changing IP addresses on every database server machine in the cell, you must also issue the **bos restart** command on every file server machine in the cell to restart the **fs** process.

8. If the machine is not a database server machine, issue the **bos restart** command to restart the **fs** process (if the machine is a database server, you already restarted the process in the previous step). The File Server automatically compiles a new list of interfaces, records them in the **/usr/afs/local/sysid** file, and registers them in its VLDB server entry.

   ```
   % bos restart <machine name> fs
   ```

9. If the machine is a database server machine, edit its entry in the **/usr/vice/etc/CellServDB** file on every client machine in the cell to list one of the new IP addresses. Instructions appear in "Maintaining Knowledge of Database Server Machines" on page 364.
10. If there are machine entries in the Protection Database for the machine's previous IP addresses, use the **pts rename** command to change them to the new addresses. For instructions, see "Changing a Protection Database Entry's Name" on page 505.

Rebooting a Server Machine

You can reboot a server machine either by typing the appropriate commands at its console or by issuing the **bos exec** command on a remote machine. Remote rebooting can be more convenient, because you do not need to leave your present location, but you cannot track the progress of the reboot as you can at the console. Remote rebooting is possible because the server machine's operating system recognizes the BOS Server, which executes the **bos exec** command, as the local superuser **root**.

Rebooting server machines is part of routine maintenance in some cells, and some instructions in the AFS documentation include it as a step. It is certainly not intended to be the standard method for recovering from AFS-related problems, however, but only a last resort when the machine is unresponsive and you have tried all other reasonable options.

Rebooting causes a service outage. If the machine stores volumes, they are all inaccessible until the reboot completes and the File Server reattaches them. If the machine is a database server machine, information from the databases can become unavailable during the reelection of the synchronization site for each database server process; the VL Server outage generally has the greatest impact, because the Cache Manager must be able to access the VLDB to fetch AFS data.

By convention, a server machine's AFS initialization file includes the following command to restart the BOS Server after each reboot. It starts the other AFS server processes listed in the local **/usr/afs/local/BosConfig** file. These instructions assume that the initialization file includes the command.

```
/usr/afs/bin/bosserver &
```

To reboot a file server machine from its console

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

   ```
   % su root
   Password: root_password
   ```

2. Issue the **bos shutdown** command to shut down all AFS server processes other than the BOS Server, which terminates safely when you reboot the machine. Include the **-localauth** flag because you are logged in as the local superuser **root** but do not necessarily have administrative tokens. For a complete description of the command, see "To stop processes temporarily" on page 122.

   ```
   # bos shutdown <machine name> -localauth [-wait]
   ```

3. Reboot the machine. On many system types, the appropriate command is **shutdown**, but the appropriate options vary; consult your UNIX administrator's guide.

```
# shutdown
```

To reboot a file server machine remotely

1. Verify that you are listed in the **/usr/afs/etc/UserList** file on the machine you are rebooting. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos shutdown** to halt AFS server processes other than the BOS Server, which terminates safely when you turn off the machine. For a complete description of the command, see "To stop processes temporarily" on page 122.

   ```
   % bos shutdown <machine name>  [-wait]
   ```

3. Issue the **bos exec** command to reboot the machine remotely.

   ```
   % bos exec <machine name> reboot_command
   ```

   where

   machine name

   Names the file server machine to reboot.

   reboot_command

   Is the rebooting command for the machine's operating system. The **shutdown** command is appropriate on many system types, but consult your operating system documentation.

Chapter 4. Monitoring and Controlling Server Processes

One of your most important responsibilities as a system administrator is ensuring that the processes on file server machines are running correctly. The BOS Server, which runs on every file server machine, relieves you of much of the responsibility by constantly monitoring the other AFS server processes on its machine. It can automatically restart processes that have failed, ordering the restarts to take interdependencies into account.

Because different file server machines run different combinations of processes, you must define which processes the BOS Server on each file server machine is to monitor (to learn how, see "Controlling and Checking Process Status" on page 111).

It is sometimes necessary to take direct control of server process status before performing routine maintenance or correcting problems that the BOS Server cannot correct (such as problems with database replication or mutual authentication). At those times, you control process status through the BOS Server by issuing **bos** commands.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Examine process status	**bos status**
Examine information from the **BosConfig file** file	**bos status** with **-long** flag
Create a process instance	**bos create**
Stop a process	**bos stop**
Start a stopped process	**bos start**
Stop a process temporarily	**bos shutdown**
Start a temporarily stopped process	**bos startup**
Stop and immediately restart a process	**bos restart**
Stop and immediately restart all processes	**bos restart** with **-bosserver** flag
Examine BOS Server's restart times	**bos getrestart**
Set BOS Server's restart times	**bos setrestart**
Examine a log file	**bos getlog**
Execute a command remotely	**bos exec**

Brief Descriptions of the AFS Server Processes

This section briefly describes the different server processes that can run on an AFS server machine. In cells with multiple server machines, not all processes necessarily run on all machines.

An AFS server process is referred to in one of three ways, depending on the context:

- The output from the **bos status** command refers to a process by the name assigned when the **bos create** command creates its entry in the **/usr/afs/local/BosConfig** file. The name can differ from machine to machine, but it is easiest to maintain the cell if you assign the same name on all machines.

The *IBM AFS Quick Beginnings* and the reference page for the **bos create** command list the conventional names. Examples are **bosserver**, **kaserver**, and **vlserver**.

- The process listing produced by the standard **ps** command generally matches the process's binary file. Examples of process binary files are **/usr/afs/bin/bosserver**, **/usr/afs/bin/kaserver**, and **/usr/afs/bin/vlserver**.
- In most contexts, including most references in the documentation, a process is referred to as (for example) the **Basic OverSeer (BOS) Server**, the **Authentication Server**, or the **Volume Location Server**.

The following sections specify each name for the process as well as some of the administrative tasks in which you use the process. For a more general description of the servers, see "AFS Server Processes and the Cache Manager" on page 8.

The bosserver Process: the Basic OverSeer Server

The **bosserver** process, which runs on every AFS server machine, is the Basic OverSeer (BOS) Server responsible for monitoring the other AFS server processes running on its machine. If a process fails, the BOS Server can restart it automatically, without human intervention. It takes interdependencies into account when restarting a process that has multiple component processes (such as the **fs** process described in "The fs Collection of Processes: the File Server, Volume Server and Salvager" on page 107).

Because the BOS Server does not monitor or restart itself, it does not appear in the output from the **bos status** command. It appears in the **ps** command's output as `/usr/afs/bin/bosserver`.

As a system administrator, you contact the BOS Server when you issue **bos** commands to perform the following kinds of tasks.

- Defining the processes for the BOS Server to monitor by creating entries in the **/usr/afs/local/BosConfig** file as described in "Controlling and Checking Process Status" on page 111
- Stopping and starting processes on the file server machines according to subsequent instructions in this chapter
- Defining your cell's database server machines in the **/usr/afs/etc/CellServDB** file as described in "Maintaining the Server CellServDB File" on page 88
- Defining AFS server encryption keys in the **/usr/afs/etc/KeyFile** file as described in "Managing Server Encryption Keys" on page 333.
- Granting system administrator privileges with respect to BOS Server, Volume Server, and Backup Server operations, by adding a user to the **/usr/afs/etc/UserList** file as described in "Administering the UserList File" on page 535
- Setting authorization checking requirements on a server machine as described in "Managing Authentication and Authorization Requirements" on page 93

The buserver Process: the Backup Server

The **buserver** process, which runs on database server machines, is the Backup Server. It maintains information about Backup System configuration and operations in the Backup Database.

The process appears as `buserver` in the **bos status** command's output, if the conventional name is assigned. It appears in the **ps** command's output as `/usr/afs/bin/buserver`.

As a system administrator, you contact the Backup Server when you issue any **backup** command that manipulates information in the Backup Database, including those that change Backup System configuration information, that dump data from volumes to permanent storage, or that restore data to AFS. See "Configuring the AFS Backup System" on page 195 and "Backing Up and Restoring AFS Data" on page 241.

The fs Collection of Processes: the File Server, Volume Server and Salvager

The **fs** process, which runs on every file server machine, combines three component processes: File Server, Volume Server and Salvager. The three components perform independent functions, but are controlled as a single process for the following reasons.

- They all operate on the same data, namely files and directories stored in AFS volumes. Combining them as a single process enables them to coordinate their actions, never attempting simultaneous operations on the same data that can possibly corrupt it.
- It enables the BOS Server to stop and restart the processes in the required order. When the File Server fails, the BOS Server stops the Volume Server and runs the Salvager to correct any corruption that resulted from the failure. (The Salvager runs only in this special circumstance or when you invoke it yourself by issuing the **bos salvage** command as instructed in "Salvaging Volumes" on page 172.) If only the Volume Server fails, the BOS Server can restart it without affecting the File Server or Salvager.

The File Server component handles AFS data at the level of files and directories, manipulating file system elements as requested by application programs and the standard operating system commands. Its main duty is to deliver requested files to client machines and store them again on the server machine when the client is finished. It also maintains status and protection information about each file and directory. It runs continuously during normal operation.

The Volume Server component handles AFS data at the level of complete volumes rather than files and directories. In response to **vos** commands, it creates, removes, moves, dumps and restores entire volumes, among other actions. It runs continuously during normal operation.

The Salvager component runs only after the failure of one of the other two processes. It checks the file system for internal consistency and repairs any errors it finds.

The process appears as `fs` in the **bos status** command's output, if the conventional name is assigned. An auxiliary message reports the status of the File Server or Salvager component. See "Displaying Process Status and Information from the BosConfig File" on page 113.

The component processes of the **fs** process appear individually in the **ps** command's output, as follows. There is no entry for the `fs` process itself.

- `/usr/afs/bin/fileserver`
- `/usr/afs/bin/volserver`
- `/usr/afs/bin/salvager`

The Cache Manager contacts the File Server component on your behalf whenever you access data or status information in an AFS file or directory or issue file manipulation commands such as the UNIX **cp** and **ls** commands. You can contact the File Server directly by issuing **fs** commands that perform the following functions

- Administering the ACL of any directory in the file system as described in "Managing Access Control Lists" on page 513
- Installing new partitions for housing AFS volumes, in which case you must restart the **fs** process for it to recognize the new partition; for instructions, see "Adding or Removing Disks and Partitions" on page 95
- Creating and deleting volume mount points in the AFS filespace as described in "Mounting Volumes" on page 149
- Setting volume quota and displaying information about the space used and available in a volume or partition as described in "Setting and Displaying Volume Quota and Current Size" on page 176

You contact the Volume Server component when you issue **vos** commands that manipulate volumes in any way--creating, removing, replicating, moving, renaming, converting to different formats, and salvaging. For instructions, see "Managing Volumes" on page 131.

The Salvager normally runs automatically in case of a failure. You can also start it with the **bos salvage** command as described in "Salvaging Volumes" on page 172.

The kaserver Process: the Authentication Server

The **kaserver** process, which runs on database server machines, is the Authentication Server responsible for several aspects of AFS security. It verifies AFS user identity by requiring a password. It maintains all AFS server encryption keys and user passwords in the Authentication Database. The Authentication Server's Ticket Granting Service (TGS) module creates the shared secrets that AFS client and server processes use when establishing secure connections.

The process appears as `kaserver` in the **bos status** command's output, if the conventional name is assigned. The **ka** string stands for *Kerberos Authentication*, reflecting the fact that AFS's authentication protocols are based on Kerberos, which was originally developed at the Massachusetts Institute of Technology's Project Athena.

It appears in the **ps** command's output as `/usr/afs/bin/kaserver`.

As a system administrator, you contact the Authentication Server when you issue **kas** commands to perform the following kinds of tasks.

- Setting a user's password. Users normally change their own passwords, so you probably perform this task only creating a new user account as described in "Creating AFS User Accounts" on page 463 and "Changing AFS Passwords" on page 476.
- Setting the AFS server encryption key in the Authentication Database, which the TGS uses to seal server tickets; see "Managing Server Encryption Keys" on page 333.
- Granting or revoking system administrator privileges with respect to the Authentication Server as described in "Granting Privilege for kas Commands: the ADMIN Flag" on page 534.

The ptserver Process: the Protection Server

The **ptserver** process, which runs on database server machines, is the Protection Server. Its main responsibility is maintaining the Protection Database which contains user, machine, and group entries. The Protection Server allocates AFS IDs and maintains the mapping between them and names. The File Server consults the Protection Server when verifying that a user is authorized to perform a requested action.

The process appears as `ptserver` in the **bos status** command's output, if the conventional name is assigned. It appears in the **ps** command's output as `/usr/afs/bin/ptserver`.

As a system administrator, you contact the Protection Server when you issue **pts** commands to perform the following kinds of tasks.

- Creating a new user, machine, or group entry in the Protection Database as described in "Administering the Protection Database" on page 487
- Adding or removing group members or otherwise manipulating Protection Database entries as described in "Administering the Protection Database" on page 487
- Granting or revoking system administrator privilege by changing the membership of the **system:administrators** group as described in "Administering the system:administrators Group" on page 532

The runntp Process

The **runntp** process, which runs on every server machine, is a controller program for the Network Time Protocol Daemon (NTPD), which synchronizes the hardware clocks on server machines. You need to run the **runntp** process if you are not already running NTP or another time synchronization protocol on your server machines.

The clocks on database server machines need to be synchronized because AFS's distributed database technology (Ubik) works properly only when the clocks agree within a narrow range of variation (see "Configuring the Cell for Proper Ubik Operation" on page 74). The clocks on file server machines need to be correct not only because the File Server sets modification time stamps on files, but because in the conventional configuration they serve as the time source for AFS client machines.

The process appears as `runntp` in the **bos status** command's output, if the conventional name is assigned. It appears in the output from the **ps** command as `/usr/afs/bin/runntp`. The **ps** command's output also includes an entry called `ntpd`; its exact form depends on the arguments you provide to the **runntp** command.

As a system administrator, you do not contact the NTPD directly once you have installed it according to the instructions in the *IBM AFS Quick Beginnings*.

The upserver and upclient Processes: the Update Server

The Update Server has two separate parts, each of which runs on a different type of server machine. The **upserver** process is the server portion of the Update Server. Its function depends on which edition of AFS you use:

- With both the United States and international editions, it runs on the binary distribution machine of each system type you use as a server machine, distributing the contents of each one's **/usr/afs/bin** directory to the other server machines of that type. This guarantees that all machines have the same version of AFS binaries. (For a list of the binaries, see "Binaries in the /usr/afs/bin Directory" on page 60.)
- In you use the United States edition of AFS, it also runs on the cell's system control machine, distributing the contents of its **/usr/afs/etc** directory to all the other server machines in order to synchronize the configuration files stored in that directory. (For a list of the configuration files, see "Common Configuration Files in the /usr/afs/etc Directory" on page 62.)

The **upclient** process is the client portion of the Update Server, and like the server portion its function depends on the AFS edition in use.

- It runs on every server machine that is not a binary distribution machine, referencing the binary distribution machine of its system type as the source for updates to the binaries in the **/usr/afs/bin** directory. The conventional process name to assign is **upclientbin**.
- If you use the United States edition of AFS, another instance of the process runs on every server machine except the system control machine. It references the system control machine as the source for updates to the common configuration files in the **/usr/afs/etc** directory. The conventional process name to assign is **upclientetc**.

In output from the **bos status** command, the server portion appears as `upserver` and the client portions as `upclientbin` and `upclientetc`, if the conventional names are assigned. In the output from the **ps** command, the server portion appears as `/usr/afs/bin/upserver` and the client portions as /usr/afs/bin/upclient.

You do not contact the Update Server directly once you have installed it. It operates automatically whenever you use **bos** commands to change the files that it distributes.

The vlserver Process: the Volume Location Server

The **vlserver** process, which runs on database server machines, is the Volume Location (VL) Server that automatically tracks which file server machines house each volume, making its location transparent to client applications.

The process appears as `vlserver` in the **bos status** command's output, if the conventional name is assigned. It appears in the **ps** command's output as `/usr/afs/bin/vlserver`.

As a system administrator, you contact the VL Server when you issue any **vos** command that changes the status of a volume (it records the status changes in the VLDB).

Controlling and Checking Process Status

To define the AFS server processes that run on a server machine, use the **bos create** command to create entries for them in the local **/usr/afs/local/BosConfig** file. The BOS Server monitors the processes listed in the **BosConfig** file that are marked with the `Run` status flag, and automatically attempts to restart them if they fail. After creating process entries, you use other commands from the **bos** suite to stop and start processes or change the status flag as desired.

Never edit the **BosConfig** file directly rather than using **bos** commands. Similarly, it is not a good practice to run server processes without listing them in the **BosConfig** file, or to stop them using process termination commands such as the UNIX **kill** command.

The Information in the BosConfig File

A process's entry in the **BosConfig** file includes the following information:

- The process's name. The recommended conventional names are defined in both the *IBM AFS Quick Beginnings* and "Creating and Removing Processes" on page 116. The name of a simple process usually matches the name of its binary file (for example, **ptserver** for the Protection Server).
- Its type, which is one of the following:

  simple

  A process that runs independently of any other on the server machine. If several simple processes fail at the same time, the BOS Server can restart them in any order. All standard AFS processes except the **fs** process are simple.

  fs

  A process type reserved for the server process for which the conventional name is also **fs**. This process combines three components: the File Server, the Volume Server, and the Salvager.

  cron

  A process that runs at a defined time rather than continuously. There are no standard processes of this type.

- Its status flag, which tells the BOS Server whether it performs the following two actions with respect to the process:

  - Start the process during BOS Server initialization
  - Restart the process if it (the process) fails

  The two possible values are `Run` (which directs the BOS Server to perform these actions) and `NotRun` (which directs the BOS Server to ignore the process). The BOS Server itself never changes the setting of this flag, even if the process fails repeatedly. Also, this flag is for internal use only; it does not appear in the **bos status** command's output.

- Its command parameters, which are the commands that the BOS Server runs to start the process.
  - A simple processes has one: the complete pathname to its binary file
  - The **fs** process has three: the complete pathnames to each of the three component processes (**/usr/afs/bin/fileserver**, **/usr/afs/bin/volserver**, and **/usr/afs/bin/salvager**)
  - A cron process has two: the first the complete pathname to its binary file, the second the time at which the BOS Server runs it

In addition to process definitions, the **BosConfig** file also records automatic restart times for processes that have new binaries, and for all server processes including the BOS Server. See "Setting the BOS Server's Restart Times" on page 126.

How the BOS Server Uses the Information in the BosConfig File

Whenever the BOS Server starts or restarts, it reads the **BosConfig** file to learn which processes it is to start and monitor. It transfers the information into kernel memory and does not read the **BosConfig** file again until it next restarts. This implies that the BOS Server's memory state can change independently of the **BosConfig** file. You can, for example, stop a process but leave its status flag in the **BosConfig** file as `Run`, or start a process even though its status flag in the **BosConfig** file is `NotRun`.

About Starting and Stopping the Database Server Processes

When you start or stop a database server process (Authentication Server, Backup Server, Protection Server, or Volume Location Server) for more than a short time, you must follow the instructions in the *IBM AFS Quick Beginnings* for installing or removing a database server machine. Here is a summary of the tasks you must perform to preserve correct AFS functioning.

- Start or stop all four database server processes on that machine. All AFS server processes and the Cache Manager processes expect all four database server processes to be running on each machine listed in the **CellServDB** file. There is no way to indicate in the file that a machine is running only some of the database server processes.
- Add or remove the machine in the **/usr/afs/etc/CellServDB** file on all server machines and the **/usr/vice/etc/CellServDB** file on all client machines.
- Restart the database server processes on the other database server machines to force an election of a new Ubik coordinator for each one.

About Starting and Stopping the Update Server

In the conventional cell configuration, one server machine of each system type acts as a binary distribution machine, running the server portion of the Update Server (**upserver** process) to distribute the contents of its **/usr/afs/bin** directory. The other server machines of its system type run an instance of the Update Server client portion (by convention called **upclientbin**) that references the binary distribution machine.

If you run the United States edition of AFS, it is conventional for the first server machine you install to act as the system control machine, running the server portion of the Update Server (**upserver** process) to distribute the contents of its **/usr/afs/etc** directory. All other server machines run an instance of the Update Server client portion (by convention called **upclientetc**) that references the system control machine.

Note: If you are using the international edition of AFS, do not use the Update Server to distribute the contents of the **/usr/afs/etc** directory (you do not run a system control machine). Ignore all references to the process in this chapter.

It is simplest not to move binary distribution or system control responsibilities to a different machine unless you completely decommission a machine that is currently serving in one of those roles. Running the Update Server usually imposes very little processing load. If you must move the functionality, perform the following related tasks.

- If you replace the system control machine, you must stop the **upclientetc** process on every other server machine and define a new one that references the new system control machine.
- If you replace a binary distribution machine, you must stop the **upclientbin** process on every other server machine of its system type and define a new one that references the new binary distribution machine (unless you are no longer running any server machines of that system type).

Displaying Process Status and Information from the BosConfig File

To display the status of the AFS server processes on a server machine, issue the **bos status** command. Adding the **-long** flag displays most of the information from each process's entry in the **BosConfig** file, including its type and command parameters. It also displays a warning message if the mode bits on files and subdirectories in the **/usr/afs** directory do not match the expected values.

To display the status of server processes and their BosConfig entries

1. Issue the **bos status** command.

   ```
   % bos status <machine name>  [<server process name>+]  [-long]
   ```

   where

   stat

   Is the shortest acceptable abbreviation of **status**.

   machine name

   Specifies the file server machine for which to display process status.

   server process name

   Names each process for which to display status, using the name assigned when its entry was defined with the **bos create** command. Omit this argument to display the status of all server processes.

   -long

   Displays, in addition to status, information from the process's entry in the **BosConfig** file: its type, its status flag, its command parameters, the associated notifier program, and so on.

The output includes an entry for each process and uses one of the following strings to indicate the process's status:

- `currently running normally` indicates that the process is running and its status flag in the **BosConfig** file is `Run`. For cron entries, this message indicates that the command is still scheduled to run, not necessarily that it is actually running when the **bos status** command was issued.
- `temporarily enabled` indicates that the process is running but that its status flag in the **BosConfig** file is `NotRun`. The most common reason is that a system administrator has used the **bos startup** command to start the process.
- `temporarily disabled` indicates that the process is not running even though its status flag in the **BosConfig** file is `Run`. The most common reasons are either that a system administrator has used the **bos shutdown** command to stop the process or that the BOS Server ceased trying to restart the process after numerous failed attempts. In the latter case, a supplementary message appears: `stopped for too many errors`.
- disabled indicates that the process is not running and that its status flag in the **BosConfig** file is `NotRun`. The BOS Server is not monitoring the process. Only a system administrator can set the flag this way; the BOS Server never does.

The output for the **fs** process always includes a message marked `Auxiliary status`, which can be one of the following:

- `file server running` indicates that the File Server and Volume Server components of the File Server process are running normally.
- `salvaging file system` indicates that the Salvager is running, which usually implies that the File Server and Volume Server are temporarily disabled. The BOS Server restarts them as soon as the Salvager is finished.

The output for a cron process also includes an `Auxiliary status` message to report when the command is scheduled to run next; see the example that follows.

The output for any process can include the supplementary message `has core file` to indicate that at some point the process failed and generated a core file in the **/usr/afs/logs** directory. In most cases, the BOS Server is able to restart the process and it is running.

The following example includes a user-defined cron entry called **backupusers**:

```
% bos status fs3.abc.com
Instance kaserver, currently running normally.
Instance ptserver, currently running normally.
Instance vlserver, has core file, currently running normally.
Instance buserver, currently running normally.
Instance fs, currently running normally.
    Auxiliary status is: file server running.
Instance upserver, currently running normally.
Instance runntp, currently running normally.
Instance backupusers, currently running normally.
    Auxiliary status is: run next at Mon Jun 7 02:00:00 1999.
```

If you include the **-long** flag to the **bos status** command, a process's entry in the output includes the following additional information from the **BosConfig** file:

- The process's type (`simple`, `fs`, or `cron`).
- The day and time the process last started or restarted.
- The number of `proc starts`, which is how many times the BOS Server has started or restarted the process since it started itself.
- The `Last exit` time when the process (or one of the component processes in the **fs** process) last terminated. This line does not appear if the process has not terminated since the BOS Server started.
- The `Last error exit` time when the process (or one of the component processes in the **fs** process) last failed due to an error. A further explanation such as `due to shutdown request` sometimes appears. This line does not appear if the process has not failed since the BOS Server started.
- Each command that the BOS Server invokes to start the process, as specified by the **-cmd** argument to the **bos create** command.
- The pathname of the notifier program that the BOS Server invokes when the process terminates (if any), as specified by the **-notifier** argument to the **bos create** command.

In addition, if the BOS Server has found that the mode bits on certain files and directories under **/usr/afs** deviate from what it expects, it prints the following warning message:

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

The expected protections for the directories and files in the **/usr/afs** directory are as follows. A question mark indicates that the BOS Server does not check the mode bit. See the *IBM AFS Quick Beginnings* for more information about setting the protections on these files and directories.

/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`

The following illustrates the extended output for the **fs** process running on the machine **fs3.abc.com**:

```
% bos status fs3.abc.com fs -long
Instance fs, (type is fs), currently running normally.
    Auxiliary status is file server running
Process last started at Mon May 3 8:29:19 1999 (3 proc starts)
Last exit at Mon May 3 8:29:19 1999
Last error exit at Mon May 3 8:29:19 1999, due to shutdown request
Command 1 is '/usr/afs/bin/fileserver'
Command 2 is '/usr/afs/bin/volserver'
Command 3 is '/usr/afs/bin/salvager'
```

Creating and Removing Processes

To start a new AFS server process on a server machine, issue the **bos create** command, which creates an entry in the **/usr/afs/local/BosConfig** file, sets the process's status flag to `Run` both in the file and in the BOS Server's memory, and starts it running immediately. The binary file for the new process must already be installed, by convention in the **/usr/afs/bin** directory (see "Installing New Binaries" on page 82).

To stop a process permanently, first issue the **bos stop** command, which changes the process's status flag to `NotRun` in both the **BosConfig** file and the BOS Server's memory; it is marked as `disabled` in the output from the **bos status** command. If desired, issue the **bos delete** command to remove the process's entry from the **BosConfig** file; the process no longer appears in the **bos status** command's output.

Note: If you are starting or stopping a database server process in the manner described in this section, follow the complete instructions in the *IBM AFS Quick Beginnings* for creating or removing a database server machine. If you run one database server process on a given machine, you must run them all; for more information, see "About Starting and Stopping the Database Server Processes" on page 112. Similarly, if you are stopping the **upserver** process on the system control machine or a binary distribution machine, you must complete the additional tasks described in "About Starting and Stopping the Update Server" on page 113.

To create and start a new process

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. **(Optional)** Verify that the process's binaries are installed in the **/usr/afs/bin** directory on this machine. If necessary, login at the console or telnet to the machine and list the contents of the **/usr/afs/bin** directory.

   If the binaries are not present, install them on the binary distribution machine of the appropriate system type, and wait for the Update Server to copy them to this machine. For instructions, see "Installing New Binaries" on page 82.

   ```
   % ls /usr/afs/bin
   ```

3. Issue the **bos create** command to create an entry in the **BosConfig** file and start the process.

   ```
   % bos create <machine name> <server process name>   \
             <server type> <command lines>+ [ -notifier <Notifier program>]
   ```

   where

   cr

   Is the shortest acceptable abbreviation of **create**.

   machine name

   Specifies the file server machine on which to create the process.

   server process name

   Names the process to create and start. For simple processes, the conventional value is the name of the process's binary file. It is best to use the same name on every server machine that runs the process. The following is a list of the conventional names for simple and fs-type processes (there are no standard cron processes).

   - **buserver** for the Backup Server
   - **fs** for the process that combines the File Server, Volume Server, and Salvager
   - **kaserver** for the Authentication Server
   - **ptserver** for the Protection Server

- **runntp** for the controller process for the Network Time Protocol Daemon
- **upclientbin** for the client portion of the Update Server that references the binary distribution machine of this machine's system type
- **upclientetc** for the client portion of the Update Server that references the system control machine
- **vlserver** for the Volume Location (VL) Server

server type

Defines the process's type. Choose one of the following values:

- **cron** for a cron process
- **fs** for the process named **fs**
- **simple** for all other processes listed as acceptable values for the server process name argument

command lines

Specifies each command the BOS Server runs to start the process. Specify no more than six commands (which can include the command's options, in which case the entire string is surrounded by double quotes); any additional commands are ignored.

For a simple process, provide the complete pathname of the process's binary file on the local disk (for example, **/usr/afs/bin/ptserver** for the Protection Server). If including any of the initialization command's options, surround the entire command in double quotes (**" "**). The **upclient** process has a required argument, and the commands for all other processes take optional arguments.

For the **fs** process, provide the complete pathname of the local disk binary file for each of the component processes: **fileserver**, **volserver**, and **salvager**, in that order. The standard binary directory is **/usr/afs/bin**. If including any of an initialization command's options, surround the entire command in double quotes (**" "**).

For a **cron** process, provide two parameters:

- The complete local disk pathname of either an executable file or a command from one of the AFS suites (complete with all of the necessary arguments). Surround this parameter with double quotes (**" "**) if it contains spaces.
- A specification of when the BOS Server executes the file or command indicated by the first parameter. There are three acceptable values:
  - The string **now**, which directs the BOS Server to execute the file or command immediately and only once. It is usually simpler to issue the command directly or issue the **bos exec** command.
  - A time of day. The BOS Server executes the file or command daily at the indicated time. Separate the hours and minutes with a colon (*hh*:*MM*), and use either 24-hour format, or a value in the range from **1:00** through **12:59** with the addition of **am** or **pm**. For example, both **14:30** and **"2:30 pm"** indicate 2:30 in the afternoon. Surround this parameter with double quotes (**" "**) if it contains a space.

- A day of the week and time of day, separated by a space and surrounded with double quotes (**" "**). The BOS Server executes the file or command weekly at the indicated day and time. For the day, provide either the whole name or the first three letters, all in lowercase letters (**sunday** or **sun**, **thursday** or **thu**, and so on). For the time, use the same format as when specifying the time alone.

-notifier

Specifies the pathname of a program that the BOS Server runs when the process terminates. For more information on notifier programs, see the **bos create** command reference page in the *IBM AFS Administration Reference*.

The following example defines and starts the Protection Server on the machine **db2.abc.com**:

```
% bos create db2.abc.com ptserver simple /usr/afs/bin/ptserver
```

The following example defines and starts the **fs** process on the machine **fs6.abc.com**.

```
% bos create fs6.abc.com fs fs /usr/afs/bin/fileserver   \
      /usr/afs/bin/volserver /usr/afs/bin/salvager
```

The following example defines and starts a cron process called **backupuser** process on the machine **fs3.abc.com**, scheduling it to run each day at 3:00 a.m.

```
% bos create fs3.abc.com backupuser cron  "/usr/afs/bin/vos backupsys -prefix user -loca
```

To stop a process and remove it from the BosConfig file

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos stop** command to change each process's status flag in the **BosConfig** file to `NotRun` and to stop it. You must issue this command even for cron processes that you wish to remove from the **BosConfig** file, even though they do not run continuously. For a detailed description of this command, see "To stop a process by changing its status to `NotRun`" on page 120.

   ```
   % bos stop <machine name> <server process name>+ [-wait]
   ```

3. Issue the **bos delete** command to remove each process from the **BosConfig** file.

   ```
   % bos delete <machine name> <server process name>+
   ```

   where

d

Is the shortest acceptable abbreviation of **delete**.

machine name

Specifies the server machine on which to remove processes from the **BosConfig** file.

server process name

Names each process entry to remove from the **BosConfig** file. Provide the same names as in Step "2" on page 119.

Stopping and Starting Processes Permanently

To stop a process so that the BOS Server no longer attempts to monitor it, issue the **bos stop** command. The process's status flag is set to `NotRun` in both the BOS Server's memory and in the **BosConfig** file. The process does not run again until you issue the **bos start** command, which sets its status flag back to `Run` in both the BOS Server's memory and in the **BosConfig** file. (You can also use the **bos startup** command to start the process again without changing its status flag in the **BosConfig** file; see "Stopping and Starting Processes Temporarily" on page 122.)

There is no entry for the BOS Server in the **BosConfig** file, so the **bos stop** and **bos start** commands do not control it. To stop and immediately restart the BOS Server along with all other processes, use the **-bosserver** flag to the **bos restart** command as described in "Stopping and Immediately Restarting Processes" on page 124.

Note: If you are starting or stopping a database server process in the manner described in this section, follow the complete instructions in the *IBM AFS Quick Beginnings* for creating or removing a database server machine. If you run one database server process on a given machine, you must run them all; for more information, see "About Starting and Stopping the Database Server Processes" on page 112. Similarly, if you are stopping the **upserver** process on the system control machine or a binary distribution machine, you must complete the additional tasks described in "About Starting and Stopping the Update Server" on page 113.

To stop a process by changing its status to NotRun

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos stop** command to stop each process and set its status flag to `NotRun` in the **BosConfig** file and the BOS Server's memory.

```
% bos stop <machine name> <server process name>+ [-wait]
```

where

sto

Is the shortest acceptable abbreviation of **stop**.

machine name

Specifies the server machine on which to stop the process.

server process name

Names each process to stop, using the name assigned when its entry was defined with the **bos create** command.

-wait

Delays the return of the command shell prompt until all specified processes have stopped. If you omit the flag, the prompt returns almost immediately, even if all processes are not yet stopped.

To start processes by changing their status flags to Run

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos start** command to change each process's status flag to `Run` in both the **BosConfig** file and the BOS Server's memory and to start it.

   ```
   %  bos start <machine name> <server process name>+
   ```

where

start

Must be typed in full.

machine name

Specifies the server machine on which to start running each process.

server process name

Specifies each process to start on machine name. Use the name assigned to the process at creation.

Stopping and Starting Processes Temporarily

It is sometimes necessary to halt a process temporarily (for example, to make slight configuration changes or to perform maintenance). The commands described in this section change a process's status in the BOS Server's memory only; the effect is immediate and lasts until you change the memory state again (or until the BOS Server restarts, at which time it starts the process according to its entry in the **BosConfig** file).

To stop a process temporarily by changing its status flag in BOS Server memory to `NotRun`, use the **bos shutdown** command. To restart a stopped process by changing its status flag in the BOS Server's memory to `Run`, use the **bos startup** command. The process starts regardless of its status flag in the **BosConfig** file. You can also use the **bos startup** command to start all processes marked with status flag `Run` in the **BosConfig** file, as described in the following instructions.

Because the **bos startup** command starts a process without changing it status flag in the **BosConfig** file, it is useful for testing a server process without enabling it permanently. To stop and start processes by changing their status flags in the **BosConfig** file, see "Stopping and Starting Processes Permanently" on page 120; to stop and immediately restart a process, see "Stopping and Immediately Restarting Processes" on page 124.

Note: Do not temporarily stop a database server process on all machines at once. Doing so makes the database completely unavailable.

To stop processes temporarily

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos shutdown** command to stop each process by changing its status flag in the BOS Server's memory to `NotRun`.

   ```
   % bos shutdown <machine name> [<instances>+] [-wait]
   ```

   where

   sh

   Is the shortest acceptable abbreviation of **shutdown**.

   machine name

   Specifies the server machine on which to stop processes temporarily.

   instances

   Specifies each process to stop temporarily. Use the name assigned to the process at creation.

-wait

Delays the return of the command shell prompt until all specified processes have actually stopped. If you omit the flag, the prompt returns almost immediately, even if all processes are not yet stopped.

To start all stopped processes that have status flag Run in the BosConfig file

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos startup** command to start each process on a machine that has status flag `Run` in the **BosConfig** file by changing its status flag in the BOS Server's memory from `NotRun` to `Run`.

   ```
   % bos startup <machine name>
   ```

   where

   startup

   Must be typed in full.

   machine name

   Specifies the server machine on which you wish to start all processes that have status flag `Run` in the **BosConfig** file.

To start specific processes

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos startup** command to start specific processes by changing their status flags in the BOS Server's memory to `Run` without changing their status flags in the **BosConfig** file.

   ```
   % bos startup <machine name> <instances>+
   ```

   where

startup

Must be typed in full.

machine name

Names the server machine on which to start processes.

instances

Specifies each process to start. Use the name assigned to the process at creation.

Stopping and Immediately Restarting Processes

Although by default the BOS Server checks each day for new installed binary files and restarts the associated processes, it is sometimes desirable to stop and restart processes immediately. The **bos restart** command provides this functionality, starting a completely new instance of each affected process:

- To stop and restart the BOS Server, which then restarts all processes marked with the `Run` status flag in the **BosConfig** file, include the **-bosserver** flag.
- To stop and restart all processes marked with the `Run` status flag in the **BosConfig** file, include the **-all** flag. The BOS Server does not restart
- To stop and restart specific processes regardless of the setting of their status flags in the **BosConfig** file, specify the name of each process to restart.

Restarting processes causes a service outage. It is usually best to schedule restarts for periods of low usage. The BOS Server automatically restarts all processes once a week, to reduce the potential for the *core leaks* that can develop as any process runs for an extended time; see "Setting the BOS Server's Restart Times" on page 126.

To stop and restart all processes including the BOS Server

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos restart** command with the **-bosserver** flag to stop and restart the BOS Server, which restarts every process marked with status flag `Run` in the **BosConfig** file.

   ```
   % bos restart <machine name>  -bosserver
   ```

   where

res

Is the shortest acceptable abbreviation of **restart**.

machine name

Specifies the server machine on which to restart all processes.

-bosserver

Stops the BOS Server and all processes running on the machine. A new BOS Server instance starts; it then starts new instances of all processes marked with status flag `Run` in the **BosConfig** file.

To stop and immediately restart all processes except the BOS Server

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **bos restart** command with the **-all** flag to stop and immediately restart every process marked with status flag `Run` in the **BosConfig** file. The BOS Server does not restart.

```
% bos restart <machine name> -all
```

where

res

Is the shortest acceptable abbreviation of **restart**.

machine name

Specifies the server machine on which to stop and restart processes.

-all

Stops and immediately restarts all processes marked with status flag `Run` in the **BosConfig** file.

To stop and immediately restart specific processes

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **bos restart** command to stop and immediately restart one or more specified processes, regardless of their status flag setting in the **BosConfig** file.

   ```
   % bos restart <machine name> <instances>+
   ```

   where

   res

   Is the shortest acceptable abbreviation of **restart**.

   machine name

   Names the server machine on which to restart the specified processes.

   instances

   Specifies each process to stop and immediately restart. Use the name assigned to the process at creation.

Setting the BOS Server's Restart Times

The BOS Server by default restarts once a week, and the new instance restarts all processes marked with status flag `Run` in the local **/usr/afs/local/BosConfig** file (this is equivalent to issuing the **bos restart** command with the **-bosserver** flag). The default restart time is Sunday at 4:00 a.m. The weekly restart is designed to minimize core leaks, which can develop as a process continues to allocate virtual memory but does not free it again. When the memory is completely exhausted, the machine can no longer function correctly.

The BOS Server also by default checks once a day for any newly installed binary files. If it finds that the modification time stamp on a process's binary file in the **/usr/afs/bin** directory is more recent than the time at which the process last started, it restarts the process so that a new instance starts using the new binary file. The default binary-checking time is 5:00 a.m.

Because restarts can cause outages during which the file system is inaccessible, the default times for restarts are in the early morning when usage is likely to be lowest. Restarting a database server process on any database server machine usually makes the entire system unavailable to everyone for a brief time, whereas restarting other types of processes inconveniences only users interacting with that process on that machine. The longest outages typically result from restarting the **fs** process, because the File Server must reattach all volumes.

The **BosConfig** file on each file server machine records the two restart times. To display the current setting, issue the **bos getrestart** command. To reset a time, use the **bos setrestart** command.

To display the BOS Server restart times

1. Issue the **bos getrestart** command to display the automatic restart times.

   ```
   % bos getrestart <machine name>
   ```

   where

   getr

   Is the shortest acceptable abbreviation of **getrestart**.

   machine name

   Specifies the server machine for which to display the restart times.

To set the general or binary restart time

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos setrestart** command with the **-general** flag to set the general restart time or the **-newbinary** flag to set the binary restart time. The command accepts only one of the flags at a time.

   ```
   % bos setrestart <machine name> "<time to restart server>" [-general]  [-newbinary]
   ```

   where

   setr

   Is the shortest acceptable abbreviation of **setrestart**.

   machine name

   Specifies the server machine.

   time to restart server

   Sets when the BOS Server restarts itself (if combined with the **-general** flag) or any process with a new binary file (if combined with the **-newbinary** flag). Provide one of the following types of values:

   - The string **never**, which directs the BOS Server never to perform the indicated type of restart.
   - A time of day (the conventional type of value for the binary restart time). Separate the hours and minutes with a colon (*hh*:*MM*), and use either 24-hour format, or a value in the range from **1:00** through **12:59** with the addition of **am** or **pm**. For example, both **14:30** and **"2:30**

pm" indicate 2:30 in the afternoon. Surround this parameter with double quotes (**" "**) if it contains a space.

- A day of the week and time of day, separated by a space and surrounded with double quotes (**" "**). This is the conventional type of value for the general restart. For the day, provide either the whole name or the first three letters, all in lowercase letters (**sunday** or **sun**, **thursday** or **thu**, and so on). For the time, use the same format as when specifying the time alone.

If desired, precede a time or day and time definition with the string **every** or **at**. These words do not change the meaning, but possibly make the output of the **bos getrestart** command easier to understand.

> **Note:** If the specified time is within one hour of the current time, the BOS Server does not perform the restart until the next eligible time (the next day for a time or next week for a day and time).

-general

Sets the general restart time when the BOS Server restarts itself.

-newbinary

Sets the restart time for processes with new binary files.

Displaying Server Process Log Files

The **/usr/afs/logs** directory on each file server machine contains log files that detail interesting events that occur during normal operation of some AFS server processes. The self-explanatory information in the log files can help you evaluate process failures and other problems. To display a log file remotely, issue the **bos getlog** command. You can also establish a connection to the server machine and use a text editor or other file display program (such as the **cat** command).

> **Note:** Log files can grow unmanageably large if you do not periodically shutdown and restart the database server processes (for example, if you disable the general restart time). In this case it is a good policy periodically to issue the UNIX **rm** command to delete the current log file. The server process automatically creates a new one as needed.

To examine a server process log file

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos getlog** command to display a log file.

   ```
   % bos getlog  <machine name>  <log file to examine>
   ```

   where

   getl

   Is the shortest acceptable abbreviation of **getlog**.

   machine name

   Specifies the server machine from which to display the log file.

   log file to examine

   Names the log file to be displayed. Provide one of the following file names to display the indicated log file from the **/usr/afs/logs** directory.

   - **AuthLog** for the Authentication Server log file
   - **BackupLog** for the Backup Server log file
   - **BosLog** for the BOS Server log file
   - **FileLog** for the File Server log file
   - **SalvageLog** for the Salvager log file
   - **VLLog** for the Volume Location (VL) Server log file
   - **VolserLog** for the Volume Server log file

   You can provide a full or relative pathname to display a file from another directory. Relative pathnames are interpreted relative to the **/usr/afs/logs** directory.

Chapter 5. Managing Volumes

This chapter explains how to manage the volumes stored on file server machines. The volume is the designated unit of administration in AFS, so managing them is a large part of the administrator's duties.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Create read/write volume	**vos create**
Create read-only volume	**vos addsite and vos release**
Create backup volume	**vos backup**
Create many backup volumes at once	**vos backupsys**
Examine VLDB entry	**vos listvldb**
Examine volume header	**vos listvol**
Examine both VLDB entry and volume header	**vos examine**
Display volume's name	**fs listquota or fs examine**
Display volume's ID number	**fs examine or vos examine or vos listvol**
Display partition's size and space available	**vos partinfo**
Display volume's location	**fs whereis or vos examine**
Create mount point	**fs mkmount**
Remove mount point	**fs rmmount**
Display mount point	**fs lsmount**
Move read/write volume	**vos move**
Synchronize VLDB with volume headers	**vos syncvldb and vos syncserv**
Set volume quota	**fs setvol or fs setquota**
Display volume quota	**fs quota or fs listquota or fs examine**
Display volume's current size	**fs listquota or fs examine**
Display list of volumes on a machine/partition	**vos listvol**
Remove read/write volume	**vos remove and fs rmmount**
Remove read-only volume	**vos remove**
Remove backup volume	**vos remove and fs rmmount**
Remove volume; no VLDB change	**vos zap**
Remove read-only site definition	**vos remsite**
Remove VLDB entry; no volume change	**vos delentry**
Dump volume	**vos dump**
Restore dumped volume	**vos restore**
Rename volume	**vos rename**, **fs rmmount and fs mkmount**
Unlock volume	**vos unlock**
Unlock multiple volumes	**vos unlockvldb**
Lock volume	**vos lock**

About Volumes

An AFS *volume* is a logical unit of disk space that functions like a container for the files in an AFS directory, keeping them all together on one partition of a file server machine. To make a volume's contents visible in the cell's file tree and accessible to users, you mount the volume at a directory location in the AFS filespace. The association between the volume and its location in the filespace is called a *mount point*, and because of AFS's internal workings it looks and acts just like a standard directory element. Users can access and manipulate a volume's contents in the same way they access and manipulate the contents of a standard UNIX directory. For more on the relationship between volumes and directories, see "About Mounting Volumes" on page 134.

Many of an administrator's daily activities involve manipulating volumes, since they are the basic storage and administrative unit of AFS. For a discussion of some of the ways volumes can make your job easier, see "How Volumes Improve AFS Efficiency" on page 133.

The Three Types of Volumes

There are three types of volumes in AFS, as described in the following list:

- The single *read/write* version of a volume houses the modifiable versions of the files and directories in that volume. It is often referred to as the *read/write* source because volumes of the other two types are derived from it by a copying procedure called *cloning*. For instructions on creating read/write volumes, see "Creating Read/write Volumes" on page 135.
- A *read-only* volume is a copy of the read/write source volume and can exist at multiple *sites* (a site is a particular partition on a particular file server machine). Placing the same data at more than one site is called *replication*; see "How Volumes Improve AFS Efficiency" on page 133. As the name suggests, a read-only volume's contents do not change automatically as the read/write source changes, but only when an administrator issues the **vos release** command. For users to have a consistent view of the AFS filespace, all copies of the read-only volume must match each other and their read/write source. All read-only volumes share the same name, which is derived by adding the **.readonly** extension to the read/write source's name. For instructions on creating of read-only volumes, see "Replicating Volumes (Creating Read-only Volumes)" on page 139.
- A *backup* volume is a clone of the read/write source volume and is stored at the same site as the source. A backup version is useful because it records the state of the read/write source at a certain time, allowing recovery of data that is later mistakenly changed or deleted (for further discussion see "How Volumes Improve AFS Efficiency" on page 133). A backup volume's name is derived by adding the **.backup** extension to the read/write source's name. For instructions on creating of backup volumes, see "Creating Backup Volumes" on page 144.

  Note: A backup volume is not the same as the backup of a volume transferred to tape using the AFS Backup System, although making a backup version of a volume is usually a stage in the process of backing up the volume to tape. For information on backing up a volume using the AFS Backup System, see "Backing Up Data" on page 251.

As noted, the three types of volumes are related to one another: read-only and backup volumes are both derived from a read/write volume through a process called cloning. Read-only and backup volumes are exact copies of the read/write source at the time they are created.

How Volumes Improve AFS Efficiency

Volumes make your cell easier to manage and more efficient in the following three ways:

- Volumes are easy to move between partitions, on the same or different machines, because they are by definition smaller than a partition. Perhaps the most common reasons to move volumes are to balance the load among file server machines or to take advantage of greater disk capacity on certain machines. You can move volumes as often as necessary without disrupting user access to their contents, because the move procedure makes the contents unavailable for only a few seconds. The automatic tracking of volume locations in the Volume Location Database (VLDB) assures that access remains transparent. For instructions on moving volumes, see "Moving Volumes" on page 166.
- Volumes are the unit of replication in AFS. *Replication* refers to creating a read-only clone from the read/write source and distributing of the clone to one or more sites. Replication improves system efficiency because more than one machine can fill requests for popular files. It also boosts system reliability by helping to keep data available in the face of machine or server process outage. In general, volumes containing popular application programs and other files that do not change often are the best candidates for replication, but you can replicate any read/write volume. See "Replicating Volumes (Creating Read-only Volumes)" on page 139.
-

  Volumes are the unit of backup in AFS, in two senses. You can create a backup volume version to preserves the state of a read/write source volume at a specified time. You can mount the backup version in the AFS filespace, enabling users to restore data they have accidentally changed or deleted without administrator assistance, which frees you for more important jobs. If you make a new backup version of user volumes once a day (presumably overwriting the former backup), then users are always be able to retrieve the previous day's version of a file. For instructions, see "Creating Backup Volumes" on page 144.

  Backup also refers to using the AFS Backup System to store permanent copies of volume contents on tape or in a special backup data. See "Configuring the AFS Backup System" on page 195and "Backing Up and Restoring AFS Data" on page 241.

Volume Information in the VLDB

The Volume Location Database (VLDB) includes entries for every volume in a cell. Perhaps the most important information in the entry is the volume's location, which is key to transparent access to AFS data. When a user opens a file, the Cache Manager consults the Volume Location (VL) Server, which maintains the VLDB, for a list of the file server machines that house the volume containing the file. The Cache Manager then requests the file from the File Server running on one of the relevant file server machines. The file location procedure is invisible to the user, who only needs to know the file's pathname.

The VLDB volume entry for a read/write volume also contains the pertinent information about the read-only and backup versions, which do not have their own VLDB entries. (The rare exception is a read-only volume that has its own VLDB entry because its read/write source has been removed.) A volume's VLDB entry records the volume's name, the unique volume ID number for each version (read/write, read-only, backup, and releaseClone), a count of the number of sites that house a read/write or read-only version, and a list of the sites.

To display the VLDB entry for one or more volumes, use the **vos listvldb** command as described in "To display VLDB entries" on page 157. To display the VLDB entry for a single volume along with its *volume header*, use the **vos examine** command as described in "To display one volume's VLDB entry and volume header" on page 163. (See the following section for a description of the volume header.)

The Information in Volume Headers

Whereas all versions of a volume share one VLDB entry, each volume on an AFS server partition has its own volume header, a data structure that maps the files and directories in the volume to physical memory addresses on the partition that stores them. The volume header binds the volume's contents into a logical unit without requiring that they be stored in contiguous memory blocks. The volume header also records the following information about the volume, some of it redundant with the VLDB entry: name, volume ID number, type, size, status (online, offline, or busy), space quota, timestamps for creation date and date of last modification, and number of accesses during the current day.

To display the volume headers on one or more partitions, use the **vos listvol** command as described in "To display volume headers" on page 159. To display the VLDB entry for a single volume along with its volume header, use the **vos examine** command as described in "To display one volume's VLDB entry and volume header" on page 163.

Keeping the VLDB and Volume Headers Synchronized

It is vital that the information in the VLDB correspond to the status of the actual volumes on the servers (as recorded in volume headers) as much of the time as possible. If a volume's location information in the VLDB is incorrect, the Cache Manager cannot find access its contents. Whenever you issue a **vos** command that changes a volume's status, the Volume Server and VL Server cooperate to keep the volume header and VLDB synchronized. In rare cases, the header and VLDB can diverge, for instance because a **vos** operation halts prematurely. For instructions on resynchronizing them, see "Synchronizing the VLDB and Volume Headers" on page 168.

About Mounting Volumes

To make a volume's contents visible in the cell's file tree and accessible to users, you mount the volume at a directory location in the AFS filespace. The association between the volume and its location in the filespace is called a *mount point*. An AFS mount point looks and functions like a regular UNIX file system directory, but structurally it is more like a symbolic link that tells the Cache Manager the name of the volume associated with the directory. A mount point looks and acts like a directory only because the Cache Manager knows how to interpret it.

Consider the common case where the Cache Manager needs to retrieve a file requested by an application program. The Cache Manager traverses the file's complete pathname, starting at the AFS root (by convention mounted at the **/afs** directory) and continuing to the file. When the Cache Manager encounters (or *crosses*) a mount point during the traversal, it reads it to learn the name of the volume mounted at that directory location. After obtaining location information about the volume from the Volume Location (VL) Server, the Cache Manager fetches the indicated volume and opens its *root directory*. The root directory of a volume lists all the files, subdirectories, and mount points that reside in it. The Cache Manager scans the root directory listing for the next element in the pathname. It continues down the path, using this method to interpret any other mount points it encounters, until it reaches the volume that houses the requested file.

Mount points act as the glue that connects the AFS file space, creating the illusion of a single, seamless file tree even when volumes reside on many different file server machines. A volume's contents are visible and accessible when the volume is mounted at a directory location, and are not accessible at all if the volume is not mounted.

You can mount a volume at more than one location in the file tree, but this is not recommended for two reasons. First, it distorts the hierarchical nature of the filespace. Second, the Cache Manager can become confused about which pathname it followed to reach the file (causing unpredictable output from the **pwd** command, for example). However, if you mount a volume at more than one directory, the access control list (ACL) associated with the volume's root directory applies to all of the mount points.

There are several types of mount points, each of which the Cache Manager handles in a different way and each of which is appropriate for a different purpose. See "Mounting Volumes" on page 149.

About Volume Names

A read/write volume's name can be up to 22 characters in length. The Volume Server automatically adds the **.readonly** and **.backup** extensions to read-only and backup volumes respectively. Do not explicitly add the extensions to volume names, even if they are appropriate.

It is conventional for a volume's name to indicate the type of data it houses. For example, it is conventional to name all user volumes **user**.username where username is the user's login name. Similarly, many cells elect to put system binaries in volumes with names that begin with the system type code. For a list of other naming conventions, see "Creating Volumes to Simplify Administration" on page 26.

Creating Read/write Volumes

A read/write volume is the most basic type of volume, and must exist before you can create read-only or backup versions of it. When you issue the **vos create** command to create a read/write volume, the VL Server creates a VLDB entry for it which records the name you specify, assigns a read/write volume ID number, and reserves the next two consecutive volume ID numbers for read-only and backup versions that possibly are to be created later. At the same time, the Volume Server creates a volume header at the site you designate, allocating space on disk to record the name of the volume's root directory. The name is filled in when you issue the **fs mkmount** command to mount the volume, and matches the mount point name. The following is also recorded in the volume header:

- An initial ACL associated with the volume's root directory. By default it grants all seven AFS access permissions to the **system:administrators** group. After you mount the volume, you can use the **fs**

setacl command to add other entries and to remove or change the entry for the **system:administrators** group. See "Setting ACL Entries" on page 521.

- A space quota, which limits the amount of disk space the read/write version of the volume can use on the file server partition. The default is of 5000 kilobyte blocks, but you can use the **-maxquota** argument to the **vos create** command to set a different quota.

  To change the quota after creation, use the **fs setquota** command as described in "Setting and Displaying Volume Quota and Current Size" on page 176.

To create (and mount) a read/write volume

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Verify that you have the **a**(**administer**), **i**(**insert**), and **l**(**lookup**) permissions on the ACL of the directory where you plan to mount the volume. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

   Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

3. Select a site (disk partition on a file server machine) for the new volume. To verify that the site has enough free space to house the volume (now, or if it grows to use its entire quota), issue the **vos partinfo** command.

   Note: The partition-related statistics in this command's output do not always agree with the corresponding values in the output of the standard UNIX **df** command. The statistics reported by this command can be up to five minutes old, because the Cache Manager polls the File Server for partition information at that frequency. Also, on some operating systems, the **df** command's report of partition size includes reserved space not included in this command's calculation, and so is likely to be about 10% larger.

   ```
   % vos partinfo <machine name> [<partition name>]
   ```

   where

   p

   Is the shortest acceptable abbreviation of **partinfo**.

machine name

Specifies the file server machine for which to display partition size and usage.

partition name

Names one partition for which to display partition size and usage. If you omit it, the output displays the size and space available for all partitions on the machine.

4. Select a volume name, taking note of the information in "About Volume Names" on page 135.
5. Issue the **vos create** command to create the volume.

```
% vos create <machine name> <partition name> <volume name> \
     [-maxquota <initial quota (KB)>]
```

where

cr

Is the shortest acceptable abbreviation of **create**.

machine name

Specifies the file server machine on which to place the volume.

partition name

Specifies the disk partition on which to place the volume.

volume name

Names the volume. It can be up to 22 alphanumeric and punctuation characters in length. Your cell possibly has naming conventions for volumes, such as beginning user volume names with the string **user** and using the period to separate parts of the name.

-maxquota

Sets the volume’s quota, as a number of kilobyte blocks. If you omit this argument, the quota is set to 5000 kilobyte blocks.

6.

(Optional) Issue the **fs mkmount** command to mount the volume in the filespace. For complete syntax, see "To create a regular or read/write mount point" on page 153.

```
% fs mkmount <directory> <volume name>
```

7. **(Optional)** Issue the **fs lsmount** command to verify that the mount point refers to the correct volume. Complete instructions appear in "To display a mount point" on page 152.

```
% fs lsmount <directory>
```

8. **(Optional)** Issue the **fs setvol** command with the **-offlinemsg** argument to record auxiliary information about the volume in its volume header. For example, you can record who owns the volume or where you have mounted it in the filespace. To display the information, use the **fs examine** command.

   ```
   % fs setvol <dir/file path> -offlinemsg  <offline message>
   ```

   where

   sv

   Is an acceptable alias for **setvol**(and **setv** the shortest acceptable abbreviation).

   dir/file path

   Names the mount point of the volume with which to associate the message. Partial pathnames are interpreted relative to the current working directory.

   Specify the read/write path to the mount point, to avoid the failure that results when you attempt to change a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

   -offlinemsg

   Specifies up to 128 characters of auxiliary information to record in the volume header.

About Clones and Cloning

To create a backup or read-only volume, the Volume Server begins by *cloning* the read/write source volume to create a *clone*. The Volume Server creates the clone automatically when you issue the **vos backup** or **vos backupsys** command (for a backup volume) or the **vos release** command (for a read-only volume). No special action is required on your part.

A clone is not a copy of the data in the read/write source volume, but rather a copy of the read/write volume's *vnode index*. The vnode index is a table of pointers between the files and directories in the volume and the physical disk blocks on the partition where the data resides. From the clone, backup and read-only volumes are created in the following manner:

- A read-only volume that occupies the same partition as its read/write source (also known as a *read-only clone*), and a backup volume, are created by attaching a volume header to the clone. These volumes initially consume very little disk space, because the clone portion (the vnode index) points to exactly the same files as the read/write volume, as illustrated in "Figure 1" on page 139. The file sharing is possible only because the clone is on the same partition as the read/write source volume. When a file in the read/write volume is deleted, it is not actually removed from the partition, because the backup or read-only clone still points to it. Similarly, when a file in the read/write is changed, the entire original file is preserved on disk because the clone still points to it, and the read/write volume's vnode index changes to point to newly space for the changed file. When this happens, the backup or read-only volume is said to grow or start occupying actual disk space.

- A read-only volume that does not occupy the same site as the read/write source is a copy of the clone and of all of the data in the read/write source volume. It occupies the same amount of disk space as the read/write volume did at the time the read-only volume was created.

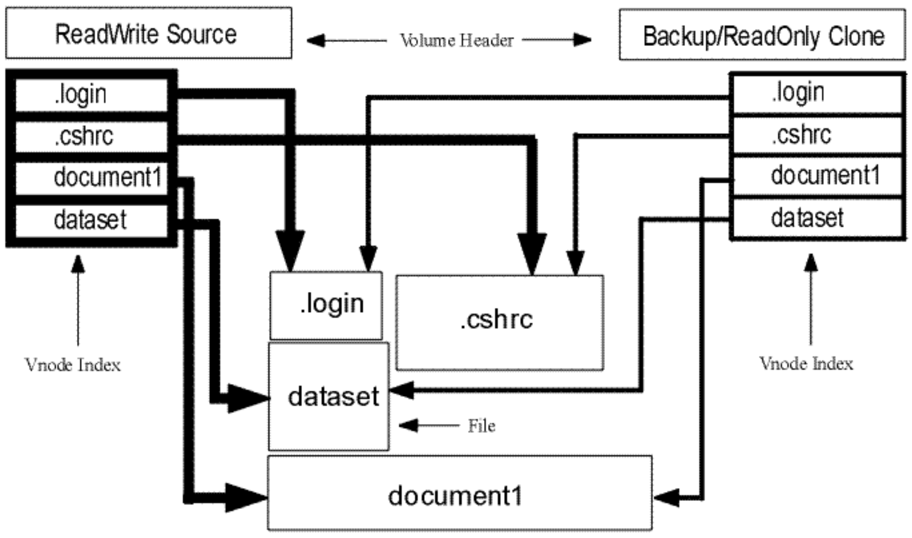


Figure 1. File Sharing Between the Read/write Source and a Clone Volume

Replicating Volumes (Creating Read-only Volumes)

Replication refers to creating a read-only copy of a read/write volume and distributing the copy to one or more additional file server machines. Replication makes a volume's contents accessible on more than one file server machine, which increases data availability. It can also increase system efficiency by reducing load on the network and File Server. Network load is reduced if a client machine's server preference ranks lead the Cache Manager to access the copy of a volume stored on the closest file server machine. Load on the File Server is reduced because it issues only one callback for all data fetched from a read-only volume, as opposed to a callback for each file fetched from a read/write volume. The single callback is sufficient for an entire read-only volume because the volume does not change except in response to administrator action, whereas each read/write file can change at any time.

Replicating a volume requires issuing two commands. First, use the **vos addsite** command to add one or more read-only site definitions to the volume's VLDB entry (a *site* is a particular partition on a file server machine). Then use the **vos release** command to clone the read/write source volume and distribute the clone to the defined read-only sites. You issue the **vos addsite** only once for each read-only site, but must reissue the **vos release** command every time the read/write volume's contents change and you want to update the read-only volumes.

For users to have a consistent view of the file system, the release of updated volume contents to read-only sites must be atomic: either all read-only sites receive the new version of the volume, or all sites keep the version they currently have. The **vos release** command is designed to ensure that all copies of the volume's read-only version match both the read/write source and each other. In cases where problems such as machine or server process outages prevent successful completion of the release operation, AFS uses two mechanisms to alert you.

First, the command interpreter generates an error message on the standard error stream naming each read-only site that did not receive the new volume version. Second, during the release operation the Volume Location (VL) Server marks site definitions in the VLDB entry with flags (`New release` and `Old release`) that indicate whether or not the site has the new volume version. If any flags remain after the operation completes, it was not successful. The Cache Manager refuses to access a read-only site marked with the `Old release` flag, which potentially imposes a greater load on the sites marked with the `New release` flag. It is important to investigate and eliminate the cause of the failure and then to issue the **vos release** command as many times as necessary to complete the release without errors.

The pattern of site flags remaining in the volume's VLDB entry after a failed release operation can help determine the point at which the operation failed. Use the **vos examine** or **vos listvldb** command to display the VLDB entry. The VL Server sets the flags in concert with the Volume Server's operations, as follows:

1. Before the operation begins, the VL Server sets the `New release` flag on the read/write site definition in the VLDB entry and the `Old release` flag on read-only site definitions (unless the read-only site has been defined since the last release operation and has no actual volume, in which case its site flag remains `Not released`).
2. If necessary, the Volume Server creates a temporary copy (a *clone*) of the read/write source called the ReleaseClone (see the following discussion of when the Volume Server does or does not create a new ReleaseClone.) It assigns the ReleaseClone its own volume ID number, which the VL Server records in the `RClone` field of the source volume's VLDB entry.
3. The Volume Server distributes a copy of the ReleaseClone to each read-only site defined in the VLDB entry. As the site successfully receives the new clone, the VL Server sets the site's flag in the VLDB entry to `New release`.
4. When all the read-only copies are successfully released, the VL Server clears all the `New release` site flags. The ReleaseClone is no longer needed, so the Volume Server deletes it and the VL Server erases its ID from the VLDB entry.

By default, the Volume Server determines automatically whether or not it needs to create a new ReleaseClone:

- If there are no flags (`New release`, `Old release`, or `Not released`) on site definitions in the VLDB entry, the previous **vos release** command completed successfully and all read-only sites currently have the same volume. The Volume Server infers that the current **vos release** command was issued because the read/write volume has changed. The Volume Server creates a new ReleaseClone and distributes it to all of the read-only sites.

- If any site definition in the VLDB entry is marked with a flag, either the previous release operation did not complete successfully or a new read-only site was defined since the last release. The Volume Server does not create a new ReleaseClone, instead distributing the existing ReleaseClone to sites marked with the `Old release` or `Not released` flag. As previously noted, the VL Server marks each VLDB site definition with the `New release` flag as the site receives the ReleaseClone, and clears all flags after all sites successfully receive it.

To override the default behavior, forcing the Volume Server to create and release a new ReleaseClone to the read-only sites, include the **-f** flag. This is appropriate if, for example, the data at the read/write site has changed since the existing ReleaseClone was created during the previous release operation.

Using Read-only Volumes Effectively

For maximum effectiveness, replicate only volumes that satisfy two criteria:

- The volume's contents are heavily used. Examples include a volume housing binary files for text editors or other popular application programs, and volumes mounted along heavily traversed directory paths such as the paths leading to user home directories. It is an inefficient use of disk space to replicate volumes for which the demand is low enough that a single File Server can easily service all requests.
- The volume's contents change infrequently. As noted, file system consistency demands that the contents of read-only volumes must match each other and their read/write source at all times. Each time the read/write volume changes, you must issue the **vos release** command to update the read-only volumes. This can become tedious (and easy to forget) if the read/write volume changes frequently.

Explicitly mounting a read-only volume (creating a mount point that names a volume with a **.readonly** extension) is not generally necessary or appropriate. The Cache Manager has a built-in bias to access the read-only version of a replicated volume whenever possible. As described in more detail in "The Rules of Mount Point Traversal" on page 149, when the Cache Manager encounters a mount point it reads the volume name inside it and contacts the VL Server for a list of the sites that house the volume. In the normal case, if the mount point resides in a read-only volume and names a read/write volume (one that does not have a **.readonly** or **.backup** extension), the Cache Manager always attempts to access a read-only copy of the volume. Thus there is normally no reason to force the Cache Manager to access a read-only volume by mounting it explicitly.

It is a good practice to place a read-only volume at the read/write site, for a couple of reasons. First, the read-only volume at the read/write site requires only a small amount of disk space, because it is a clone rather a copy of all of the data (see "About Clones and Cloning" on page 138). Only if a large number of files are removed or changed in the read/write volume does the read-only copy occupy much disk space. That normally does not happen because the appropriate response to changes in a replicated read/write volume is to reclone it. The other reason to place a read-only volume at the read/write site is that the Cache Manager does not attempt to access the read/write version of a replicated volume if all read-only copies become inaccessible. If the file server machine housing the read/write volume is the only accessible machine, the Cache Manager can access the data only if there is a read-only copy at the read/write site.

The number of read-only sites to define depends on several factors. Perhaps the main trade-off is between the level of demand for the volume's contents and how much disk space you are willing to use for multiple copies of the volume. Of course, each prospective read-only site must have enough available space to accommodate the volume. The limit on the number of read-only copies of a volume is determined by the maximum number of site definitions in a volume's VLDB entry, which is defined in the *IBM AFS Release Notes*. The site housing the read/write and backup versions of the volume counts as one site, and each read-only site counts as an additional site (even the read-only site defined on the same file server machine and partition as the read/write site counts as a separate site). Note also that the Volume Server permits only one read-only copy of a volume per file server machine.

Replication Scenarios

The instructions in the following section explain how to replicate a volume for which no read-only sites are currently defined. However, you can also use the instructions in other common situations:

- If you are releasing a new clone to sites that already exist, you can skip Step "2" on page 142. It can still be useful to issue the **vos examine** command, however, to verify that the desired read-only sites are defined.
- If you are adding new read-only sites to existing ones, perform all of the steps. In Step "3" on page 143, issue the **vos addsite** command for the new sites only.
- If you are defining sites but do not want to release a clone to them yet, stop after Step "3" on page 143and continue when you are ready.
- If you are removing one or more sites before releasing a new clone to the remaining sites, follow the instructions for site removal in "Removing Volumes and their Mount Points" on page 180and then start with Step "4" on page 143.

To replicate a read/write volume (create a read-only volume)

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Select one or more sites at which to replicate the volume. There are several factors to consider:

   - How many sites are already defined. As previously noted, it is usually appropriate to define a read-only site at the read/write site. Also, the Volume Server permits only one read-only copy of a volume per file server machine. To display the volume's current sites, issue the **vos examine** command, which is described fully in "Displaying One Volume's VLDB Entry and Volume Header" on page 163.

     ```
     % vos examine <volume name or ID>
     ```

     The final lines of output display the volume's site definitions from the VLDB.

- Whether your cell dedicates any file server machines to housing read-only volumes only. In general, only very large cells use read-only server machines.
- Whether a site has enough free space to accommodate the volume. A read-only volume requires the same amount of space as the read/write version (unless it is at the read/write site itself). The first line of output from the **vos examine** command displays the read/write volume's current size in kilobyte blocks, as shown in "Displaying One Volume's VLDB Entry and Volume Header" on page 163.

  To display the amount of space available on a file server machine's partitions, use the **vos partinfo** command, which is described fully in "Creating Read/write Volumes" on page 135.

  ```
  % vos partinfo <machine name> [<partition name>]
  ```

3. Issue the **vos addsite** command to define each new read-only site in the VLDB.

   ```
   % vos addsite <machine name> <partition name> <volume name or ID>
   ```

   where

   ad

   Is the shortest acceptable abbreviation of **addsite**.

   machine name

   Defines the file server machine for the new site.

   partition name

   Names a disk partition on the machine machine name.

   volume name or ID

   Identifies the read/write volume to be replicated, either by its complete name or its volume ID number.

4. **(Optional)** Verify that the **fs** process (which incorporates the Volume Server) is functioning normally on each file server machine where you have defined a read-only site, and that the **vlserver** process (the Volume Location Server) is functioning correctly on each database server machine. Knowing that they are functioning eliminates two possible sources of failure for the release. Issue the **bos status** command on each file server machine housing a read-only site for this volume and on each database server machine. The command is described fully in "Displaying Process Status and Information from the BosConfig File" on page 113.

   ```
   % bos status <machine name> fs vlserver
   ```

5. Issue the **vos release** command to clone the read/write source volume and distribute the clone to each read-only site.

   ```
   % vos release <volume name or ID> [-f]
   ```

where

rel

Is the shortest acceptable abbreviation of **release**.

volume name or ID

Identifies the read/write volume to clone, either by its complete name or volume ID number. The read-only version is given the same name with a **.readonly** extension. All read-only copies share the same read-only volume ID number.

-f

Creates and releases a brand new clone.

6.

(Optional) Issue the **vos examine** command to verify that no site definition in the VLDB entry is marked with an `Old release` or `New release` flag. The command is described fully in "Displaying One Volume's VLDB Entry and Volume Header" on page 163.

```
% vos examine <volume name or ID>
```

If any flags appear in the output from Step "6" on page 144, repeat Steps "4" on page 143and "5" on page 143until the Volume Server does not produce any error messages during the release operation and the flags no longer appear. Do not issue the **vos release** command when you know that the read/write site or any read-only site is inaccessible due to network, machine or server process outage.

Creating Backup Volumes

A *backup volume* is a clone that resides at the same site as its read/write source (to review the concept of cloning, see "About Clones and Cloning" on page 138). Creating a backup version of a volume has two purposes:

- It is by convention the first step when dumping a volume's contents to tape with the AFS Backup System. A volume is inaccessible while it is being dumped, so instead of dumping the read/write volume, you create and dump a backup version. Users do not normally access the backup version, so it is unlikely that the dump will disturb them. For more details, see "Backing Up Data" on page 251.
- It enables users to restore mistakenly deleted or changed data themselves, freeing you for more crucial tasks. The backup version captures the state of its read/write source at the time the backup is made, and its contents cannot change. Mount the backup version in the filespace so that users can restore a file to its state at the time you made the backup. See "Making the Contents of Backup Volumes Available to Users" on page 146.

Backing Up Multiple Volumes at Once

The **vos backupsys** command creates a backup version of many read/write volumes at once. This command is useful when preparing for large-scale backups to tape using the AFS Backup System.

To clone every read/write volume listed in the VLDB, omit all of the command's options. Otherwise, combine the command's options to clone various groups of volumes. The options use one of two basic criteria to select volumes: location (the **-server** and **-partition** arguments) or presence in the volume name of one of a set of specified character strings (the **-prefix**, **-exclude**, and **-xprefix** options).

To clone only volumes that reside on one file server machine, include the **-server** argument. To clone only volumes that reside on one partition, combine the **-server** and **-partition** arguments. The **-partition** argument can also be used alone to clone volumes that reside on the indicated partition on every file server machine. These arguments can be combined with those that select volumes based on their names.

Combine the **-prefix**, **-exclude**, and **-xprefix** options (with or without the **-server** and **-partition** arguments) in the indicated ways to select volumes based on character strings contained in their names:

- To clone every read/write volume at the specified location whose name includes one of a set of specified character strings (for example, begins with **user.** or includes the string **afs**), use the **-prefix** argument or combine the **-xprefix** and **-exclude** options.
- To clone every read/write volume at the specified location except those whose name includes one of a set of specified character strings, use the **-xprefix** argument or combine the **-prefix** and **-exclude** options.
- To clone every read/write volume at the specified location whose name includes one of one of a set of specified character strings, except those whose names include one of a different set of specified character strings, combine the **-prefix** and **-xprefix** arguments. The command creates a list of all volumes that match the **-prefix** argument and then removes from the list the volumes that match the **-xprefix** argument. For effective results, the strings specified by the **-xprefix** argument must designate a subset of the volumes specified by the **-prefix** argument.

  If the **-exclude** flag is combined with the **-prefix** and **-xprefix** arguments, the command creates a list of all volumes that do not match the **-prefix** argument and then adds to the list any volumes that match the **-xprefix** argument. As when the **-exclude** flag is not used, the result is effective only if the strings specified by the **-xprefix** argument designate a subset of the volumes specified by the **-prefix** argument.

The **-prefix** and **-xprefix** arguments both accept multiple values, which can be used to define disjoint groups of volumes. Each value can be one of two types:

1. A simple character string, which matches volumes whose name begin with the string. All characters are interpreted literally (that is, characters that potentially have special meaning to the command shell, such as the period, have only their literal meaning).

2. A regular expression, which matches volumes whose names contain the expressions. Place a caret (**^**) at the beginning of the expression, and enclose the entire string in single quotes (**' '**). Explaining regular expressions is outside the scope of this reference page; see the UNIX manual page for **regexp(5)** or (for a brief introduction) "Defining and Displaying Volume Sets and Volume Entries" on page 209. As an example, the following expression matches volumes that have the string **aix** anywhere in their names:

   ```
   -prefix '^.*aix'
   ```

To display a list of the volumes to be cloned, without actually cloning them, include the **-dryrun** flag. To display a statement that summarizes the criteria being used to select volume, include the **-verbose** flag.

To back up a single volume, use the **vos backup** command, which employs a more streamlined technique for finding a single volume.

Automating Creation of Backup Volumes

Most cells find that it is best to make a new backup version of relevant volumes each day. It is best to create the backup versions at a time when usage is low, because the backup operation causes the read/write volume to be unavailable momentarily.

You can either issue the necessary the **vos backupsys** or **vos backup** commands at the console or create a **cron** entry in the **BosConfig** file on a file server machine, which eliminates the need for an administrator to initiate the backup operation.

The following example command creates a **cron** process called **backupusers** in the **/usr/afs/local/BosConfig** file on the machine **fs3.abc.com**. The process runs every day at 1:00 a.m. to create a backup version of every volume in the cell whose name starts with the string **user**. The **-localauth** flag enables the process to invoke the privileged **vos backupsys** command while unauthenticated. Note that the **-cmd** argument specifies a complete pathname for the **vos** binary, because the PATH environment variable for the BOS Server (running as the local superuser **root**) generally does not include the path to AFS binaries.

```
  % bos create fs3.abc.com backupusers cron\
-cmd "/usr/afs/bin/vos backupsys -prefix user -localauth" "1:00"
```

Making the Contents of Backup Volumes Available to Users

As noted, a backup volume preserves the state of the read/write source at the time the backup is created. Many cells choose to mount backup volumes so that users can access and restore data they have accidentally deleted or changed since the last backup was made, without having to request help from administrators. The most sensible place to mount the backup version of a user volume is at a subdirectory of the user's home directory. Suitable names for this directory include **OldFiles** and **Backup**. The subdirectory looks just like the user's own home directory as it was at the time the backup was created, with all files and subdirectories in the same relative positions.

If you do create and mount backup volumes for your users, inform users of their existence. The *IBM AFS User Guide* does not mention backup volumes because making them available to users is optional. Explain to users how often you make a new backup, so they know what they can recover. Remind them also that the data in their backup volume cannot change; however, they can use the standard UNIX **cp** command to copy it into their home volume and modify it there. Reassure users that the data in their backup volumes does not count against their read/write volume quota.

To create and mount a backup volume

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Verify that you have the **insert**(**i**) and **administer**(**a**) permissions on the ACL of the directory in which you wish to mount the volume. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

   Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

3. Issue the **vos backup** command to create a backup version of a read/write source volume. The message shown confirms the success of the backup operation.

   ```
   % vos backup <volume name or ID> Created backup volume for volume name or ID
   ```

   where

   backup

   Must be typed in full.

   volume name or ID

   Identifies the read/write volume to back up, either by its complete name or volume ID number. The backup volume has the same name with the addition of the **.backup** extension. It has its own volume ID number.

4. **(Optional)** Issue the **fs mkmount** to mount the backup volume. While this step is optional, Cache Managers cannot access the volume's contents if it is not mounted.

   ```
   % fs mkmount <directory> <volume name> .backup
   ```

   where

mk

Is the shortest acceptable abbreviation of **mkmount**.

directory

Names the mount point to create. Do not create a file or directory of the same name beforehand. Partial pathnames are interpreted relative to the current working directory. For the backup version of a user volume, the conventional location is the user's home directory.

volume name.backup

Is the full name of the backup volume.

5. **(Optional)** Issue the **fs lsmount** command to verify that the mount point refers to the correct volume. Complete instructions appear in "To display a mount point" on page 152.

```
% fs lsmount <directory>
```

To create multiple backup volumes at once

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **vos backupsys** command to create a backup version of every read/write volume that shares the same prefix or site. The effects of combining the three arguments are described in "Backing Up Multiple Volumes at Once" on page 145.

```
% vos backupsys [-prefix <common prefix on volume(s)>+] \
     [-server <machine name>] [-partition <partition name>] \
     [-exclude] [-xprefix <negative prefix on volume(s)>+] \
     [-dryrun] [-verbose]
```

where

backups

Is the shortest acceptable abbreviation of **backupsys**.

-prefix

Specifies one or more simple character strings or regular expressions of any length; a volume whose name includes the string is placed on the list of volumes to be cloned. Include field separators (such as periods) if appropriate. This argument can be combined with any combination of the **-server**, **-partition**, **-exclude**, and **-xprefix** options.

-server

Specifies the file server machine housing the volumes to backup. Can be combined with any combination of the **-prefix**, **-partition**, **-exclude**, and **-xprefix** options.

-partition

Specifies the partition housing the volumes you wish to backup. Can be combined with any combination of the **-prefix**, **-server**, **-exclude**, and **-xprefix** options.

-exclude

Indicates that all volumes except those indicated with the **-prefix** argument are to be backed up. The **-prefix** argument must be provided along with this one. Can also be combined with any combination of the **-prefix**, **-server**, and **-partition** arguments; or with both the **-prefix** and **-xprefix** arguments, but not with the **-xprefix** argument alone.

-xprefix

Specifies one or more simple character strings or regular expressions of any length; a volume whose name does not include the string is placed on the list of volumes to be cloned. Can be combined with any combination of the **-prefix**, **-server**, and **-partition** arguments; in addition, it can be combined with both the **-prefix** and **-exclude** options, but not with the **-exclude** flag alone.

-dryrun

Displays on the standard output stream a list of the volumes to be cloned, without actually cloning them.

-verbose

Displays on the standard output stream a statement that summarizes the criteria being used to select volumes, if combined with the **-dryrun** flag; otherwise, traces the cloning operation for each volume.

Mounting Volumes

Mount points make the contents of AFS volumes visible and accessible in the AFS filespace, as described in "About Mounting Volumes" on page 134. This section discusses in more detail how the Cache Manager handles mount points as it traverses the filespace. It describes the three types of mount points, their purposes, and how to distinguish between them, and provides instructions for creating, removing, and examining mount points.

The Rules of Mount Point Traversal

The Cache Manager observes three basic rules as it traverses the AFS filespace and encounters mount points:

- **Rule 1:** Access Backup and Read-only Volumes When Specified

  When the Cache Manager encounters a mount point that specifies a volume with either a **.readonly** or a **.backup** extension, it accesses that type of volume only. If a mount point does not have either a **.backup** or **.readonly** extension, the Cache Manager uses Rules 2 and 3.

  For example, the Cache Manager never accesses the read/write version of a volume if the mount point names the backup version. If the specified version is inaccessible, the Cache Manager reports an error.

- **Rule 2:** Follow the Read-only Path When Possible

  If a mount point resides in a read-only volume and the volume that it references is replicated, the Cache Manager attempts to access a read-only copy of the volume; if the referenced volume is not replicated, the Cache Manager accesses the read/write copy. The Cache Manager is thus said to prefer a *read-only* path through the filespace, accessing read-only volumes when they are available.

  The Cache Manager starts on the read-only path in the first place because it always accesses a read-only copy of the **root.afs** volume if it exists; the volume is mounted at the root of a cell's AFS filespace (named **/afs** by convention). That is, if the **root.afs** volume is replicated, the Cache Manager attempts to access a read-only copy of it rather than the read/write copy. This rule then keeps the Cache Manager on a read-only path as long as each successive volume is replicated. The implication is that both the **root.afs** and **root.cell** volumes must be replicated for the Cache Manager to access replicated volumes mounted below them in the AFS filespace. The volumes are conventionally mounted at the **/afs** and **/afs/***`cellname`* directories, respectively.

- **Rule 3:** Once on a Read/write Path, Stay There

  If a mount point resides in a read/write volume and the volume name does not have a **.readonly** or a **.backup** extension, the Cache Manager attempts to access only the a read/write version of the volume. The access attempt fails with an error if the read/write version is inaccessible, even if a read-only version is accessible. In this situation the Cache Manager is said to be on a *read/write path* and cannot switch back to the read-only path unless mount point explicitly names a volume with a **.readonly** extension. (Cellular mount points are an important exception to this rule, as explained in the following discussion.

The Three Types of Mount Points

AFS uses three types of mount points, each appropriate for a different purpose because of how the Cache Manager handles them.

- When the Cache Manager crosses a *regular* mount point, it obeys all three of the mount point traversal rules previously described.

AFS performs best when the vast majority of mount points in the filespace are regular, because the mount point traversal rules promote the most efficient use of both replicated and nonreplicated volumes. Because there are likely to be multiple read-only copies of a replicated volume, it makes sense for the Cache Manager to access one of them rather than the single read/write version, and the second rule leads it to do so. If a volume is not replicated, the third rule means that the Cache Manager still accesses the read/write volume when that is the only type available. In other words, a regular mount point does not force the Cache Manager always to access read-only volumes (it is explicitly not a "read-only mount point").

To create a regular mount point, use the **fs mkmount** command as described in "To create a regular or read/write mount point" on page 153.

> **Note:** To enable the Cache Manager to access the read-only version of a replicated volume named by a regular mount point, all volumes that are mounted above it in the pathname must also be replicated. That is the only way the Cache Manager can stay on a read-only path to the target volume.

- When the Cache Manager crosses a *read/write* mount point, it attempts to access only the volume version named in the mount point. If the volume name is the base (read/write) form, without a **.readonly** or **.backup** extension, the Cache Manager accesses the read/write version of the volume, even if it is replicated. In other words, the Cache Manager disregards the second mount point traversal rule when crossing a read/write mount point: it switches to the read/write path through the filespace.

  It is conventional to create only one read/write mount point in a cell's filespace, using it to mount the cell's **root.cell** volume just below the AFS filespace root (by convention, **/afs/.**`cellname`). As indicated, it is conventional to place a period at the start of the read/write mount point's name (for example, **/afs/.abc.com**). The period distinguishes the read/write mount point from the regular mount point for the **root.cell** volume at the same level. This is the only case in which it is conventional to create two mount points for the same volume. A desirable side effect of this naming convention for this read/write mount point is that it does not appear in the output of the UNIX **ls** command unless the **-a** flag is included, essentially hiding it from regular users who have no use for it.

  The existence of a single read/write mount point at this point in the filespace provides access to the read/write version of every volume when necessary, because it puts the Cache Manager on a read/write path right at the top of the filespace. At the same time, the regular mount point for the **root.cell** volume puts the Cache Manager on a read-only path most of the time.

  Using a read/write mount point for a read-only or backup volume is acceptable, but unnecessary. The first rule of mount point traversal already specifies that the Cache Manager accesses them if the volume name in a regular mount point has a **.readonly** or **.backup** extension.

  To create a read/write mount point, use the **-rw** flag on the **fs mkmount** command as described in "To create a regular or read/write mount point" on page 153.

- When the Cache Manager crosses a *cellular* mount point, it accesses the indicated volume in the specified cell, which is normally a foreign cell. (If the mount point does not name a cell along with the volume, the Cache Manager accesses the volume in the cell where the mount point resides.) When crossing a regular cellular mount point, the Cache Manager disregards the third mount point traversal rule. Instead, it accesses a read-only version of the volume if it is replicated, even if the volume that houses the mount point is read/write.

It is inappropriate to circumvent this behavior by creating a read/write cellular mount point, because traversing the read/write path imposes an unfair load on the foreign cell's file server machines. The File Server must issue a callback for each file fetched from the read/write volume, rather than single callback required for a read-only volume. In any case, only a cell's own administrators generally need to access the read/write versions of replicated volumes.

It is conventional to create cellular mount points only at the second level in a cell's filespace, using them to mount foreign cells' **root.cell** volumes just below the AFS filespace root (by convention, at **/afs/***`foreign_cellname`*). The mount point enables local users to access the foreign cell's filespace, assuming they have the necessary permissions on the ACL of the volume's root directory and that there is an entry for the foreign cell in each local client machine's **/usr/vice/etc/CellServDB** file, as described in "Maintaining Knowledge of Database Server Machines" on page 364.

Creating cellular mount points at other levels in the filespace and mounting foreign volumes other than the **root.cell** volume is not generally appropriate. It can be confusing to users if the Cache Manager switches between cells at various points in a pathname.

To create a regular cellular mount point, use the **-cell** argument to specify the cell name, as described in "To create a cellular mount point" on page 154.

To examine a mount point, use the **fs lsmount** command as described in "To display a mount point" on page 152. The command's output uses distinct notation to identify regular, read/write, and cellular mount points. To remove a mount point, use the **fs rmmount** command as described in "To remove a mount point" on page 156.

Creating a mount point in a foreign cell

Creating a mount point in a foreign cell's filespace (as opposed to mounting a foreign volume in the local cell) is basically the same as creating a mount point in the local filespace. The differences are that the **fs mkmount** command's directory argument specifies a pathname in the foreign cell rather than the local cell, and you must have the required permissions on the ACL of the foreign directory where you are creating the mount point. The **fs mkmount** command's **-cell** argument always specifies the cell in which the volume resides, not the cell in which to create the mount point.

To display a mount point

1. Issue the **fs lsmount** command.

   ```
   % fs lsmount <directory>
   ```

   where

ls

Is the shortest acceptable abbreviation of **lsmount**.

directory

Names the mount point to display.

If the specified directory is a mount point, the output is of the following form:

```
'directory' is a mount point for volume 'volume name'
```

For a regular mount point, a number sign (`#`) precedes the volume name string, as in the following example command issued on a client machine in the **abc.com** cell.

```
% fs lsmount /afs/abc.com/usr/terry
'/afs/abc.com/usr/terry' is a mount point for volume '#user.terry'
```

For a read/write mount point, a percent sign (`%`) precedes the volume name string, as in the following example command issued on a client machine in the **abc.com** cell. The cell's administrators have followed the convention of preceding the read/write mount point's name with a period.

```
% fs lsmount /afs/.abc.com
'/afs/.abc.com' is a mount point for volume '%root.cell'
```

For a cellular mount point, a cell name and colon (`:`) follow the number or percent sign and precede the volume name string, as in the following example command issued on a client machine in the **abc.com** cell.

```
% fs lsmount /afs/ghi.gov
'/afs/ghi.gov' is a mount point for volume '#ghi.gov:root.cell'
```

For a symbolic link to a mount point, the output is of the form shown in the following example command issued on a client machine in the **abc.com** cell.

```
% fs lsmount /afs/abc
'/afs/abc' is a symbolic link, leading to a mount point for volume '#root.cell'
```

If the directory is not a mount point or is not in AFS, the output reads as follows.

```
'directory' is not a mount point.
```

If the output is garbled, it is possible that the mount point has become corrupted in the local cache. Use the **fs flushmount** command as described in "To flush one or more mount points" on page 375. This forces the Cache Manager to refetch the mount point.

To create a regular or read/write mount point

1. Verify that you have the **i**(**insert**) and **a**(**administer**) permissions on the ACL of the directory where you are placing the mount point. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

2. Issue the **fs mkmount** command to create the mount point. Include the **-rw** flag if creating a read/write mount point.

   ```
   % fs mkmount <directory> <volume name> [-rw]
   ```

   where

   mk

   Is the shortest acceptable abbreviation for **mkmount**.

   directory

   Names the mount point to create. A file or directory with the same name cannot already exist. A partial pathname is interpreted relative to the current working directory.

   Specify the read/write path to the mount point, to avoid the failure that results when you attempt to create a new mount point in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

   volume name

   Specifies the volume's full name, including the **.backup** or **.readonly** extension for a backup or read-only volume, if appropriate.

   -rw

   Creates a read/write mount point.

To create a cellular mount point

1. Verify that you have the **i**(**insert**) and **a**(**administer**) permissions on the ACL of the directory where you are placing the mount point. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

2. If you are mounting one or more foreign cells' **root.cell** volume at the second level in your filespace and your cell's **root.afs** volume is replicated, you must create a temporary mount point for the **root.afs** volume's read/write version in a directory on which the ACL grants you the **i** and **a** permissions. The following command creates a mount point called **new_cells** in your cell's **/afs/.**`cellname` directory (the entry point to the read/write path in your cell).

   Substitute your cell's name for cellname.

   ```
   % cd /afs/.cellname
   % fs mkmount new_cells root.afs
   % cd new_cells
   ```

3. Issue the **fs mkmount** command with the **-cell** argument to create a cellular mount point. Repeat the command for each cellular mount point as required.

   ```
   % fs mkmount <directory> <volume name> -cell <cell name>
   ```

   where

   mk

   Is the shortest acceptable abbreviation for **mkmount**.

   directory

   Names the mount point to create. A file or directory with the same name cannot already exist. A partial pathname is interpreted relative to the current working directory. If you are mounting a foreign cell's **root.cell** volume, the standard value for this argument is the cell's complete Internet domain name.

   volume name

   Specifies the volume's full name, usually **root.cell** for a cellular mount point.

   -cell

   Specifies the complete Internet domain name of the cell in which the volume resides.

4. If you performed the instructions in Step "2" on page 155, issue the **vos release** command to release the new version of the **root.afs** volume to its read-only sites. (This command requires that you be listed in your cell's **/usr/afs/etc/UserList** file. If necessary, verify by issuing the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.)

   Also issue the **fs checkvolumes** command to force the local Cache Manager to access the new replica of the **root.afs** volume. If desired, you can also remove the temporary **new_cells** mount point from the **/afs/.**`cellname` directory.

   ```
   % vos release root.afs
   % fs checkvolumes
   % cd /afs/.cellname
   % fs rmmount new_cells
   ```

For your users to access a newly mounted foreign cell, you must also create an entry for it in each client machine's local **/usr/vice/etc/CellServDB** file and either reboot the machine or use the **fs newcell** command to insert the entry directly into its kernel memory. See the instructions in "Maintaining Knowledge of Database Server Machines" on page 364.

To remove a mount point

1. Verify that you have the **d**(**delete**) permission on the ACL of the directory from which you are removing the mount point. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

   Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

2. Issue the **fs rmmount** command to remove the mount point. The volume still exists, but its contents are inaccessible if this is the only mount point for it.

   ```
   % fs rmmount <directory>
   ```

   where

   rm

   Is the shortest acceptable abbreviation of **rmmount**.

   directory

   Names the mount point to remove. A partial pathname is interpreted relative to the current working directory.

   Specify the read/write path to the mount point, to avoid the failure that results when you attempt to delete a mount point from a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

Displaying Information About Volumes

This section explains how to display information about volumes. If you know a volume's name or volume ID number, there are commands for displaying its VLDB entry, its volume header, or both. Other commands display the name or location of the volume that contains a specified file or directory.

For instructions on displaying a volume's quota, see "Setting and Displaying Volume Quota and Current Size" on page 176.

Displaying VLDB Entries

The **vos listvldb** command displays the VLDB entry for the volumes indicated by the combination of arguments you provide. The possibilities are listed here from most to least inclusive:

- To display every entry in the VLDB, provide no arguments. It can take a long time to generate the output, depending on the number of entries.
- To display every VLDB entry that mentions a specific file server machine as the site of a volume, specify the machine's name with the **-server** argument.
- To display every VLDB entry that mentions a certain partition on any file server machine as the site of a volume, specify the partition name with the **-partition** argument.
- To display every VLDB entry that mentions a certain partition on a certain file server machine as the site of a volume, combine the **-server** and **-partition** arguments.
- To display a single VLDB entry, specify a volume name or ID number with the **-name** argument.
- To display the VLDB entry only for volumes with locked VLDB entries, use the **-locked** flag with any of the site definitions mentioned previously.

To display VLDB entries

1. Issue the **vos listvldb** command.

   ```
   % vos listvldb [-name <volume name or ID>] [-server <machine name>] \
         [-partition <partition name>] [-locked]
   ```

   where

   listvl

   Is the shortest acceptable abbreviation of **listvldb**.

   -name

   Identifies one volume either by its complete name or volume ID number. Do not combine this argument with the **-server** or **-partition** arguments.

-server

Specifies a file server machine. Combine this argument with the **-partition** argument if desired, but not with the **-name** argument.

-partition

Specifies a partition. Combine this argument with the **-server** argument if desired, but not with the **-name** argument.

-locked

Displays only locked VLDB entries. Combine this flag with any of the other options.

The VLDB entry for each volume includes the following information:

- The base (read/write) volume name. The read-only and backup versions have the same name with a **.readonly** and **.backup** extension, respectively.
- The volume ID numbers allocated to the versions of the volume that actually exist, in fields labeled `RWrite` for the read/write, `ROnly` for the read-only, `Backup` for the backup, and `RClone` for the ReleaseClone. (If a field does not appear, the corresponding version of the volume does not exist.) The appearance of the `RClone` field normally indicates that a release operation did not complete successfully; the `Old release` and `New release` flags often also appear on one or more of the site definition lines described just following.
- The number of sites that house a read/write or read-only copy of the volume, following the string `number of sites ->`.
- A line for each site that houses a read/write or read-only copy of the volume, specifying the file server machine, partition, and type of volume (`RW` for read/write or `RO` for read-only). If a backup version exists, it is understood to share the read/write site. Several flags can appear with a site definition:

  `Not released`

  Indicates that the **vos release** command has not been issued since the **vos addsite** command was used to define the read-only site.

  `Old release`

  Indicates that a **vos release** command did not complete successfully, leaving the previous, obsolete version of the volume at this site.

  `New release`

  Indicates that a **vos release** command did not complete successfully, but that this site did receive the correct new version of the volume.

- If the VLDB entry is locked, the string `Volume is currently LOCKED`.

For further discussion of the `New release` and `Old release` flags, see "Replicating Volumes (Creating Read-only Volumes)" on page 139.

An example of this command and its output for a single volume:

```
% vos listvldb user.terry
user.terry
    RWrite: 50489902    Backup: 50489904
    number of sites -> 1
       server fs3.abc.com partition /vicepc RW Site
```

Displaying Volume Headers

The **vos listvol** command displays the volume header for every volume on one or all partitions on a file server machine. The **vos** command interpreter obtains the information from the Volume Server on the specified machine. You can control the amount of information displayed by including one of the **-fast**, the **-long**, or the **-extended** flags described following the instructions in "To display volume headers" on page 159.

To display a single volume's volume header of one volume only, use the **vos examine** command as described in "Displaying One Volume's VLDB Entry and Volume Header" on page 163.

To display volume headers

1. Issue the **vos listvol** command.

   ```
   % vos listvol <machine name> [<partition name>] [-fast] [-long] [-extended]
   ```

   where

   listvo

   Is the shortest acceptable abbreviation of **listvol**.

   machine name

   Names the file server machine for which to display volume headers. Provide this argument alone or with the partition name argument.

   partition name

   Names one partition on the file server machine named by the machine name argument, which must be provided along with this one.

   -fast

   Displays only the volume ID numbers of relevant volumes. Do not combine this flag with the **-long** or **-extended** flag.

-long

Displays more detailed information about each volume. Do not combine this flag with the **-fast** or **-extended** flag.

-extended

Displays all of the information displayed by the **-long** flag, plus tables of statistics about reads and writes to the files in the volume. Do not combine this flag with the **-fast** or **-long** flag.

The output is ordered alphabetically by volume name and by default provides the following information on a single line for each volume:

- Name
- Volume ID number
- Type (the flag is `RW` for read/write, `RO` for read-only, `BK` for backup)
- Size in kilobytes (`1024` equals a megabyte)
- Number of files in the volume, if the **-extended** flag is provided
- Status on the file server machine, which is one of the following:

`On-line`

The volume is completely accessible to Cache Managers.

`Off-line`

The volume is not accessible to Cache Managers, but does not seem to be corrupted. This status appears while a volume is being dumped, for example.

`Off-line**needs salvage**`

The volume is not accessible to Cache Managers, because it seems to be corrupted. Use the **bos salvage** or **salvager** command to repair the corruption.

If the following message appears instead of the previously listed information, it indicates that a volume is not accessible to Cache Managers or the **vos** command interpreter, for example because a clone is being created.

```
**** Volume volume_ID is busy ****
```

If the following message appears instead of the previously listed information, it indicates that the File Server is unable to attach the volume, perhaps because it is seriously corrupted. The **FileLog** and **VolserLog** log files in the **/usr/afs/logs** directory on the file server machine possibly provide additional information; use the **bos getlog** command to display them.

```
**** Could not attach volume volume_ID ****
```

(For instructions on salvaging a corrupted or unattached volume, see "Salvaging Volumes" on page 172.)

The information about individual volumes is bracketed by summary lines. The first line of output specifies the number of volumes in the listing. The last line of output summarizes the number of volumes that are online, offline, and busy, as in the following example:

```
% vos listvol  fs2.abc.com /vicepb
Total number of volumes on server fs2.abc.com \
                                    partition /vicepb : 66
sys                  1969534847 RW       1582 K On-line
sys.backup           1969535105 BK       1582 K On-line
      .                    .     .          .   .    .
      .                    .     .          .   .    .
user.pat             1969534536 RW      17518 K On-line
user.pat.backup      1969534538 BK      17537 K On-line
Total volumes onLine 66 ; Total volumes offLine 0 ;  Total busy 0
```

Output with the -fast Flag

If you include the **-fast** flag displays only the volume ID number of each volume, arranged in increasing numerical order, as in the following example. The final line (which summarizes the number of on-line, off-line, and busy volumes) is omitted.

```
% vos listvol fs3.abc.com /vicepa -f
Total number of volumes on server fs3.abc.com  \
                                    partition /vicepa: 37
50489902
50489904
   .
   .
35970325
49732810
```

Output with the -long Flag

When you include the **-long** flag, , the output for each volume includes all of the information in the default listing plus the following. Each item in this list corresponds to a separate line of output:

- The file server machine and partition that house the volume, as determined by the command interpreter as the command runs, rather than derived from the VLDB or the volume header.
- The volume ID numbers associated with the various versions of the volume: read/write (`RWrite`), read-only (`ROnly`), backup (`Backup`), and ReleaseClone (`RClone`). One of them matches the volume ID number that appears on the first line of the volume's output. If the value in the `RWrite`, `ROnly`, or `Backup` field is `0` (zero), there is no volume of that type. If there is currently no ReleaseClone, the `RClone` field does not appear at all.
- The maximum space quota allotted to the read/write copy of the volume, expressed in kilobyte blocks in the `MaxQuota` field.
- The date and time the volume was created, in the `Creation` field. If the volume has been restored with the **backup diskrestore**, **backup volrestore**, or **vos restore** command, this is the restore time.
- The date and time when the contents of the volume last changed, in the `Last Update` field. For read-only and backup volumes, it matches the timestamp in the `Creation` field.

- The number of times the volume has been accessed for a fetch or store operation since the later of the two following times:
  - 12:00 a.m. on the day the command is issued
  - The last time the volume changed location

An example of the output when the **-long** flag is included:

```
% vos listvol fs2.abc.com b -long
Total number of volumes on server fs2.abc.com
                                    partition /vicepb: 66
      .                   .      .         .   .    .
      .                   .      .         .   .    .
user.pat             1969534536 RW      17518 K On-line
     fs2.abc.com /vicepb
     RWrite 1969534536 ROnly 0        Backup 1969534538
     MaxQuota      20000 K
     Creation    Mon Jun 12 09:02:25 1989
     Last Update Thu Jan  4 17:39:34 1990
     1573 accesses in the past day (i.e., vnode references)
user.pat.backup      1969534538 BK      17537 K On-line
     fs2.abc.com /vicepb
     RWrite 1969534536 ROnly 0        Backup 1969534538
     MaxQuota      20000 K
     Creation    Fri Jan  5 06:37:59 1990
     Last Update Fri Jan  5 06:37:59 1990
     0 accesses in the past day (i.e., vnode references)
         .               .         .     .       .
         .               .         .     .       .
Total volumes onLine 66 ; Total volumes offLine 0 ; Total busy 0
```

Output with the -extended Flag

When you include the **-extended** flag, the output for each volume includes all of the information reported with the **-long** flag, plus two tables of statistics:

- The table labeled `Raw Read/Write Stats` table summarizes the number of times the volume has been accessed for reading or writing.
- The table labeled `Writes Affecting Authorship` table contains information on writes made to files and directories in the specified volume.

An example of the output when the **-extended** flag is included:

```
% vos listvol fs3.abc.com a -extended
common.bboards   1969535592 RW    23149 K used 9401 files On-line
    fs3.abc.com /vicepa
    RWrite 1969535592 ROnly          0 Backup 1969535594
    MaxQuota      30000 K
    Creation    Mon Mar  8 14:26:05 1999
    Last Update Mon Apr 26 09:20:43 1999
    11533 accesses in the past day (i.e., vnode references)
                      Raw Read/Write Stats
              |-------------------------------------------|
              |    Same Network     |    Diff Network     |
              |----------|----------|----------|----------|
              |  Total   |   Auth   |  Total   |   Auth   |
              |----------|----------|----------|----------|
Reads         |      151 |      151 |     1092 |     1068 |
Writes        |        3 |        3 |      324 |      324 |
              |-------------------------------------------|
                   Writes Affecting Authorship
              |-------------------------------------------|
              |   File Authorship   | Directory Authorship|
              |----------|----------|----------|----------|
              |   Same   |   Diff   |   Same   |   Diff   |
              |----------|----------|----------|----------|
0-60 sec      |       92 |        0 |      100 |        4 |
1-10 min      |        1 |        0 |       14 |        6 |
10min-1hr     |        0 |        0 |       19 |        4 |
1hr-1day      |        1 |        0 |       13 |        0 |
1day-1wk      |        1 |        0 |        1 |        0 |
> 1wk         |        0 |        0 |        0 |        0 |
              |-------------------------------------------|
```

Displaying One Volume's VLDB Entry and Volume Header

The **vos examine** command displays information from both the VLDB and the volume header for a single volume. There is some redundancy in the information from the two sources, which allows you to compare the VLDB and volume header.

Because the volume header for each version of a volume (read/write, read-only, and backup) is different, you can specify which one to display. Include the **.readonly** or **.backup** extension on the volume name or ID argument as appropriate. The information from the VLDB is the same for all three versions.

To display one volume's VLDB entry and volume header

1. Issue the **vos examine** command.

   ```
   % vos examine <volume name or ID>
   ```

   where

   e

   Is the shortest acceptable abbreviation of **examine**.

volume name or ID

Identifies one volume either by its complete name or volume ID number. It can be a read/write, read-only, or backup type. Use the **.backup** or **.readonly** extension if appropriate.

The top part of the output displays the same information from a volume header as the **vos listvol** command with the **-long** flag, as described following the instructions in "To display volume headers" on page 159. If you specify the read-only version of the volume and it exists at more than one site, the output includes all of them. The bottom part of the output lists the same information from the VLDB as the **vos listvldb** command, as described following the instructions in "To display VLDB entries" on page 157.

Below is an example for a volume whose VLDB entry is currently locked.

```
% vos examine user.terry
user.terry                    536870981 RW   3459 K On-line
    fs3.abc.com /vicepa
    Write 5360870981   ROnly          0  Backup 536870983
    MaxQuota      40000 K
    Creation    Mon Jun 12 15:22:06 1989
    Last Update Fri Jun 16 09:34:35 1989
    5719 accesses in the past day (i.e., vnode references)
    RWrite: 5360870981   Backup: 536870983
    number of sites -> 1
       server fs3.abc.com partition /vicepa RW Site
    Volume is currently LOCKED
```

Displaying the Name or Location of the Volume that Contains a File

This section explains how to learn the name, volume ID number, or location of the volume that contains a file or directory.

You can also use one piece of information about a volume (for example, its name) to obtain other information about it (for example, its location). The following list points you to the relevant instructions:

- To use a volume’s name to learn the volume ID numbers of all its existing versions, use the **vos examine** command as described in "To display one volume’s VLDB entry and volume header" on page 163.

  You can also use the command to learn a volume’s name by providing its ID number.

- To use a volume’s name or ID number to learn its location, use the **vos listvldb** command as described in "To display VLDB entries" on page 157.

To display the name of the volume that contains a file

1. Issue the **fs listquota** command.

   ```
   % fs listquota [<dir/file path>]
   ```

   where

   lq

   Is an acceptable alias for **listquota**(and **listq** the shortest acceptable abbreviation).

   dir/file path

   Names a directory or file housed in the volume for which to display the name. Partial pathnames are interpreted relative to the current working directory, which is the default if this argument is omitted.

The following is an example of the output:

```
% fs listquota /afs/abc.com/usr/terry
Volume Name      Quota     Used     % Used   Partition
user.terry       15000     5071        34%         86%
```

To display the ID number of the volume that contains a file

1. Issue the **fs examine** command.

   ```
   % fs examine [<dir/file path>]
   ```

   where

   exa

   Is the shortest acceptable abbreviation of **examine**.

   dir/file path

   Names a directory or file housed in the volume for which to display the volume ID. Partial pathnames are interpreted relative to the current working directory, which is the default if this argument is omitted.

The following example illustrates how the output reports the volume ID number in the `vid` field.

```
% fs examine /afs/abc.com/usr/terry
Volume status for vid = 50489902 named user.terry
Current maximum quota is 15000
```

```
Current blocks used are 5073
The partition has 46383 blocks available out of 333305
```

Note: The partition-related statistics in this command's output do not always agree with the corresponding values in the output of the standard UNIX **df** command. The statistics reported by this command can be up to five minutes old, because the Cache Manager polls the File Server for partition information at that frequency. Also, on some operating systems, the **df** command's report of partition size includes reserved space not included in this command's calculation, and so is likely to be about 10% larger.

To display the location of the volume that contains a file

1. Issue the **fs whereis** command to display the name of the file server machine that houses the volume containing a file or directory.

   ```
   % fs whereis [<dir/file path>]
   ```

   where

   whe

   Is the shortest acceptable abbreviation of **whereis**.

   dir/file path

   Names a directory or file for which to report the location. Partial pathnames are interpreted relative to the current working directory, which is the default if this argument is omitted.

   The output displays the file server machine that houses the volume containing the file, as in the following example:

   ```
   % fs whereis /afs/abc.com/user/terry
   File /afs/abc.com/usr/terry is on host fs2.abc.com
   ```

2. If you also want to know which partition houses the volume, first issue the **fs listquota** command to display the volume's name. For complete syntax, see "To display the name of the volume that contains a file" on page 164.

   ```
   % fs listquota [<dir/file path>]
   ```

   Then issue the **vos listvldb** command, providing the volume name as the volume name or ID argument. For complete syntax and a description of the output, see "To display VLDB entries" on page 157.

   ```
   % vos listvldb <volume name or ID>
   ```

Moving Volumes

There are three main reasons to move volumes:

- To place volumes on other partitions or machines temporarily while repairing or replacing a disk or file server machine.
- To free space on a partition that is becoming overcrowded. One symptom of overcrowding is that users cannot to save files even though the relevant volume is below its quota. The following error message confirms the problem:

  ```
  afs: failed to store file (partition full)
  ```

  You can track available space on AFS server partitions by using the **scout** or **afsmonitor** programs described in "Monitoring and Auditing AFS Performance" on page 295.
- A file server machine is becoming overloaded because it houses many more volumes than other machines of the same size, or has volumes with more popular files in them.

To move a read/write volume, use the **vos move** command as described in the following instructions. Before attempting to move the volume, the **vos** command interpreter verifies that there is enough free space for it on the destination partition. If not, it does not attempt the move operation and prints the following message.

```
vos: no space on target partition destination_part to move volume volume
```

To move a read-only volume, you actually remove the volume from the current site by issuing the **vos remove** command as described in "To remove a volume and unmount it" on page 182. Then define a new site and release the volume to it by issuing the **vos addsite** and **vos release** commands as described in "To replicate a read/write volume (create a read-only volume)" on page 142.

A backup volume always resides at the same site as its read/write source volume, so you cannot move a backup volume except as part of moving the read/write source. The **vos move** command automatically deletes the backup version when you move a read/write volume. To create a new backup volume at the new site as soon as the move operation completes, issue the **vos backup** command as described in "To create and mount a backup volume" on page 147.

To move a read/write volume

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **vos move** command to move the volume. Type it on a single line; it appears on multiple lines here only for legibility.

   ```
   % vos move <volume name or ID> \ <machine name on source>
          <partition name on source > \ <machine name on destination> <partition name on
          destination>
   ```

where

m

Is the shortest acceptable abbreviation of **move**.

volume name or ID

Specifies the name or volume ID number of the read/write volume to move.

machine name on source

Names the file server machine currently housing the volume.

partition name on source

Names the partition currently housing the volume.

machine name on destination

Names the file server machine to which to move the volume.

partition name on destination

Names the partition to which to move the volume.

Note: It is best not to halt a **vos move** operation before it completes, because parts of the volume can be left on both the source and destination machines. For more information, see the command's reference page in the *IBM AFS Administration Reference*.

3. **(Optional)** Issue the **vos listvldb** command to confirm the success of the move. Complete instructions appear in "To display VLDB entries" on page 157.

```
% vos listvldb <volume name or ID>
```

4. If a backup version existed at the read/write volume's previous site, create a new backup at the new site by issuing the **vos backup** command, which is fully described in "To create and mount a backup volume" on page 147.

```
% vos backup <volume name or ID>
```

Synchronizing the VLDB and Volume Headers

AFS can provide transparent file access because the Volume Location Database (VLDB) constantly tracks volume locations. When the Cache Manager needs a file, it contacts the Volume Location (VL) Server, which reads the VLDB for the current location of the volume containing the file. Therefore, the VLDB must accurately reflect the state of volumes on the file server machines at all times. The Volume

Server and VL Server automatically update a volume's VLDB entry when its status changes during a **vos** operation, by performing the following series of steps.

1. The VL Server locks the VLDB entry. The lock advises other operations not to manipulate any of the volume versions (read/write, read-only, or backup), which prevents the inconsistency that can result from multiple simultaneous operations.
2. The VL Server sets an *intention flag* in the VLDB entry that indicates the kind of operation to be performed. This flag never appears in VLDB listings because it is for internal use only. In case the operation terminates prematurely, this flag tells the Salvager which operation was interrupted. (The Salvager then determines the steps necessary either to complete the operation or return the volume to a previous consistent state. For more information on salvaging, see "Salvaging Volumes" on page 172.)
3. The Volume Server manipulates the volume. It usually sets the `Off-line` flag in the volume header, which makes the volume inaccessible to the File Server and other Volume Server operations during the manipulation. When the operation completes, the volume is again marked `On-line`.
4. The VL Server records any changes resulting from the operation in the VLDB entry. Once the operation is complete, it removes the intention flag set in Step "2" on page 169and releases the lock set in Step "1" on page 169.

If a **vos** operation fails while the Volume Server is manipulating the volume (corresponding to Step "3" on page 169), the volume can be left in an intermediate state, which is termed *corruption*. In this case, the `Off-line` or `Off-line**needs salvage**` marker usually appears at the end of the first line of output from the **vos examine** command. To repair the corruption, run the Salvager before attempting to resynchronize the VLDB and volume headers. For salvaging instructions, see "Salvaging Volumes" on page 172.

More commonly, an interruption while flags are being set or removed (corresponding to Step "1" on page 169, Step "2" on page 169, or Step "4" on page 169) causes a discrepancy between the VLDB and volume headers. To resynchronize the VLDB and volumes, use the **vos syncvldb** and **vos syncserv** commands. To achieve complete VLDB consistency, it is best to run the **vos syncvldb** command on all file server machines in the cell, and then run the **vos syncserv** command on all file server machines in the cell.

There are several symptoms that indicate a volume operation failed:

- Error messages on the standard error stream or in server process log files indicate that an operation terminated abnormally. Perhaps you had to halt the operation before it completed (for instance, by using a signal such as **Ctrl-c**), or a file server machine or server process was not functioning when the operation ran. To determine if a machine or process is still not functioning, issue the **bos status** command as described in "Displaying Process Status and Information from the BosConfig File" on page 113.
- A subsequent **vos** operation fails because a previous failure left a VLDB entry locked. Sometimes an error message reports that a volume is locked. To display a list of locked volumes, use the **-locked** flag on the **vos listvldb** command as described in "Displaying VLDB Entries" on page 157.

  If the only problem with a volume is that its VLDB entry is locked, you probably do not need to synchronize the entire VLDB. Instead use the **vos unlock** or **vos unlockvldb** command to unlock the entry, as described in "Unlocking and Locking VLDB Entries" on page 191.

- A subsequent **vos** operation fails because a previous failure left a volume marked as offline. To check a volume's current status, check the first line of output from the **vos examine** command as described in "Displaying One Volume's VLDB Entry and Volume Header" on page 163.

The **vos syncvldb** command corrects the information in the Volume Location Database (VLDB) either about all volumes housed on a file server machine, about the volumes on just one partition, or about a single volume. If checking about one or more partitions, the command contacts the Volume Server to obtain a list of the volumes that actually reside on each partition. It then obtains the VLDB entry for each volume from the VL Server. It changes the VLDB entry as necessary to reflect the state of the volume on the partition. For example, it creates or updates a VLDB entry when it finds a volume for which the VLDB entry is missing or incomplete. However, if there is already a VLDB entry that defines a different location for the volume, or there are irreconcilable conflicts with other VLDB entries, it instead writes a message about the conflict to the standard error stream. The command never removes volumes from the file server machine.

When checking a single volume's VLDB entry, the command also automatically performs the operations invoked by the **vos syncserv** command: it not only verifies that the VLDB entry is correct for the specified volume type (read/write, backup, or read-only), but also checks that any related volume types mentioned in the VLDB entry actually exist at the site listed in the entry.

The **vos syncserv** command verifies that each volume type (read/write, read-only, and backup) mentioned in a VLDB entry actually exists at the site indicated in the entry. It checks all VLDB entries that mention a site either on any of a file server machine's partitions or on one partition. Note that command can end up inspecting sites other than on the specified machine or partition, if there are read-only versions of the volume at sites other than the read/write site.

The command alters any incorrect information in the VLDB, unless there is an irreconcilable conflict with other VLDB entries. In that case, it writes a message to the standard error stream instead. The command never removes volumes from their sites.

To synchronize the VLDB with volume headers

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **vos syncvldb** command to make the VLDB reflect the true state of all volumes on a machine or partition, or the state of one volume.

   Note: To synchronize the VLDB completely, issue the command repeatedly, substituting each file server machine in your cell for the **-server** argument in turn and omitting the **-partition** and **-volume** arguments, before proceeding to Step "3" on page 171.

```
% vos syncvldb -server <machine name> [-partition <partition name>] \
     [-volume <volume name or ID>] [-verbose >> file]
```

where

syncv

Is the shortest acceptable abbreviation of **syncvldb**.

-server

Names the file server machine housing the volumes for which to verify VLDB entries. If you are also providing the **-volume** argument, this argument must name the machine where the volume actually resides.

-partition

Identifies the partition (on the file server machine specified by the **-server** argument) housing the volumes for which to verify VLDB entries. In general, it is best to omit this argument so that either the VLDB entries for all volumes on a server machine are corrected (if you do not provide the **-volume** argument), or so that you do not need to guarantee that the partition actually houses the volume named by the **-volume** argument.

-volume

Specifies the name or volume ID number of a single volume for which to verify the VLDB entry.

-verbose >> file

Directs a detailed trace to the file called file, which can be either in AFS or on the local disk of the machine on which you are issuing the command. The command often writes a large amount of output to the standard output stream; writing it to a file enables you to examine the output more carefully.

3. Issue the **vos syncserv** command to inspect each volume for which the VLDB lists a version at the specified site.

   Note: To synchronize the VLDB completely, issue the command repeatedly, substituting each file server machine in your cell for the machine name argument in turn and omitting the partition name argument.

```
% vos syncserv <machine name> [<partition name>] [-v >> file]
```

where

syncs

Is the shortest acceptable abbreviation of **syncserv**.

machine name

Names the file server machine mentioned in each VLDB entry to check.

partition name

Identifies the partition mentioned in each VLDB entry to check. If synchronizing the entire VLDB, omit this argument.

-v >> file

Directs a detailed trace to the file called file, which can be either in AFS or on the local disk of the machine on which you are issuing the command. The command often writes a large amount of output to the standard output stream; writing it to a file enables you to examine the output more carefully.

Salvaging Volumes

An unexpected interruption while the Volume Server or File Server is manipulating the data in a volume can leave the volume in an intermediate state (*corrupted*), rather than just creating a discrepancy between the information in the VLDB and volume headers. For example, the failure of the operation that saves changes to a file (by overwriting old data with new) can leave the old and new data mixed together on the disk.

If an operation halts because the Volume Server or File Server exits unexpectedly, the BOS Server automatically shuts down all components of the **fs** process and invokes the Salvager. The Salvager checks for and repairs any inconsistencies it can. Sometimes, however, there are symptoms of the following sort, which indicate corruption serious enough to create problems but not serious enough to cause the File Server component to fail. In these cases you can invoke the Salvager yourself by issuing the **bos salvage** command.

- **Symptom:** A file appears in the output of the **ls** command, but attempts to access the file fail with messages indicating that it does not exist.

  Possible cause: The Volume Server or File Server exited in the middle of a file-creation operation, after changing the directory structure, but before actually storing data. (Other possible causes are that the ACL on the directory does not grant the permissions you need to access the file, or there is a process, machine, or network outage. Check for these causes before assuming the file is corrupted.)

  Salvager's solution: Remove the file's entry from the directory structure.

- **Symptom:** A volume is marked `Off-line` in the output from the **vos examine** and **vos listvol** commands, or attempts to access the volume fail.

  Possible cause: Two files or versions of a file are sharing the same disk blocks because of an interrupted operation. The File Server and Volume Server normally refuse to attach volumes that exhibit this type of corruption, because it can be very dangerous. If the Volume Server or File Server do attach the volume but are unsure of the status of the affected disk blocks, they sometimes try to write yet more data there. When they cannot perform the write, the data is lost. This effect can cascade, causing loss of all data on a partition.

Salvager's solution: Delete the data from the corrupted disk blocks in preference to losing an entire partition.

- **Symptom:** There is less space available on the partition than you expect based on the size statistic reported for each volume by the **vos listvol** command.

  Possible cause: There are orphaned files and directories. An *orphaned* element is completely inaccessible because it is not referenced by any directory that can act as its parent (is higher in the file tree). An orphaned element is not counted in the calculation of a volume's size (or against its quota), even though it occupies space on the server partition.

  Salvager's solution: By default, print a message to the **/usr/afs/logs/SalvageLog** file reporting how many orphans were found and the approximate number of kilobytes they are consuming. You can use the **-orphans** argument to remove or attach orphaned elements instead. See "To salvage volumes" on page 174.

When you notice symptoms such as these, use the **bos salvage** command to invoke the Salvager before corruption spreads. (Even though it operates on volumes, the command belongs to the **bos** suite because the BOS Server must coordinate the shutdown and restart of the Volume Server and File Server with the Salvager. It shuts them down before the Salvager starts, and automatically restarts them when the salvage operation finishes.)

All of the AFS data stored on a file server machine is inaccessible during the salvage of one or more partitions. If you salvage just one volume, it alone is inaccessible.

When processing one or more partitions, the command restores consistency to corrupted read/write volumes where possible. For read-only or backup volumes, it inspects only the volume header:

- If the volume header is corrupted, the Salvager removes the volume completely and records the removal in its log file, **/usr/afs/logs/SalvageLog**. Issue the **vos release** or **vos backup** command to create the read-only or backup volume again.
- If the volume header is intact, the Salvager skips the volume (does not check for corruption in the contents). However, if the File Server notices corruption as it initializes, it sometimes refuses to attach the volume or bring it online. In this case, it is simplest to remove the volume by issuing the **vos remove** or **vos zap** command. Then issue the **vos release** or **vos backup** command to create it again.

Combine the **bos salvage** command's arguments as indicated to salvage different numbers of volumes:

- To salvage all volumes on a file server machine, combine the **-server** argument and the **-all** flag.
- To salvage all volumes on one partition, combine the **-server** and **-partition** arguments.
- To salvage only one read/write volume, combine the **-server**, **-partition**, and **-volume** arguments. Only that volume is inaccessible to Cache Managers, because the BOS Server does not shutdown the File Server and Volume Server processes during the salvage of a single volume. Do not name a read-only or backup volume with the **-volume** argument. Instead, remove the volume, using the **vos**

remove or **vos zap** command. Then create a new copy of the volume with the **vos release** or **vos backup** command.

The Salvager always writes a trace to the **/usr/afs/logs/SalvageLog** file on the file server machine where it runs. To record the trace in another file as well (either in AFS or on the local disk of the machine where you issue the **bos salvage** command), name the file with the **-file** argument. Or, to display the trace on the standard output stream as it is written to the **/usr/afs/logs/SalvageLog** file, include the **-showlog** flag.

By default, multiple Salvager subprocesses run in parallel: one for each partition up to four, and four subprocesses for four or more partitions. To increase or decrease the number of subprocesses running in parallel, provide a positive integer value for the **-parallel** argument.

If there is more than one server partition on a physical disk, the Salvager by default salvages them serially to avoid the inefficiency of constantly moving the disk head from one partition to another. However, this strategy is often not ideal if the partitions are configured as logical volumes that span multiple disks. To force the Salvager to salvage logical volumes in parallel, provide the string **all** as the value for the **-parallel** argument. Provide a positive integer to specify the number of subprocesses to run in parallel (for example, **-parallel 5all** for five subprocesses), or omit the integer to run up to four subprocesses, depending on the number of logical volumes being salvaged.

The Salvager creates temporary files as it runs, by default writing them to the partition it is salvaging. The number of files can be quite large, and if the partition is too full to accommodate them, the Salvager terminates without completing the salvage operation (it always removes the temporary files before exiting). Other Salvager subprocesses running at the same time continue until they finish salvaging all other partitions where there is enough disk space for temporary files. To complete the interrupted salvage, reissue the command against the appropriate partitions, adding the **-tmpdir** argument to redirect the temporary files to a local disk directory that has enough space.

The **-orphans** argument controls how the Salvager handles orphaned files and directories that it finds on server partitions it is salvaging. An orphaned element is completely inaccessible because it is not referenced by the vnode of any directory that can act as its parent (is higher in the filespace). Orphaned objects occupy space on the server partition, but do not count against the volume's quota.

During the salvage, the output of the **bos status** command reports the following auxiliary status for the **fs** process:

```
Salvaging file system
```

To salvage volumes

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos salvage** command to salvage one or more volumes.

   ```
   % bos salvage  -server <machine name>  [-partition <salvage partition>]  \
                  [-volume <salvage volume number or volume name>]  \
                  [-file salvage log output file]  [-all]  [-showlog]  \
                  [-parallel <# of max parallel partition salvaging>]  \
   ```

```
                 [-tmpdir <directory to place tmp files>]  \
                 [-orphans <ignore | remove | attach >]
```

where

-server

Names the file server machine on which to salvage volumes. This argument can be combined either with the **-all** flag, the **-partition** argument, or both the **-partition** and **-volume** arguments.

-partition

Names a single partition on which to salvage all volumes. The **-server** argument must be provided along with this one.

-volume

Specifies the name or volume ID number of one read/write volume to salvage. Combine this argument with the **-server** and **-partition** arguments.

-file

Specifies the complete pathname of a file into which to write a trace of the salvage operation, in addition to the **/usr/afs/logs/SalvageLog** file on the server machine. If the file pathname is local, the trace is written to the specified file on the local disk of the machine where the **bos salvage** command is issued. If the **-volume** argument is included, the file can be in AFS, though not in the volume being salvaged. Do not combine this argument with the **-showlog** flag.

-all

Salvages all volumes on all of the partitions on the machine named by the **-server** argument.

-showlog

Displays the trace of the salvage operation on the standard output stream, as well as writing it to the **/usr/afs/logs/SalvageLog** file.

-parallel

Specifies the maximum number of Salvager subprocesses to run in parallel. Provide one of three values:

- An integer from the range **1** to **32**. A value of **1** means that a single Salvager process salvages the partitions sequentially.
- The string **all** to run up to four Salvager subprocesses in parallel on partitions formatted as logical volumes that span multiple physical disks. Use this value only with such logical volumes.
- The string **all** followed immediately (with no intervening space) by an integer from the range **1** to **32**, to run the specified number of Salvager subprocesses in parallel on partitions formatted as logical volumes. Use this value only with such logical volumes.

The BOS Server never starts more Salvager subprocesses than there are partitions, and always starts only one process to salvage a single volume. If this argument is omitted, up to four Salvager subprocesses run in parallel.

-tmpdir

Specifies the full pathname of a local disk directory to which the Salvager process writes temporary files as it runs. By default, it writes them to the partition it is currently salvaging.

-orphans

Controls how the Salvager handles orphaned files and directories. Choose one of the following three values:

ignore

Leaves the orphaned objects on the disk, but prints a message to the **/usr/afs/logs/SalvageLog** file reporting how many orphans were found and the approximate number of kilobytes they are consuming. This is the default if you omit the **-orphans** argument.

remove

Removes the orphaned objects, and prints a message to the **/usr/afs/logs/SalvageLog** file reporting how many orphans were removed and the approximate number of kilobytes they were consuming.

attach

Attaches the orphaned objects by creating a reference to them in the vnode of the volume's root directory. Since each object's actual name is now lost, the Salvager assigns each one a name of the following form:
_ _ORPHANFILE_ _. index for files
_ _ORPHANDIR_ _. index for directories

where index is a two-digit number that uniquely identifies each object. The orphans are charged against the volume's quota and appear in the output of the **ls** command issued against the volume's root directory.

Setting and Displaying Volume Quota and Current Size

Every AFS volume has an associated quota which limits the volume's size. The default quota for a newly created volume is 5,000 kilobyte blocks (slightly less that 5 MB). When a volume reaches its quota, the File Server rejects attempts to create new files or directories in it. If an application is writing data into an existing file in a full volume, the File Server allows a defined overage (by default, 1 MB). (You can use the **fileserver** command's **-spare** or **-pctspare** argument to change the default overage; see the command's reference page in the *IBM AFS Administration Reference*.)

To set a quota other than 5000 KB as you create a volume, include the **-maxquota** argument to the **vos create** command, as described in "Creating Read/write Volumes" on page 135. To modify an existing volume's quota, issue either the **fs setquota** or the **fs setvol** command as described in the following instructions. Do not set an existing volume's quota lower than its current size.

In general, smaller volumes are easier to administer than larger ones. If you need to move volumes, say for load-balancing purposes, it is easier to find enough free space on other partitions for small volumes. Move operations complete more quickly for small volumes, reducing the potential for outages or other errors to interrupt the move. AFS supports a maximum volume size, which can vary for different AFS releases; see the *IBM AFS Release Notes* for the version you are using. Also, the size of a partition or logical places an absolute limit on volume size, because a volume cannot span multiple partitions or logical volumes.

It is generally safe to overpack partitions by putting more volumes on them than can actually fit if all the volumes reach their maximum quota. However, only experience determines to what degree overpacking works in your cell. It depends on what kind of quota you assign to volumes (particularly user volumes, which are more likely than system volumes to grow unpredictably) and how much information people generate and store in comparison to their quota.

There are several commands that display a volume's quota, as described in the following instructions. They differ in how much related information they produce.

To set quota for a single volume

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **fs setquota** command to set the volume's maximum quota.

   ```
   % fs setquota [<dir/file path>] -max <max quota in kbytes>
   ```

   where

   sq

   Is an acceptable alias for **setquota**.

   dir/file path

   Names a file or directory in the volume for which to set the indicated quota. Partial pathnames are interpreted relative to the current working directory, which is the default if you omit this argument.

   Specify the read/write path to the file or directory, to avoid the failure that results when you attempt to change a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

max quota in kbytes

Sets the volume's quota, expressed in kilobyte blocks (**1024** equals a megabyte). A value of **0** grants an unlimited quota, but the size of the partition imposes an absolute limit. You must include the **-max** switch if omitting the dir/file path argument (to set the quota on the volume that houses the current working directory).

To set maximum quota on one or more volumes

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

```
% pts membership system:administrators
```

2. Issue the **fs setvol** command to set the quota on one or more volumes.

```
% fs setvol [<dir/file path>+] -max <disk space quota in 1K units>
```

where

sv

Is an acceptable alias for **setvol**.

dir/file path

Names one file or directory that resides in each volume for which to set the indicated quota. Partial pathnames are interpreted relative to the current working directory, which is the default if you omit this argument.

disk space quota in 1K units

Sets the maximum quota on each volume, expressed in kilobytes blocks (**1024** equals a megabyte). A value of **0** grants an unlimited quota, but the size of the partition does impose an absolute limit.

To display percent quota used

1. Issue the **fs quota** command.

```
% fs quota [<dir/file path>+]
```

where

q

Is the shortest acceptable abbreviation of **quota**.

dir/file path

Names a directory or file in each volume for which to display percent quota used. Partial pathnames are interpreted relative to the current working directory, which is the default if you omit this argument.

The following example illustrates the output produced by this command:

```
% fs quota /afs/abc.com/usr/terry
34% of quota used.
```

To display quota, current size, and other information

1. Issue the **fs listquota** command.

   ```
   % fs listquota [<dir/file path>+]
   ```

   where

   lq

   Is an alias for **listquota**.

   dir/file path

   Names a directory or file in each volume for which to display quota along with volume name and current space usage. Partial pathnames are interpreted relative to the current working directory, which is the default if you omit this argument.

As illustrated in the following example, the output reports the volume's name, its quota and current size (both in kilobyte units), the percent quota used, and the percentage of space on the volume's host partition that is used.

```
% fs listquota /afs/abc.com/usr/terry
Volume Name      Quota     Used    % Used   Partition
user.terry       15000     5071       34%         86%
```

To display quota, current size, and more partition information

1. Issue the **fs examine** command.

   ```
   % fs examine [<dir/file path>+]
   ```

   where

   exa

   Is the shortest acceptable abbreviation of **examine**.

   dir/file path

   Names a directory or file in each volume for which to display quota information and information about the host partition. Partial pathnames are interpreted relative to the current working directory, which is the default if you omit this argument.

As illustrated in the following example, the output displays the volume's volume ID number and name, its quota and current size (both in kilobyte units), and the free and total number of kilobyte blocks on the volume's host partition.

```
% fs examine /afs/abc.com/usr/terry
Volume status for vid = 50489902 named user.terry
Current maximum quota is 15000
Current blocks used are 5073
The partition has 46383 blocks available out of 333305
```

Note: The partition-related statistics in this command's output do not always agree with the corresponding values in the output of the standard UNIX **df** command. The statistics reported by this command can be up to five minutes old, because the Cache Manager polls the File Server for partition information at that frequency. Also, on some operating systems, the **df** command's report of partition size includes reserved space not included in this command's calculation, and so is likely to be about 10% larger.

Removing Volumes and their Mount Points

To remove a volume from its site and its record from the VLDB, use the **vos remove** command. Use it to remove any of the three types of volumes; the effect depends on the type.

- If you indicate the read/write volume by specifying the volume's base name without a **.readonly** or **.backup** extension, the command removes both the read/write and associated backup volume from the partition that houses them. You do not need to provide the **-server** and **-partition** arguments, because there can be only one read/write site. The site information is also removed from the VLDB entry, and the site count (reported by the **vos examine** and **vos listvldb** commands as `number of sites`) decrements by one. The read/write and backup volume ID numbers no longer appear in the output from the **vos examine** and **vos listvldb** commands, but they are preserved internally. Read-only sites, if any, are not affected, but cannot be changed unless a read/write site is again defined. The entire VLDB entry is removed if there are no read-only sites.

If there are no read-only copies left, it is best to remove the volume's mount point to prevent attempts to access the volume's contents. Do not remove the mount point if copies of the read-only volume remain.

- If you indicate a read-only volume by including the **.readonly** extension on its name, it is removed from the partition that houses it, and the corresponding site information is removed from the VLDB entry. The site count reported by the **vos examine** and **vos listvldb** commands as `number of sites` decrements by one for each volume you remove.

  If there is more than one read-only site, you must include the **-server** argument (and optionally **-partition** argument) to specify the site from which to remove the volume. If there is only one read-only site, the volume name is sufficient; if no read/write volume exists in this case, the entire VLDB entry is removed.

  It is not generally appropriate to remove the volume's mount point when removing a read-only volume, especially if the read/write version of the volume still exists. If the read/write version no longer exists, remove the mount point as described in Step "5" on page 183of "To remove a volume and unmount it" on page 182.

- If you indicate a backup volume by including the **.backup** extension on its name, it is removed from the partition that houses it and its site information is removed from the VLDB entry. You do not need to provide the **-server** and **-partition** arguments, because there can be only one backup site. The backup volume ID number no longer appears in the output from the **vos examine** or **vos listvldb** command, but is preserved internally.

  In the standard configuration, there is a separate mount point for the backup version of a user volume. Remember to remove the mount point to prevent attempt to access the nonexistent volume's contents.

Other Removal Commands

The **vos remove** command is almost always the appropriate way to remove a volume, because it automatically removes a volume's VLDB entry and both the volume header and all data from the partition. If either the VLDB entry or volume header does not exist, it is sometimes necessary to use other commands that remove only the remaining element. Do not use these commands in the normal case when both the VLDB entry and the volume header exist, because by definition they create discrepancies between them. For details on the commands' syntax, see their reference pages in the *IBM AFS Administration Reference*.

The **vos zap** command removes a volume from its site by removing the volume header and volume data for which a VLDB entry no longer exists. You can tell a VLDB entry is missing if the **vos listvol** command displays the volume header but the **vos examine** or **vos listvldb** command cannot locate the VLDB entry. You must run this command to correct the discrepancy, because the **vos syncvldb** and **vos syncserv** commands never remove volume headers.

The **vos remsite** command removes a read-only site definition from the VLDB without affecting the volume on the file server machine. Use this command when you have mistakenly issued the **vos addsite** command to define a read-only site, but have not yet issued the **vos release** command to release the volume to the site. If you have actually released a volume to the site, use the **vos remove** command instead.

The **vos delentry** command removes the entire VLDB entry that mentions the volume you specify. If versions of the volume actually exist on file server machines, they are not affected. This command is useful if you know for certain that a volume removal was not recorded in the VLDB (perhaps you used the **vos zap** command during an emergency), and do not want to take the time to resynchronize the entire VLDB with the **vos syncvldb** and **vos syncserv** commands.

To remove a volume and unmount it

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If removing the volume's mount point, verify that you have the **d**(**delete**) permission on its parent directory's ACL. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

   Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

3.

   (Optional) Dump the volume to a file or to tape, in case you want to restore it later. To copy the volume's contents to a file, use the **vos dump** command as instructed in "Dumping and Restoring Volumes" on page 183. You can then copy the file to tape using a third-party backup utility or an archiving utility such as the UNIX **tar** command.

   Alternatively, use the AFS Backup System to create a tape copy. In this case, it can be convenient to create a temporary volume set that includes only the volume of interest. Temporary volume sets are not recorded in the Backup Database, and so do not clutter database with records for volume sets that you use only once. For instructions, see "To create a dump" on page 263.

4. Issue the **vos remove** command to remove the volume. If removing a read-only volume from multiple sites, repeat the command for each one.

   ```
   % vos remove [-server machine name>] [-partition <partition name>] \
                -id <volume name or ID>
   ```

where

remo

Is the shortest acceptable abbreviation of **remove**.

-server

Specifies the file server machine on which the volume resides. It is necessary only when the **-id** argument names a read-only volume that exists at multiple sites.

-partition

Specifies the partition on machine name where the volume resides. It is necessary only when the **-id** argument names a read-only volume that exists at multiple sites. Provide the **-server** argument along with this one.

-id

Identifies the volume to remove, either by its complete name or volume ID number. If identifying a read-only or backup volume by name, include the appropriate extension (**.readonly** or **.backup**).

5. If you are removing the last existing version of the volume, issue the **fs rmmount** command remove the corresponding mount point. Complete instructions appear in "To remove a volume and unmount it" on page 182.

   If you are removing a backup volume that is mounted in the conventional way (at a subdirectory of its read/write volume's root directory), then removing the source volume's mount point in this step is sufficient to remove the backup volume's mount point. If you mounted the backup at a completely separate directory, you need to repeat this step for the backup volume's mount point.

   ```
   % fs rmmount <directory>
   ```

6. **(Optional)** If you created a dump file in Step "3" on page 182, transfer it to tape. The preferred method is to use the AFS Backup System, which is described in "Configuring the AFS Backup System" on page 195and "Backing Up and Restoring AFS Data" on page 241.

Dumping and Restoring Volumes

Dumping a volume with the **vos dump** command converts its contents into ASCII format and writes them to the file you specify. The **vos restore** command places a dump file's contents into a volume after converting them into the volume format appropriate for the indicated file server machine.

About Dumping Volumes

Dumping a volume can be useful in several situations, including the following:

- You want to back it up to tape, perhaps by using a third-party backup utility. To facilitate this type of backup operation, the **vos dump** command can write to a named pipe. To learn about using the AFS Backup System instead, see "Configuring the AFS Backup System" on page 195and "Backing Up and Restoring AFS Data" on page 241.
- You are removing the volume from your cell (perhaps because its owner is leaving your cell). The **vos dump** command enables you to create a copy for safekeeping without incurring the overhead of the Backup System. For complete instructions on removing a volume, see "Removing Volumes and their Mount Points" on page 180.
- You want to create a copy of the volume for safekeeping on a non-AFS server partition, perhaps while you move the actual volume to another machine or perform maintenance tasks on the partition that houses the volume.
- You need to replace a corrupted read/write volume. If an uncorrupted read-only or backup version of the volume exists, dump it and restore the data into the read/write volume, overwriting the corrupted contents.
- You want to copy or transfer the contents of the volume to another cell. You cannot use the **vos move** command, because AFS supports volume moves only between file server machines that belong to the same cell.
- You want to have another read/write copy of the volume's contents. The second volume must have a different name than the original one. If you want the contents of the two volumes to remain identical, you must update them both manually. AFS provides no facility for keeping read/write volumes synchronized in this way.
- You want a copy of only the files and directories in the volume with modification time stamps after a certain date. The **vos dump** command can create an incremental dump file as described in Step "3" on page 185of the following instructions.

You can use the **vos dump** command to create a *full dump*, which contains the complete contents of the volume at the time you issue the command, or an *incremental dump*, which contains only those files and directories with modification timestamps (as displayed by the **ls -l** command) that are later than a date and time you specify. See Step "3" on page 185of the following instructions.

Dumping a volume does not change its VLDB entry or permanently affect its status on the file server machine, but the volume's contents are inaccessible during the dump operation. To avoid interrupting access to the volume, it is generally best to dump the volume's backup version, just after using the **vos backup** or **vos backupsys** command to create a new backup version.

If you do not provide a filename into which to write the dump, the **vos dump** command directs the output to the standard output stream. You can pipe it directly to the **vos restore** command if you wish.

Because a volume dump file is in ASCII format, you can read its contents using a text editor or a command such as the **cat** command. However, dump files sometimes contain special characters that do not have alphanumeric correlates, which can cause problems for some display programs.

By default, the **vos** command interpreter consults the Volume Location Database (VLDB) to learn the volume's location, so the **-server** and **-partition** arguments are not required. If the **-id** argument identifies a read-only volume that resides at multiple sites, then the command dumps the version from just one of them (normally, the one listed first in the volume's VLDB entry as reported by the **vos examine** or **vos listvldb** command). To dump the read-only volume from a particular site, use the **-server** and **-partition** arguments to specify the site. To bypass the VLDB lookup entirely, provide a volume ID number (rather than a volume name) as the value for the **-id** argument, along with the **-server** and **-partition** arguments. This makes it possible to dump a volume for which there is no VLDB entry.

To dump a volume

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Verify that you have the permissions necessary to create the dump file. If placing it in AFS, you must have the **i**(**insert**) permission on the ACL of the file's directory. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

   Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

3. Issue the **vos dump** command to dump the volume.

   ```
   % vos dump -id <volume name or ID> [-time <dump from time>] [-file <arg>] [-server <s
   ```

   where

   -id

   Identifies the volume to be dumped by its complete name or volume ID number. If you want to dump the read-only or backup version, specify its volume ID number or add the appropriate extension (**.readonly** or **.backup**) to the name.

   To bypass the normal VLDB lookup of the volume's location, provide the volume ID number and combine this argument with the **-server** and **-partition** arguments.

   -time

   Specifies whether the dump is full or incremental. Omit this argument to create a full dump, or provide one of three acceptable values:

   - The value **0**(zero) to create a full dump.
   - A date in the format mm **/** dd **/** yyyy (month, day and year) to create an incremental dump that includes only files and directories with modification timestamps later than midnight

(12:00 a.m.) on the indicated date. Valid values for the year range from **1970** to **2037**; higher values are not valid because the latest possible date in the standard UNIX representation is in 2038. The command interpreter automatically reduces later dates to the maximum value. An example is **01/13/1999**.

- A date and time in the format **"** mm **/** dd **/** yyyy hh **:** MM **"** to create an incremental dump that includes only files and directories with modification timestamps later than the specified date and time. The date format is the same as for a date alone. Express the time as hours and minutes (hh:MM) in 24-hour format (for example, **20:30** is 8:30 p.m.). Surround the entire expression with double quotes (" ") because it contains a space. An example is **"01/13/1999 22:30"**.

-file

Specifies the pathname of the file to which to write the dump. The file can be in AFS, but not in the volume being dumped. A partial pathname is interpreted relative to the current working directory. Omit this argument to direct the dump to the standard output stream.

-server

Specifies the file server machine on which the volume resides. Provide the **-partition** argument along with this one.

-partition

Specifies the partition on which the volume resides. Provide the **-server** argument along with this one.

About Restoring Volumes

Although you can dump any of the three types of volumes (read/write, read-only, or backup), you can restore a dump file to the file system only as a read/write volume, using the **vos restore** command. The command automatically translates the dump file's contents from ASCII back into the volume format appropriate for the file server machine that stores the restored version. As with the **vos dump** command, you can restore a dump file via a named pipe, which facilitates interoperation with third-party backup utilities.

You can restore the contents of a dump file in one of two basic ways. In either case, you must restore a full dump of the volume before restoring any incremental dumps. Any incremental dumps that you then restore must have been created after the full dump. If there is more than one incremental dump, you must restore them in the order they were created.

- You can restore volume data into a brand new volume with a new name and at a location that you specify. See "To restore a dump into a new volume and mount it" on page 187.

  You can assign a volume ID number as you restore the volume, though it is best to have the Volume Server allocate a volume number automatically. The most common reason for specifying the volume

ID is that a volume's VLDB entry has disappeared for some reason, but you know the former read/write volume ID number and want to reuse it.

- You can restore volume data into an existing volume (usually the one that was previously dumped), overwriting its current contents. This is convenient if the current contents are corrupted or otherwise incorrect, because it allows you to replace them with a coherent version from the past or from one of the volume's clones. See "To restore a dump file, overwriting an existing volume" on page 189.

  Provide the **-overwrite** argument to preconfirm that you wish to overwrite the volume's contents, and to specify whether you are restoring a full or incremental dump. If you omit the **-overwrite** argument, the Volume Server generates the following prompt to confirm that you want to overwrite the existing volume with either a full (**f**) or incremental (**i**) dump:

  ```
  Do you want to do a full/incremental restore or abort? [fia](a):
  ```

  If you pipe in the dump file via the standard input stream instead of using the **-file** argument to name it, you must include the **-overwrite** argument because there is nowhere for the Volume Server to display the prompt in this case.

  You can move the volume to a new site as you overwrite it with a full dump, by using the **-server** and **-partition** arguments to specify the new site. You cannot move the volume when restoring an incremental dump.

The **vos restore** command sets the restored volume's creation date in the volume header to the time of the restore operation, as reported in the `Creation` field in the output from the **vos examine** and **vos listvol** commands.

To restore a dump into a new volume and mount it

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Verify that you have permissions needed to read the dump file and to mount the new volume. If the dump file resides in AFS, you need the **r**(**read**) permission on the ACL of its directory. You need the **i**(**insert**) and **a**(**administer**) permissions on the ACL of the directory where you are mounting the new volume. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

   Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

3. Select a site (disk partition on a file server machine) for the new volume. If your cell groups different types of volumes onto different file server machines, that can guide your decision. It often makes sense to put the volume on the emptiest partition that meets your other criteria. To display how much space is available on a file server machine's partitions, use the **vos partinfo** command, which is described fully in "Creating Read/write Volumes" on page 135.

   ```
   % vos partinfo <machine name> [<partition name>]
   ```

4. Issue the **vos restore** command to create a new volume and restore the dump file into it. Type it on a single line; it appears on multiple lines here only for legibility.

   ```
   % vos restore <machine name> <partition name> \
                 <name of volume to be restored> \
                 [-file <dump file>] [-id <volume ID>]
   ```

   where

   res

   Is the shortest acceptable abbreviation of **restore**.

   machine name

   Names the file server machine on which to create the new volume.

   partition name

   Names the partition on which to create the new volume.

   name of volume to be restored

   Names the new read/write volume, which must not already have a VLDB entry. It can be up to 22 characters in length.

   -file

   Is the dump file to restore. Partial pathnames are interpreted with respect to the current working directory. Omit this argument if using a pipe to read in the dump file from the standard input stream.

   -volume

   Specifies the new volume's ID number. It is appropriate only if you are restoring a volume that no longer exists and want to use the volume ID number it had previously.

5. Issue the **fs mkmount** command to mount the new volume, making its contents accessible. Complete instructions appear in "To create a regular or read/write mount point" on page 153.

   ```
   % fs mkmount <directory> <volume name>
   ```

6. **(Optional)** Issue the **fs lsmount** command to verify that the mount point refers to the correct volume. Complete instructions appear in "To display a mount point" on page 152.

   ```
   % fs lsmount <directory>
   ```

To restore a dump file, overwriting an existing volume

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Verify that you have permissions needed to read the dump file. If it resides in AFS, you need the **r**(**read**) permission on the ACL of its directory. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

   Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

3. Restore the contents of the dump file into a read/write volume, overwriting the current contents. The volume retains its current volume ID number. Type it on a single line; it appears on multiple lines here only for legibility.

   ```
   % vos restore <machine name> <partition name>  \
                 <name of volume to be restored>   \
                 [-file <dump file>] [-id <volume ID>]
   ```

   where

   res

   Is the shortest acceptable abbreviation of **restore**.

   machine name

   Names the file server machine where the volume already exists, or the machine to which to move it. In the latter case, the value for the **-overwrite** argument must be **full**.

   partition name

   Names the partition where the volume already exists, or the partition to which to move it. In the latter case, the value for the **-overwrite** argument must be **full**.

   name of volume to be restored

   Names the read/write volume to overwrite with the contents of the dump file.

   -file

   Is the dump file to restore. Partial pathnames are interpreted with respect to the current working directory. Omit this argument if using a pipe to read in the dump file from the standard input stream; in this case, you must provide the **-overwrite** argument.

-overwrite

Preconfirms that you want to overwrite the existing volume and specifies which type of dump file you are restoring. Provide one of the following values:

- **f** or **full** if restoring a full dump file
- **i** or **incremental** if restoring an incremental dump file. This value is not acceptable if you are moving the volume while restoring it.
- **a** to terminate the restore operation

4. If the volume is replicated, issue the **vos release** command to release the newly restored contents to read-only sites. Complete instructions appear in "Replicating Volumes (Creating Read-only Volumes)" on page 139.

```
% vos release <volume name or ID>
```

5. Issue the **vos backup** command to create a new backup version of the volume. Complete instructions appear in "Creating Backup Volumes" on page 144.

```
% vos backup <volume name or ID>
```

Renaming Volumes

You can use the **vos rename** command to rename a volume. For example, it is appropriate to rename a user's home volume if you use the **user.** username convention for user volume names and you change the username. (For complete instructions for changing usernames, see "Changing Usernames" on page 478.)

The **vos rename** command accepts only read/write volume names, but automatically changes the names of the associated read-only and backup volumes. As directed in the following instructions, you need to replace the volume's current mount point with a new one that reflects the name change.

To rename a volume

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Verify that you have the **a**(**administer**), **d**(**delete**), and **i**(**insert**) access permissions for the directory in which you are replacing the volume's mount point. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

```
% fs listacl [<dir/file path>]
```

Members of the **system:administrators** group always implicitly have the **a**(**administer**) and by default also the **l**(**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

3. Issue the **vos rename** command to rename the volume.

   ```
   % vos rename <old volume name> <new volume name>
   ```

   where

   ren

   Is the shortest acceptable abbreviation of **rename**.

   old volume name

   Is the current name of a read/write volume.

   new volume name

   Is the new name for the volume. It cannot be more than 22 characters in length.

   If there is no Volume Location Database (VLDB) entry for the specified current volume name, the command fails with the following error message:

   ```
   vos: Could not find entry for volume old_volume_name.
   ```

4. Issue the **fs rmmount** command to remove the mount point that refers to the volume's old name. Complete instructions appear in "To remove a mount point" on page 156.

   ```
   % fs rmmount <directory>
   ```

5. Issue the **fs mkmount** to create a mount point that indicates the volume's new name. Complete instructions appear in "To create a regular or read/write mount point" on page 153.

   ```
   % fs mkmount <directory> <volume name> [-rw]
   ```

Unlocking and Locking VLDB Entries

As detailed in "Synchronizing the VLDB and Volume Headers" on page 168, The Volume Location (VL) Server locks the Volume Location Database (VLDB) entry for a volume before the Volume Server executes any operation on it. No other operation can affect a volume with a locked VLDB entry, so the lock prevents the inconsistency or corruption that can result from multiple simultaneous operations on a volume.

To verify that a VLDB entry is locked, issue the **vos listvldb** command as described in "To display VLDB entries" on page 157. The command has a **-locked** flag that displays locked entries only. If the VLDB entry is locked, the string `Volume is currently LOCKED` appears on the last line of the volume's output.

To lock a VLDB entry yourself, use the **vos lock** command. This is useful when you suspect something is wrong with a volume and you want to prevent any changes to it while you are investigating the problem.

To unlock a locked VLDB entry, issue the **vos unlock** command, which unlocks a single VLDB entry, or the **vos unlockvldb** command, which unlocks potentially many entries. This is useful when a volume operation fails prematurely and leaves a VLDB entry locked, preventing you from acting to correct the problems resulting from the failure.

To lock a VLDB entry

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **vos lock** to lock the entry.

   ```
   % vos lock <volume name or ID>
   ```

   where

   lo

   Is the shortest acceptable abbreviation of **lock**.

   volume name or ID

   Identifies the volume to be locked, either by its complete name or volume ID number. It can be any of the three versions of the volume.

To unlock a single VLDB entry

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **vos unlock** command to unlock the entry.

   ```
   % vos unlock <volume name or ID>
   ```

   where

   unlock

   Must be typed in full.

   volume name or ID

   Identifies the volume to be unlocked, either by its complete name or volume ID number. It can be any of the three versions of the volume.

To unlock multiple VLDB entries

1. Verify that you are listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **vos unlockvldb** command to unlock the desired entries.

   ```
   % vos unlockvldb [<machine name>] [<partition name>]
   ```

   where

   unlockv

   Is the shortest acceptable abbreviation of **unlockvldb**.

   machine name

   Specifies a file server machine. Provide this argument alone to unlock all VLDB entries that mention the machine in a site definition. Omit both this argument and the partition name argument to unlock all VLDB entries.

   partition name

   Specifies a partition. Provide this argument alone to unlock all VLDB entries that mention the partition (on any machine) in a site definition. Omit both this argument and the machine name argument to unlock all VLDB entries.

Chapter 6. Configuring the AFS Backup System

The AFS Backup System helps you to create backup copies of data from AFS volumes and to restore data to the file system if it is lost or corrupted. This chapter explains how to configure the Backup System. For instructions on backing up and restoring data and displaying dump records, see "Backing Up and Restoring AFS Data" on page 241.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Determine tape capacity and filemark size	**fms**
Define Tape Coordinator entry in Backup Database	**backup addhost**
Remove Tape Coordinator entry from Backup Database	**backup delhost**
Display Tape Coordinator entries from Backup Database	**backup listhosts**
Create volume set	**backup addvolset**
Add volume entry to volume set	**backup addvolentry**
List volume sets and entries	**backup listvolsets**
Delete volume set from Backup Database	**backup delvolset**
Delete volume entry from volume set	**backup delvolentry**
Define dump level	**backup adddump**
Change expiration date on existing dump level	**backup setexp**
Delete dump level from dump hierarchy	**backup deldump**
Display dump hierarchy	**backup listdumps**
Label tape	**backup labeltape**
Read label on tape	**backup readlabel**

Introduction to Backup System Features

The AFS Backup System is highly flexible, enabling you to control most aspects of the backup process, including how often backups are performed, which volumes are backed up, and whether to dump all of the data in a volume or just the data that has changed since the last dump operation. You can also take advantage of several features that automate much of the backup process.

To administer and use the Backup System most efficiently, it helps to be familiar with its basic features, which are described in the following sections. For pointers to instructions for implementing the features as you configure the Backup System in your cell, see "Overview of Backup System Configuration" on page 200.

Volume Sets and Volume Entries

When you back up AFS data, you specify which data to include in terms of complete volumes rather than individual files. More precisely, you define groups of volumes called *volume sets*, each of which includes one or more volumes that you want to back up in a single operation. You must include a volume in a volume set to back it up, because the command that backs up data (the **backup dump** command) does not accept individual volume names.

A volume set consists of one or more *volume entries*, each of which specifies which volumes to back up based on their location (file server machine and partition) and volume name. You can use a wildcard notation to include all volumes that share a location, a common character string in their names, or both.

For instructions on creating and removing volume sets and volume entries, see "Defining and Displaying Volume Sets and Volume Entries" on page 209.

Dumps and Dump Sets

A *dump* is the collection of data that results from backing up a volume set. A *full dump* includes all of the data in every volume in the volume set, as it exists at the time of the dump operation. An *incremental dump* includes only some of the data from the volumes in the volume set, namely those files and directory structures that have changed since a specified previous dump operation was performed. The previous dump is referred to as the incremental dump's *parent dump*, and it can be either a full dump or an incremental dump itself.

A *dump set* is a collection of one or more dumps stored together on one or more tapes. The first dump in the dump set is the *initial dump*, and any subsequent dump added onto the end of an existing dump set is an *appended dump*. Appending dumps is always optional, but maximizes use of a tape's capacity. In contrast, creating only initial dumps can result in many partially filled tapes, because an initial dump must always start on a new tape, but does not necessarily extend to the end of the tape. Appended dumps do not have to be related to one another or to the initial dump (they do not have to be dumps of the same or related volume sets), but well-planned appending can reduce the number of times you have to change tapes during a restore operation. For example, it can make sense to append incremental dumps of a volume set together in a single dump set.

All the records for a dump set are indexed together in the Backup Database based on the initial dump (for more on the Backup Database, see "The Backup Database and Backup Server Process" on page 199). To delete the database record of an appended dump, you must delete the initial dump record, and doing so deletes the records for all dumps in the dump set. Similarly, you cannot recycle just one tape in a dump set without deleting the database records of all tapes in the dump set.

For instructions on creating an initial dump, see "Backing Up Data" on page 251, and to learn how to append dumps, see "Appending Dumps to an Existing Dump Set" on page 260.

Dump Hierarchies, Dump Levels and Expiration Dates

A *dump hierarchy* is a logical structure that defines the relationship between full and incremental dumps; that is, it defines which dump serves as the parent for an incremental dump. Each individual component of a hierarchy is a *dump level*. When you create a dump by issuing the **backup dump** command, you specify a volume set name and a dump level name. The Backup System uses the dump level to determine whether the dump is full or incremental, and if incremental, which dump level to use as the parent.

You can associate an *expiration date* with a dump level, to define when a dump created at that level expires. The Backup System refuses to overwrite a tape until all dumps in the dump set to which the tape belongs have expired, so assigning expiration dates automatically determines how you recycle tapes. You can define an expiration date either in absolute terms (for example, 13 January 2000) or relative terms (for example, 30 days from when the dump is created). You can also change the expiration date associated with a dump level (but not with an actual dump that has already been created at that level).

For instructions on creating dump hierarchies, assigning expiration dates, and establishing a tape recycling schedule, see "Defining and Displaying the Dump Hierarchy" on page 215.

Dump Names and Tape Names

When you create a dump, the Backup System creates a Backup Database record for it, assigning a name comprising the volume set name and the last element in the dump level pathname:

```
volume_set_name.dump_level_name
```

For example, a dump of the volume set **user** at the dump level **/sunday/friday** is called **user.friday**. The Backup System also assigns a unique *dump ID* number to the dump to distinguish it from other dumps with the same name that possibly exist.

The Backup System assigns a similar *AFS tape name* to each tape that contains a dump set, reflecting the volume set and dump level of the dump set's initial dump, plus a numerical index of the tape's position in the dump set, and a unique dump ID number:

```
volume_set_name.dump_level_name.tape_index (dump ID)
```

For example, the second tape in a dump set whose initial dump is of the volume set **uservol** at the dump level **/sunday/friday** has AFS tape name like **uservol.friday.2** (**914382400**).

In addition to its AFS tape name, a tape can have an optional *permanent name* that you assign. Unlike the AFS tape name, the permanent name does not have to indicate the volume set and dump level of the initial (or any other) dump, and so does not change depending on the contents of the tape. The Backup System does not require a certain format for permanent names, so you need to make sure that each tape's name is unique. If a tape has a permanent name, the Backup System uses it rather than the AFS tape name when referring to the tape in prompts and the output from most **backup** commands, but still tracks the AFS tape name internally.

Tape Labels, Dump Labels, and EOF Markers

Every tape used in the Backup System has a magnetic label at the beginning that records the tape's name, capacity, and other information. You can use the **backup labeltape** command to write a label, or the **backup dump** command creates one automatically if you use an unlabeled tape. The label records the following information:

- The tape's permanent name, which you can assign by using the **-pname** argument to the **backup labeltape** command. It can be any string of up to 32 characters. If you do not assign a permanent name, the Backup System records the value `<NULL>` when you use the **backup labeltape** command to assign an AFS tape name, or when you use the **backup dump** command to write a dump to the tape.
- The tape's AFS *tape name*, which can be one of three types of values:
  - A name that reflects the volume set and dump level of the dump set's initial dump and the tape's place in the sequence of tapes for the dump set, as described in "Dump Names and Tape Names" on

page 197. If the tape does not have a permanent name, you can assign the AFS tape name by using the **-name** argument to the **backup labeltape** command.

- The value `<NULL>`, which results when you assign a permanent name, or provide no value for the **backup labeltape** command's **-name** argument.
- No AFS tape name at all, indicating that you have never labeled the tape or written a dump to it.

If a tape does not already have an actual AFS tape name when you write a dump to it, the Backup System constructs and records the appropriate AFS tape name. If the tape does have an AFS tape name and you are writing an initial dump, then the name must correctly reflect the dump's volume set and dump level.

- The capacity, or *size*, of the tape, followed by a letter that indicates the unit of measure (`k` or `K` for kilobytes, `m` or `M` for megabytes, `g` or `G` for gigabytes, or `t` or `T` for terabytes). The tape's manufacturer determines the tape's capacity. For further discussion of how the Backup System uses the value in the capacity field, see "Configuring the tapeconfig File" on page 200.

For information about labeling tapes, see "Writing and Reading Tape Labels" on page 223.

In addition to the tape label, the Backup System writes a *dump label* on the tape for every appended dump (the tape label and dump label are the same for the initial dump). A dump label records the following information:

- The name of the tape containing the dump
- The date and time that the dump operation began
- The cell to which the volumes in the dump belong
- The dump's size in kilobytes
- The dump's dump level
- The dump's dump ID

The Backup System writes a *filemark* (also called an End-of-File or EOF marker) between the data from each volume in a dump. The tape device's manufacturer determines the filemark size, which is typically between 2 KB and 2 MB; in general, the larger the usual capacity of the tapes that the device uses, the larger the filemark size. If a dump contains a small amount of data from each of a large number of volumes, as incremental dumps often do, then the filemark size can significantly affect how much volume data fits on the tape. To enable the Backup System to factor in filemark size as it writes a dump, you can record the filemark size in a configuration file; see "Configuring the tapeconfig File" on page 200.

Tape Coordinator Machines, Port Offsets, and Backup Data Files

A *Tape Coordinator machine* is a machine that drives one or more attached tape devices used for backup operations. It must run the AFS client software (the Cache Manager) but reside in a physically secure location to prevent unauthorized access to its console. Before backup operations can run on a Tape Coordinator machine, each tape device on the machine must be registered in the Backup Database, and certain files and directories must exist on the machine's local disk; for instructions, see "To configure a Tape Coordinator machine" on page 205.

Each tape device on a Tape Coordinator machine listens for backup requests on a different UNIX port. You pick the port indirectly by assigning a *port offset number* to the tape device. The Backup System sets the device's actual port by adding the port offset to a base port number that it determines internally. For instructions on assigning port offset numbers, see "Configuring the tapeconfig File" on page 200.

For a tape device to perform backup operations, a Backup Tape Coordinator (**butc**) process dedicated to the device must be running actively on the Tape Coordinator machine. You then direct backup requests to the device's Tape Coordinator by specifying its port offset number with the **-portoffset** argument to the **backup** command.

In addition to writing backup data to tape, you can direct it to a *backup data file* on the local disk of a Tape Coordinator machine. You can then to transfer the data to a data-archiving system, such as a hierarchical storage management (HSM) system, that you use in conjunction with AFS and the Backup System. A backup data file has a port offset like a tape device. For instructions on configuring backup data files, see "Dumping Data to a Backup Data File" on page 236.

The Backup Database and Backup Server Process

The *Backup Database* is a replicated administrative database maintained by the Backup Server process on the cell's database server machines. Like the other AFS database server processes, the *Backup Server* uses the Ubik utility to keep the various copies of the database synchronized (for a discussion of Ubik, see "Replicating the AFS Administrative Databases" on page 31).

The Backup Database records the following information:

- The Tape Coordinator machine's hostname and the port offset number for each tape device used for backup operations
- The dump hierarchy, which consists of its component dump levels and their associated expiration dates
- The volume sets and their component volume entries
- A record for each dump, which includes the name of each tape it appears on, a list of the volumes from which data is included, the dump level, the expiration date, and the dump ID of the initial dump with which the dump is associated
- A record for each tape that houses dumped data

Interfaces to the Backup System

The **backup** suite of commands is the administrative interface to the Backup System. You can issue the commands in a command shell (or invoke them in a shell script) on any AFS client or server machine from which you can access the **backup** binary. In the conventional configuration, the binary resides on the local disk.

The **backup** command suite provides an *interactive mode*, in which you can issue multiple commands over a persistent connection to the Backup Server and the Volume Location (VL) Server. Interactive mode has several convenient features, including the following:

- You need to type only the operation code, omitting the initial **backup** string.
- If you assume another AFS identity or specify a foreign cell as you enter interactive mode, it applies to all subsequent commands.
- You do not need to enclose shell metacharacters in double quotes.
- You can track current and pending operations with the **(backup) jobs** command, which is available only in this mode.
- You can cancel current and pending operations with the **(backup) kill** command, which is available only in this mode.

Before issuing a command that requires reading or writing a tape (or backup data file), you must also open a connection to the Tape Coordinator machine that is attached to the relevant tape device (or that has the backup data file on its local disk), and issue the **butc** command to initialize the Tape Coordinator process. The process must continue to run and the connection remain open as long as you need to use the tape device or file for backup operations.

For further discussion and instructions, see "Using the Backup System's Interfaces" on page 241.

Overview of Backup System Configuration

Before you can use the Backup System to back up and restore data, you must configure several of its basic components. The indicated sections of this chapter explain how to perform the following configuration tasks:

- Determining a tape's capacity and a tape device's filemark size, and recording them in the **/usr/afs/backup/tapeconfig** file (see "Configuring the tapeconfig File" on page 200)
- Determining how to grant administrative privilege to backup operators (see "Granting Administrative Privilege to Backup Operators" on page 204)
- Configuring Tape Coordinator machines, tape devices, and backup data files (see "Configuring Tape Coordinator Machines and Tape Devices" on page 205)
- Defining volume sets and volume entries (see "Defining and Displaying Volume Sets and Volume Entries" on page 209)
- Defining dump levels to create a dump hierarchy (see "Defining and Displaying the Dump Hierarchy" on page 215)
- Labeling tapes (see "Writing and Reading Tape Labels" on page 223)
- Creating a device configuration file to automate the backup process (see "Automating and Increasing the Efficiency of the Backup Process" on page 228)

If you have already configured all of the components required for performing a backup dump or restore operation, you can proceed to the instructions in "Backing Up Data" on page 251 and "Restoring and Recovering Data" on page 274.

Configuring the tapeconfig File

Several factors interact to determine how much data the Tape Coordinator can fit on a tape:

- The tape's capacity (size), as set by the tape manufacturer.
- The tape device's filemark size, as set by the tape device's manufacturer. Recall from "Tape Labels, Dump Labels, and EOF Markers" on page 197 that the Tape Coordinator writes a filemark between the data from each volume in a dump. If a dump contains a small amount of data from each of a large number of volumes, as incremental dumps often do, then the filemark size can significantly affect how much volume data fits on the tape.
- Whether or not you use the tape device's compression mode.

(The amount of data that can fit in a backup data file is determined by amount of space available on the partition, and the operating system's maximum file size. The Tape Coordinator does not write filemarks when writing to a backup data file. For further information about configuring a Tape Coordinator to write to a backup data file, see "Dumping Data to a Backup Data File" on page 236.)

As the Tape Coordinator (**butc**) process initializes, it reads the **/usr/afs/backup/tapeconfig** file on its local disk to learn the tape capacity and filemark size (for a tape device) or the file size (for a backup data file) to use for dump operations. When you begin a dump operation, the Tape Coordinator also reads the tape or backup data file's label to see if you have recorded a different tape capacity or file size. If you have, the value on the label overrides the default value from the **tapeconfig** file.

As the Tape Coordinator writes data to a tape during a dump operation, it uses the capacity and filemark information to track how much tape it has used and how much remains before the physical end-of-tape (EOT). Shortly before reaching EOT, the Tape Coordinator stops writing and requests a new tape. Similarly, it uses a backup data file's size to know when it is about to exhaust the space in the file. If the Tape Coordinator reaches the EOT unexpectedly, it recovers by obtaining a new tape and writing to it the entire contents of the volume it was writing when it reached EOT. The interrupted volume remains on the first tape, but is never used.

Many tape devices use tapes that can accommodate multiple gigabytes, or even multiple terabytes, of backup data, especially if you use the device's compression mode. When writing to such devices and tapes, allowing the Tape Coordinator to hit the EOT unexpectedly is generally recommended. The devices write data so quickly that it usually does not take much extra time to rewrite the interrupted volume on the new tape. Similarly, they compress data so well that the data abandoned on the first tape from the interrupted volume does not constitute a waste of much tape.

When writing to tapes that accommodate a smaller amount of data (say, less than two GB), it is better to avoid having the Tape Coordinator hit EOT unexpectedly. AFS supports volumes up to 2 GB in size, so an interrupted volume can in fact take up most of the tape. For such tapes, recording accurate values for tape capacity and filemark size, if possible, helps to maximize both use of tape and the efficiency of dump operations. The following discussion of the fields in the **tapeconfig** file explains how to determine the appropriate values.

Use a text editor to create an entry in a Tape Coordinator's **tapeconfig** file for each tape device or backup data file that it uses. Each device or file's entry is on its own line and has the following format:

```
[capacity   filemark_size]    device_name    port_offset
```

where

capacity

Specifies the capacity of the tapes used with a tape device, or the amount of data to write into a backup data file. Specify an integer value followed by a letter that indicates units, with no intervening space. The letter **k** or **K** indicates kilobytes, **m** or **M** indicates megabytes, **g** or **G** indicates gigabytes, and **t** or **T** indicates terabytes. If the units letter is omitted, the default is kilobytes.

To determine the capacity of a tape under two GB in size that you are going to use in regular (noncompression) mode, you can either use the value that the tape's manufacturer specifies on the tape's packaging or use the **fms** command to calculate the capacity, as described later in this section. To avoid having the Tape Coordinator reach the EOT unexpectedly, it is best to record in the **tapeconfig** file or on the label a capacity that is about 10% smaller than the actual capacity of the tape. To calculate the appropriate value for a small tape used in compression mode, one method is to multiply the tape capacity (as recorded by the manufacturer) by the device's compression ratio.

For tapes that hold multiple gigabytes or terabytes of data, or if using a tape drive's compression mode, the recommended configuration is to record a value quite a bit (for instance, two times) larger than the maximum amount you believe can fit on the tape. It is not generally worthwhile to run the **fms** command on large tapes, even in noncompression mode. The command definitely does not yield accurate results in compression mode. The Tape Coordinator is likely to reach the EOT unexpectedly, but compression mode fits so much data on the tape that the data abandoned from an interrupted volume does not represent much of the tape's capacity.

For a backup data file, record a value slightly smaller than the amount of space available on the partition, and definitely smaller than the operating system's maximum file size. It is also best to limit the ability of other processes to write to the partition, to prevent them from using up the space in the partition.

If this field is empty, the Tape Coordinator uses the maximum acceptable value (2048 GB or 2 TB). Either leave both this field and the filemark_size field empty, or provide a value in both of them.

filemark_size

Specifies the tape device's filemark size, which usually falls between 2 KB and 2 MB. Use the same notation as for the capacity field, but note that if you omit the units letter, the default unit is bytes rather than kilobytes.

For a tape device in regular (noncompression) mode, you can use the **fms** command to determine filemark size, or use the value reported by the device's manufacturer. To help the Tape Coordinator avoid reaching EOT unexpectedly, increase the value by about 10% when recording it in the **tapeconfig** file.

The recommended value for a tape device in compression mode is **0** (zero). The **fms** command does not yield accurate results in compression mode, so you cannot use it to determine the filemark size.

The recommended value for a backup data file is also **0** (zero). The Tape Coordinator does not use filemarks when writing to a file, but a value must appear in this field nevertheless if there is also a value in the capacity field.

If this field is empty, the Tape Coordinator uses the value **0** (zero). Either leave both this field and the capacity field empty, or provide a value in both of them.

device_name

Specifies the complete pathname of the tape device or backup data file. The format of tape device names depends on the operating system, but on UNIX systems, device names generally begin with the string **/dev/**. For a backup data file, this field defines the complete pathname, but for suggestions on how to name a backup data file, see "Dumping Data to a Backup Data File" on page 236.

port_offset

Specifies the port offset number for a specific tape device or backup data file. Each tape device listens for backup requests on a different UNIX port. You pick the port indirectly by recording a value in this field. The Backup System sets the device's actual port by adding the port offset to a base port number that it determines internally.

Legal values are the integers **0** through **58510** (the Backup System can track a maximum of 58,511 port offset numbers). Each value must be unique among the cell's Tape Coordinators, but you do not have to assign port offset numbers sequentially, and you can associate any number of them with a single machine or even tape device. For example, if you plan to use a device in both compression and noncompression mode, assign it two different port offsets with appropriate tape capacity and filemark values for the different modes.

Assign port offset **0** (zero) to the Tape Coordinator for the tape device or backup data file that you use most often for backup operations; doing so enables you to omit the **-portoffset** argument from the largest possible number of **backup** commands.

The following example **tapeconfig** file includes entries for two tape devices, **/dev/rmt0h** and **/dev/rmt1h**. Each one uses tapes with a capacity of 2 GB and has a filemark size of 1 MB. Their port offset numbers are `0` and `1`.

```
2g 1m /dev/rmt0h 0
2G 1M /dev/rmt1h 1
```

The **fms** command reports the capacity of the tape you have inserted and the tape device's filemark size, both on the standard output stream (stdout) and in its **fms.log** file, which it writes in the current working directory. The command interpreter must write data to the entire tape, so running the command can take from several hours to more than a day, depending on the size of the tape.

To run the fms command on a noncompressing tape device

1. If an **fms.log** file does not already exist in the current directory, verify that you can insert and write to files in the current directory. If the log file already exists, you must be able to write to the file.
2. Insert a tape into the drive. Running the command completely overwrites the tape, so use a blank tape or one that you want to recycle.

3. Issue the **fms** command.

   ```
   % fms <tape special file>
   ```

   where

   fms

   Must be typed in full.

   tape special file

   Specifies the tape device's UNIX device name, such as **/dev/rmt0h**.

The following example output reports that the tape in the device with device name **/dev/rmt0h** has a capacity of 2136604672 bytes (about 2 GB), and that the device's filemark size is 1910205 bytes (close to 2 MB).

```
% fms /dev/rmt0h
wrote block: 130408
Finished data capacity test - rewinding
wrote 1109 blocks, 1109 file marks
Finished file mark test
Tape capacity is 2136604672 bytes
File marks are 1910205 bytes
```

Granting Administrative Privilege to Backup Operators

Each person who issues the **backup** and **butc** commands in your cell must be listed in the **/usr/afs/etc/UserList** file on every database server machine that stores the Backup Database and Volume Location Database (VLDB), and every machine that houses a volume included in a volume set. By convention, the **UserList** file is the same on every server machine in the cell; the instructions in this document assume that your cell is configured in this way. To edit the **UserList** file, use the **bos adduser** and **bos removeuser** commands as described in "Administering the UserList File" on page 535.

In addition to being listed in the **UserList** file, backup operators who issue the **butc** command must be able to write to the files stored in each Tape Coordinator machine's local **/usr/afs/backup** directory, which are protected by UNIX mode bits. Before configuring your cell's first Tape Coordinator machine, decide which local user and group to designate as the owner of the directory and the files in it. Among the possible ownership options are the following:

- The local superuser **root**. With this option, the issuer of the **butc** command must log onto the local file system as the local superuser **root**. If the Tape Coordinator is also a server machine, the **-localauth** flag is used on the **butc** command to construct a server ticket from the local **/usr/afs/etc/KeyFile** file. On non-server machine, the issuer must issue the **klog** command to authenticate as an AFS administrator while logged in as **root**.

- A single AFS administrator. Logging in and authenticating are a single step if an AFS-modified login utility is used. The administrator is the only user who can start the Tape Coordinator.
- An administrative account for which several operators know the password. This allows them all to start the Tape Coordinator.

Another option is to define a group in the local group file (**/etc/group** or equivalent) to which all backup operators belong. Then turn on the **w** mode bit (**write** permission) in the group mode bits rather than the user mode bits of the **/usr/afs/backup** directory and files in it. An advantage over the methods listed previously is that each operator can retain an individual administrative account for finer granularity in auditing.

For instructions on implementing your choice of protection methods, see "Configuring Tape Coordinator Machines and Tape Devices" on page 205.

Configuring Tape Coordinator Machines and Tape Devices

This section explains how to configure a machine as a Tape Coordinator machine, and how to configure or remove the Tape Coordinator associated with a single tape device or backup data file.

> **Note:** When configuring a tape device attached to an AIX system, you must set the device's tape block size to **0** (zero) to indicate variable block size. If you do not, it is possible that devices attached to machines of other system types cannot read the tapes made on the AIX system. Use the AIX **smit** program to verify or change the value of the tape block size for a tape device, as instructed in Sep "3" on page 205.

To configure a Tape Coordinator machine

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

3. Install one or more tape devices on the Tape Coordinator machine according to the manufacturer's instructions. The Backup System can track a maximum of 58,511 tape devices or backup data files per cell.

   If the Tape Coordinator machine is an AIX system, issue the following command to change the tape device's tape block size to **0** (zero), which indicates variable block size. Repeat for each tape device.

   ```
   # chdev -l 'device_name' -a block_size='0'
   ```

   where device_name is the tape device's device name (for example, **/dev/rmt0h**).

4. Verify that the binary files for the **backup**, **butc**, and **fms** commands are available on the local disk. If the machine is an AFS client, the conventional location is the **/usr/afsws/etc** directory.

   ```
   # ls /usr/afsws/etc
   ```

5. Create the **/usr/afs** directory. (If the Tape Coordinator machine is also configured as a file server machine, this directory already exists.) Then create the **/usr/afs/backup** directory.

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

6. Use a text editor to create the **/usr/afs/backup/tapeconfig** file. Include a single line for each tape device or backup data file, specifying the following information in the indicated order. For syntax details and suggestions on the values to use in each field, see "Configuring the tapeconfig File" on page 200.

   - The capacity of tapes to be used in the device, or the size of the backup data file
   - The device's filemark size
   - The device's device name, starting with the string **/dev/**
   - The device's port offset number

7. Decide which user and group are to own the **/usr/afs/backup** directory and **/usr/afs/backup/tapeconfig** file, based on the suggestions in "Granting Administrative Privilege to Backup Operators" on page 204. Correct the UNIX mode bits on the directory and file, if necessary.

   ```
   # chown admin_owner /usr/afs/backup
   # chown admin_owner /usr/afs/backup/tapeconfig
   # chgrp admin_group /usr/afs/backup
   # chgrp admin_group /usr/afs/backup/tapeconfig
   # chmod 774 /usr/afs/backup
   # chmod 664 /usr/afs/backup/tapeconfig
   ```

8. Issue the **backup addhost** command to create a Tape Coordinator entry in the Backup Database. Repeat the command for each Tape Coordinator.

   ```
   # backup addhost <tape machine name> [<TC port offset>]
   ```

   where

   addh

   Is the shortest acceptable abbreviation of **addhost**.

   tape machine name

   Specifies the Tape Coordinator machine's fully qualified hostname.

   TC port offset

   Specifies the tape device's port offset number. Provide the same value as you specified for the device in the **tapeconfig** file. You must provide this argument unless the default value of 0 (zero) is appropriate.

To configure an additional Tape Coordinator on an existing Tape Coordinator machine

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

3. Install the tape device on the Tape Coordinator machine according to the manufacturer's instructions.

   If the Tape Coordinator machine is an AIX system, issue the following command to change the tape device's tape block size to **0** (zero), which indicates variable block size.

   ```
   # chdev -l 'device_name' -a block_size='0'
   ```

4. Choose the port offset number to assign to the tape device. If necessary, use the **backup listhosts** command to display the port offset numbers that are already used; for a discussion of the output, see "To display the list of configured Tape Coordinators" on page 208.

   ```
   # backup listhosts
   ```

   where **listh** is the shortest acceptable abbreviation of **listhosts**.

5. Use a text editor to add one or more entries for the device to the **/usr/afs/backup/tapeconfig** file. Specify the following information in the indicated order. For syntax details and suggestions on the values to use in each field, see "Configuring the tapeconfig File" on page 200.

   - The capacity of tapes to be used in the device, or the size of the backup data file
   - The device's filemark size
   - The device's device name, starting with the string **/dev/**
   - The device's port offset number

6. Issue the **backup addhost** command to create an entry in the Backup Database for the Tape Coordinator. For complete syntax, see Step "8" on page 206 in "To configure a Tape Coordinator machine" on page 205.

   ```
   # backup addhost <tape machine name> [<TC port offset>]
   ```

To unconfigure a Tape Coordinator

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Using a text editor, remove each of the Tape Coordinator's entries from the **/usr/afs/backup/tapeconfig** file.
3. Issue the **backup delhost** command to delete the Tape Coordinator's Backup Database entry.

   ```
   % backup delhost <tape machine name> [<TC port offset>]
   ```

   where

   delh

   Is the shortest acceptable abbreviation of **delhost**.

   tape machine name

   Is the complete Internet host name of the Tape Coordinator machine.

   TC port offset

   Is the same port offset number removed from the **tapeconfig** file. You must provide this argument unless the default value of **0** (zero) is appropriate.

To display the list of configured Tape Coordinators

1. Issue the **backup listhosts** command to list the Tape Coordinators and port offset numbers currently configured in the Backup Database.

   ```
   % backup listhosts
   ```

   where

   listh

   Is the shortest acceptable abbreviation of **listhosts**.

The output lists each Tape Coordinator machine and the port offset numbers currently allocated to it in the Backup Database. The appearance of a port offset number does not imply that the associated Tape Coordinator is actually running. Machine names appear in the format in which they were specified with the **backup addhost** command.

The following example output lists the Tape Coordinators currently defined in the Backup Database of the ABC Corporation cell:

```
% backup listhosts
Tape hosts:
    Host backup1.abc.com, port offset 0
    Host backup1.abc.com, port offset 2
    Host backup2.abc.com, port offset 1
    Host backup2.abc.com, port offset 3
```

Defining and Displaying Volume Sets and Volume Entries

The Backup System handles data at the level of volumes rather than individual files. You must define groups of volumes called *volume sets* before performing backup operations, by using the **backup addvolset** command. A volume set name can be up to 31 characters long and can include any character other than the period (**.**), but avoid using metacharacters that have special meanings to the shell.

After creating a volume set, use the **backup addvolentry** command to place one or more *volume entries* in it. They define the volumes that belong to it in terms of their location (file server machine and partition) and name. Use the command's required **-server** argument to designate the file server machine that houses the volumes of interest and its required **-partition** argument to designate the partition. Two types of values are acceptable:

- The fully qualified hostname of one machine or full name of one partition (such as **/vicepm**)
- The regular expression **.*** (period and asterisk), which matches every machine name or partition name in the VLDB

For the volume name (the required **-volume** argument), specify a combination of alphanumeric characters and one or more metacharacters to specify part or all of the volume name with a wildcard. You can use any of the following metacharacters in the volume name field:

.

The period matches any single character.

*****

The asterisk matches zero or more instances of the preceding character. Combine it with any other alphanumeric character or metacharacter.

[ ]

Square brackets around a list of characters match a single instance of any of the characters, but no other characters; for example, **[abc]** matches a single **a** or **b** or **c**, but not **d** or **A**. You can combine this expression with the asterisk.

^

The caret, when used as the first character in a square-bracketed set, designates a match with any single character other than the characters that follow it; for example, **[^a]** matches any single character except lowercase **a**. You can combine this expression with the asterisk.

\

A backslash preceding any of the metacharacters in this list makes it match its literal value only. For example, the expression **\.** (backslash and period) matches a single period, ***** matches a single asterisk, and \\ matches a single backslash. You can combine such expressions with the asterisk (for example, **\.*** matches any number of periods).

Perhaps the most common regular expression is the period followed by an asterisk (**.***). This expression matches any string of any length, because the period matches any character and the asterisk means any number of that character. As mentioned, it is the only acceptable regular expression in the file server and partition fields of a volume entry. In the volume name field, it can stand alone (in which case it matches every volume listed in the VLDB), or can combine with alphanumeric characters. For example, the string **user.*\.backup** matches any volume name that begins with the string **user** and ends with **.backup**.

Issuing the **backup addvolentry** command in interactive mode is simplest. If you issue it at the shell prompt, you must surround any string that includes a regular expression with double quotes (**" "**) so that the shell passes them uninterpreted to the **backup** command interpreter rather than resolving them.

To define various combinations of volumes, provide the following types of values for the **backup addvolentry** command's three arguments. The list uses the notation appropriate for interactive mode; if you issue the command at the shell prompt instead, place double quotes around any string that includes a regular expression. To create a volume entry that includes:

- All volumes listed in the VLDB, use the regular expression **.*** for all three arguments (**-server .* -partition .* -volume .***)
- Every volume on a specific file server machine, specify its fully qualified hostname as the **-server** argument and use the regular expression **.*** for the **-partition** and -**volume** arguments (for example: **-server fs1.abc.com -partition .* -volume .***)
- All volumes that reside on a partition with the same name on various file server machines, specify the complete partition name as the **-partition** argument and use the regular expression **.*** for the **-server** and **-volume** arguments (for example: **-server .* -partition /vicepd -volume .***)
- Every volume with a common string in its name, use the regular expression **.*** for the **-server** and **-partition** arguments, and provide a combination of alphanumeric characters and metacharacters as the **-volume** argument (for example: **-server .* -partition .* -volume .*\.backup** includes all volumes whose names end in the string **.backup**).
- All volumes on one partition, specify the machine's fully qualified hostname as the **-server** argument and the full partition name as the **-partition** argument, and use the regular expression **.*** for the **-volume** argument (for example: **-server fs2.abc.com -partition /vicepb -volume .***).
- A single volume, specify its complete name as the **-volume** argument. To bypass the potentially time-consuming search through the VLDB for matching entries, you can specify an actual machine and partition name for the **-server** and **-partition** arguments respectively. However, if it is possible

that you need to move the volume in future, it is best to use the regular expression **.*** for the machine and partition name.

As you create volume sets, define groups of volumes you want to dump to the same tape at the same time (for example, weekly or daily) and in the same manner (fully or incrementally). In general, a volume set that includes volumes with similar contents (as indicated by similar names) is more useful than one that includes volumes that share a common location, especially if you often move volumes for load-balancing or space reasons. Most often, then, it is appropriate to use the regular expression **.*** (period followed by a backslash) for the **-server** and **-partition** arguments to the **backup addvolentry** command.

It is generally more efficient to include a limited number of volumes in a volume entry. Dumps of a volume set that includes a large number of volume can take a long time to complete, increasing the possibility that the operation fails due to a service interruption or outage.

To remove a volume entry from a volume set, use the **backup delvolentry** command. To remove a volume set and all of its component volume entries from the Backup Database, use the **backup delvolset** command. To display the volume entries in a volume set, use the **backup listvolsets** command.

By default, a Backup Database record is created for the new volume set. Sometimes it is convenient to create volume sets without recording them permanently in the Backup Database, for example when using the **backup volsetrestore** command to restore a group of volumes that were not necessarily backed up together (for further discussion, see "Using the backup volsetrestore Command" on page 281). To create a *temporary* volume set, include the **-temporary** flag to the **backup addvolset** command. A temporary volume set exists only during the lifetime of the current interactive session, so the flag is effective only when used during an interactive session (opened by issuing the **backup (interactive)** command). You can use the **backup delvolset** command to delete a temporary volume set before the interactive session ends, if you wish, but as noted it is automatically deleted when you end the session. One advantage of temporary volume sets is that the **backup addvolset** command, and any **backup addvolentry** commands subsequently used to add volume entries to it, complete more quickly than for regular volume sets, because you are not creating any Backup Database records.

To create a volume set

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. **(Optional)** Issue the **backup** command to enter interactive mode. If you are going to define volume entries right away with the **backup addvolentry** command, this eliminates the need to surround metacharacter expressions with double quotes. You must enter interactive mode if creating a temporary volume set.

   ```
   % backup
   ```

3. Issue the **(backup) addvolset** command to create the volume set. You must then issue the **(backup) addvolentry** command to define volume entries in it.

   ```
   backup>  addvolset <volume set name> [-temporary]
   ```

where

addvols

Is the shortest acceptable abbreviation of **addvolset**.

volume set name

Names the volume set. The name can include no more than 31 characters, cannot include periods, and must be unique within the Backup Database. (A temporary volume set can have the same name as an existing permanent volume set, but this is not recommended because of the confusion it can cause.)

-temporary

Creates a temporary volume set, which exists only during the current interactive session.

To add a volume entry to a volume set

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. **(Optional)** Issue the **backup** command to enter interactive mode if you have not already. This makes it simpler to use metacharacter expressions, because you do not need to surround them with double quotes. If you are adding entries to a temporary volume set, you must already have entered interactive mode before creating the volume set.

   ```
   % backup
   ```

3. Issue the **(backup) addvolentry** command to define volume entries in an existing volume set. The Backup System assigns each volume entry an index within the volume set, starting with 1 (one).

   ```
   backup> addvolentry  -name <volume set name>  \
                        -server <machine name>  \
                        -partition <partition name>  \
                        -volumes <volume name (regular expression)>
   ```

   where

   addvole

   Is the shortest acceptable abbreviation of **addvolentry**.

   -name

   Names the volume set to which to add the volume entry. It must already exist (use the **backup addvolset** command to create it).

-server

Defines the set of one or more file server machines that house the volumes in the volume entry. Provide either one fully-qualified hostname (such as **fs1.abc.com**) or the metacharacter expression **.*** (period and asterisk), which matches all machine names in the VLDB.

-partition

Defines the set of one or more partitions that house the volumes in the volume entry. Provide either one complete partition name (such as **/vicepa**) or the metacharacter expression **.*** (period and asterisk), which matches all partition names.

-volumes

Defines the set of one or more volumes included in the volume entry, identifying them by name. This argument can include a combination of alphanumeric characters and one or more of the metacharacter expressions discussed in the introductory material in this section.

To display volume sets and volume entries

1. Issue the **backup listvolsets** command to display the volume entries in a specific volume set or all of them. If you are displaying a temporary volume set, you must still be in the interactive session in which you created it.

   ```
   % backup listvolsets [<volume set name>]
   ```

   where

   listv

   Is the shortest acceptable abbreviation of **listvolsets**.

   volume set name

   Names the volume set to display. Omit this argument to display all defined volume sets.

   The output from the command uses the wildcard notation used when the volume entries were created. The string `(temporary)` marks a temporary volume set. The following example displays all three of the volume sets defined in a cell's Backup Database, plus a temporary volume set **pat+jones** created during the current interactive session:

   ```
   backup> listv
   Volume set pat+jones (temporary):
     Entry 1: server fs1.abc.com, partition /vicepe, volumes: user.pat.backup
     Entry 2: server fs5.abc.com, partition /viceph, volumes: user.jones.backup
   Volume set user:
     Entry 1: server .*, partition .*, volumes: user.*\.backup
   Volume set sun:
     Entry 1: server .*, partition .*, volumes: sun4x_55\..*
   ```

```
  Entry 2: server .*, partition .*, volumes: sun4x_56\..*
Volume set rs:
  Entry 1: server .*, partition .*, volumes: rs_aix42\..*
```

To delete a volume set

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **backup delvolset** command to delete one or more volume sets and all of the component volume entries in them. If you are deleting a temporary volume set, you must still be in the interactive session in which you created it.

   ```
   % backup delvolset <volume set name>+
   ```

   where

   delvols

   Is the shortest acceptable abbreviation of **delvolset**.

   volume set name

   Names each volume set to delete.

To delete a volume entry from a volume set

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

3. If the volume set includes more than one volume entry, issue the **(backup) listvolsets** command to display the index number associated with each one (if there is only one volume entry, its index is 1). For a more detailed description of the command's output, see "To display volume sets and volume entries" on page 213.

   ```
   backup> listvolsets <volume set name>
   ```

where

listv

Is the shortest acceptable abbreviation of **listvolsets**.

volume set name

Names the volume set for which to display volume entries.

4. Issue the **(backup) delvolentry** command to delete the volume entry.

```
backup> delvolentry <volume set name>  <volume entry index>
```

where

delvole

Is the shortest acceptable abbreviation of **delvolentry**.

volume set name

Names the volume set from which to delete a volume entry.

volume entry index

Specifies the index number of the volume entry to delete.

Defining and Displaying the Dump Hierarchy

A dump hierarchy is a logical structure in the Backup Database that defines the relationship between full and incremental dumps; that is, it defines which dump serves as the parent for an incremental dump. Each individual component of a hierarchy is a dump level.

As you define dump levels with the **backup adddump** command, keep the following rules and suggestions in mind:

- Each full dump level is the top level of a hierarchy. You can create as many hierarchies as you need to dump different volume sets on different schedules.
- The name of a full dump level consists of an initial slash (**/**), followed by a string of up to 28 alphanumeric characters.
- The name of an incremental dump level resembles a pathname, starting with the name of a full dump level, then the first incremental level, and so on, down to the final incremental level. Precede each level name with a slash to separate it from the preceding level. Like the full level, each component level in the name can have up to 28 alphanumeric characters, not including the slash.

- A hierarchy can have any have any number of levels, but the maximum length of a complete dump level name is 256 characters, including the slashes.
- Before defining a given incremental level, you must define all of the levels above it in the hierarchy.
- Do not use the period (**.**) in dump level names. The Backup System uses the period as the separator between a dump's volume set name and dump level name when it creates the dump name and AFS tape name. Any other alphanumeric and punctuation characters are allowed, but it is best to avoid metacharacters. If you include a metacharacter, you must precede it with a backslash (****) or surround the entire dump level name with double quotes (**" "**).
- Naming dump levels for days or other actual time points reminds you when to perform dumps, and makes it easier to track the relationship between dumps performed at different levels. However, the names have no meaning to the Backup System: it does not automatically create dumps according to the names, and does not prevent you from, for example, using the **/sunday** level when creating a dump on a Tuesday.
- It is best not to use the same name for more than one component level in a hierarchy, because it means the resulting dump name no longer indicates which level was used. For example, if you name a dump level **/full/incr/incr**, then the dump name and AFS tape name that result from dumping a volume set at the first incremental level (**/full/incr**) look the same as the names that result from dumping at the second incremental level (**/full/incr/incr**).
- Individual levels in different hierarchies can have the same name, but the complete pathnames must be unique. For example, **/sunday1/monday** and **/sunday2/monday** share the same name at the final level, but are unique because they have different names at the full level (belong to different hierarchies). However, using the same name in multiple hierarchies means that dump and AFS tape names do not unambiguously indicate which hierarchy was used.

The following example shows three hierarchies. Each begins with a full dump at the top: **sunday1** for the first hierarchy, **sunday2** for the second hierarchy, and **sunday_bin** for the third hierarchy. In all three hierarchies, each of the other dump levels is an incremental level.

```
/sunday1
        /monday
        /tuesday
        /wednesday
        /thursday
        /friday
/sunday2
        /monday
               /tuesday
                       /wednesday
                                 /thursday
                                          /friday
 /sunday_bin
            /monday
                   /wednesday
                             /friday
```

In the first hierarchy, each incremental dump level refers to the full level **/sunday1** as its parent. When (for example) you dump a volume set at the **/sunday1/wednesday** level, it includes data that has changed since the volume set was dumped at the **/sunday1** level.

In contrast, each incremental dump level in the second hierarchy refers to the immediately preceding dump level as its parent. When you dump a volume set at the corresponding level in the second hierarchy (**/sunday2/monday/tuesday/wednesday**), the dump includes only data that has changed since the volume set was dumped at the **/sunday2/monday/tuesday** level (presumably the day before). Assuming you create dumps on the indicated days, an incremental dump made using this hierarchy contains less data than an incremental dump made at the corresponding level in the first hierarchy.

The third hierarchy is more appropriate for dumping volumes for which a daily backup is excessive because the data does not change often (for example, system binaries).

Creating a Tape Recycling Schedule

If your cell is like most cells, you have a limited amount of room for storing backup tapes and a limited budget for new tapes. The easiest solution is to recycle tapes by overwriting them when you no longer need the backup data on them. The Backup System helps you implement a recycling schedule by enabling you to associate an expiration date with each dump level. The expiration date defines when a dump created at that level expires. Until that time the Backup System refuses to overwrite a tape that contains the dump. Thus, assigning expiration dates automatically determines how you recycle tapes.

When designing a tape-recycling schedule, you must decide how far in the past and to what level of precision you want to guarantee access to backed up data. For instance, if you decide to guarantee that you can restore a user’s home volume to its state on any given day in the last two weeks, you cannot recycle the tape that contains a given daily dump for at least two weeks after you create it. Similarly, if you decide to guarantee that you can restore home volumes to their state at the beginning of any given week in the last month, you cannot recycle the tapes in a dump set containing a weekly dump for at least four weeks. The following example dump hierarchy implements this recycling schedule by setting the expiration date for each daily incremental dump to 13 days and the expiration date of the weekly full dumps to 27 days.

The tapes used to store dumps created at the daily incremental levels in the **/sunday1** hierarchy expire just in time to be recycled for daily dumps in the **/sunday3** hierarchy (and vice versa), and there is a similar relationship between the **/sunday2** and **/sunday4** hierarchies. Similarly, the tape that houses a full dump at the **/sunday1** level expires just in time to be used for a full dump on the first Sunday of the following month.

```
/sunday1   expires in 27d
        /monday1  expires in 13d
        /tuesday1  expires in 13d
        /wednesday1  expires in 13d
        /thursday1  expires in 13d
        /friday1  expires in 13d
/sunday2   expires in 27d
        /monday2  expires in 13d
        /tuesday2  expires in 13d
        /wednesday2  expires in 13d
        /thursday2  expires in 13d
        /friday2  expires in 13d
/sunday3   expires in 27d
```

```
            /monday1  expires in 13d
            /tuesday1  expires in 13d
            /wednesday1  expires in 13d
            /thursday1  expires in 13d
            /friday1  expires in 13d
    /sunday4   expires in 27d
            /monday2  expires in 13d
            /tuesday2  expires in 13d
            /wednesday2  expires in 13d
            /thursday2  expires in 13d
            /friday2  expires in 13d
```

If you use appended dumps in your cell, keep in mind that all dumps in a dump set are subject to the latest (furthest into the future) expiration date associated with any of the constituent dumps. You cannot recycle any of the tapes that contain a dump set until all of the dumps have reached their expiration date. See also "Appending Dumps to an Existing Dump Set" on page 260.

Most tape manufacturers recommend that you write to a tape a limited number of times, and it is best not to exceed this limit when recycling tapes. To help you track tape usage, the Backup System records a `useCount` counter on the tape's label. It increments the counter each time the tape's label is rewritten (each time you use the **backup labeltape** or **backup dump** command). To display the `useCount` counter, use the **backup readlabel** or **backup scantape** command or include the **-id** and **-verbose** options when you issue the **backup dumpinfo** command. For instructions see "Writing and Reading Tape Labels" on page 223 or "Displaying Backup Dump Records" on page 266.

Archiving Tapes

Even if you make extensive use of tape recycling, there is probably some backup data that you need to archive for a long (or even an indefinite) period of time. You can use the Backup System to archive data on a regular schedule, and you can also choose to archive data on tapes that you previously expected to recycle.

If you want to archive data on a regular basis, you can create date-specific dump levels in the dump hierarchy. For example, if you decide to archive a full dump of all data in your cell at the beginning of each quarter in the year 2000, you can define the following levels in the dump hierarchy:

```
/1Q2000
/2Q2000
/3Q2000
/4Q2000
```

If you decide to archive data that is on tapes you previously planned to recycle, you must gather all of the tapes that contain the relevant dumps, both full and incremental. To avoid accidental erasure, it is best to set the switch on the tapes that makes them read-only, before placing them in your archive storage area. If the tapes also contain a large amount of extraneous data that you do not want to archive, you can restore just the relevant data into a new temporary volume, and back up that volume to the smallest number of tapes possible. One reason to keep a dump set small is to minimize the amount of irrelevant data in a dump set you end up needing to archive.

If you do not expect to restore archived data to the file system, you can consider using the **backup deletedump** command to remove the associated dump records from the Backup Database, which helps keep it to an efficient size. If you ever need to restore the data, you can use the **-dbadd** flag to the **backup scantape** command to reinsert the dump records into the database. For instructions, see "To scan the contents of a tape" on page 272.

Defining Expiration Dates

To associate an expiration date with a dump level as you create it, use the **-expires** argument to the **backup adddump** command. To change an existing dump level's expiration date, use the **-expires** argument to the **backup setexp** command. (Note that it is not possible to change the expiration date of an actual dump that has already been created at that level). With both commands, you can define an expiration date either in absolute terms (for example, 13 January 2000) or relative terms (for example, 30 days from when the dump is created).

- To define an absolute expiration date, provide a value for the **-expires** argument with the following format:

  ```
  [at] mm/dd/yyyy [hh:MM]
  ```

  where mm indicates the month, dd the day, and yyyy the year when the dump expires. Valid values for the year fall in the range **1970** through **2037** (the latest possible date that the UNIX time representation can express is in early 2038). If you provide a time, it must be in 24-hour format with hh the hours and MM the minutes (for example, **21:50** is 9:50 p.m.). If you omit the time, the default is 00:00 hours (12:00 midnight) on the indicated date.

- To define a relative expiration date, provide a value for the **-expires** argument with the following format:

  ```
  [in] [yearsy] [monthsm] [daysd]
  ```

  where each of years, months, and days is an integer. Provide at least one of them together with the corresponding units letter (**y**, **m**, or **d** respectively), with no intervening space. If you provide more than one of the three, list them in the indicated order.

  The Backup System calculates a dump's actual expiration date by adding the indicated relative value to the start time of the dump operation. For example, it assigns an expiration date 1 year, 6 months, and 2 days in the future to a dump created at a dump level with associated expiration date **in 1y 6m 2d**.

- To indicate that a dump backed up at the corresponding dump level never expires, provide the value **NEVER** instead of a date and time. To recycle tapes that contain dumps created at such a level, you must use the **backup readlabel** command to overwrite the tape's label.

If you omit the **-expires** argument to the **backup adddump** command, then the expiration date is set to UNIX time zero (00:00 hours on 1 January 1970). The Backup System considers dumps created at such a dump level to expire at their creation time. If no dumps in a dump set have an expiration date, then the Backup System does not impose any restriction on recycling the tapes that contain the dump set. If you need to prevent premature recycling of the tapes that contain the dump set, you must use a manual tracking system.

To add a dump level to the dump hierarchy

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. **Optional**. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

3. Issue the **backup adddump** command to define one or more dump levels. If you are defining an incremental level, then all of the parent levels that precede it in its pathname must either already exist or precede it on the command line.

   ```
   backup> adddump -dump <dump level name>+ [-expires <expiration date>+]
   ```

   where

   addd

   Is the shortest acceptable abbreviation of **adddump**.

   -dump

   Names each dump level to added. If you specify more than one dump level name, you must include the **-dump** switch.

   Provide the entire pathname of the dump level, preceding each level in the pathname with a slash (**/**). Each component level can be up to 28 characters in length, and the pathname can include up to 256 characters including the slashes.

   -expires

   Sets the expiration date associated with each dump level. Specify either a relative or absolute expiration date, as described in "Defining Expiration Dates" on page 219, or omit this argument to assign no expiration date to the dump levels.

   Note: A plus sign follows this argument in the command's syntax statement because it accepts a multiword value which does not need to be enclosed in double quotes or other delimiters, not because it accepts multiple dates. Provide only one date (and optionally, time) definition to be associated with each dump level specified by the **-dump** argument.

To change a dump level's expiration date

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. **Optional**. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

3. Issue the **(backup) setexp** command to change the expiration date associated with one or more dump levels.

   ```
   backup> setexp -dump <dump level name>+ [-expires <expiration date>+]
   ```

   where

   se

   Is the shortest acceptable abbreviation of **setexp**.

   -dump

   Names each existing dump level for which to change the expiration date.

   -expires

   Sets the expiration date associated with each dump level. Specify either a relative or absolute expiration date, as described in "Defining Expiration Dates" on page 219; omit this argument to remove the expiration date currently associated with each dump level.

   Note: A plus sign follows this argument in the command's syntax statement because it accepts a multiword value which does not need to be enclosed in double quotes or other delimiters, not because it accepts multiple dates. Provide only one date (and optionally, time) definition to be associated with each dump level specified by the **-dump** argument.

To delete a dump level from the dump hierarchy

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. **Optional**. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

3. Issue the **(backup) deldump** command to delete the dump level. Note that the command automatically removes all incremental dump levels for which the specified level serves as parent, either directly or indirectly.

   ```
   backup> deldump <dump level name>
   ```

   where

   deld

   Is the shortest acceptable abbreviation of **deldump**.

   dump level name

   Specifies the complete pathname of the dump level to delete.

To display the dump hierarchy

1. Issue the **backup listdumps** command to display the dump hierarchy.

   ```
   % backup listdumps
   ```

   where **listd** is the shortest acceptable abbreviation of **listdumps**.

   The output from this command displays the dump hierarchy, reporting the expiration date associated with each dump level, as in the following example.

   ```
    % backup listdumps
   /week1  expires in  27d
         /tuesday  expires in  13d
                 /thursday  expires in  13d
         /sunday  expires in  13d
                /tuesday expires in  13d
                        /thursday expires in  13d
   /week3  expires in  27d
         /tuesday  expires in  13d
                 /thursday  expires in  13d
         /sunday  expires in  13d
                /tuesday  expires in  13d
                        /thursday  expires in  13d
   sunday1   expires in  27d
          /monday1  expires in  13d
          /tuesday1  expires in  13d
          /wednesday1  expires in  13d
          /thursday1  expires in  13d
          /friday1  expires in  13d
   sunday2   expires in  27d
   ```

```
              /monday2  expires in  13d
              /tuesday2  expires in  13d
              /wednesday2  expires in  13d
              /thursday2  expires in  13d
              /friday2  expires in  13d
       sunday3   expires in  27d
              /monday1  expires in  13d
              /tuesday1  expires in  13d
              /wednesday1  expires in  13d
              /thursday1  expires in  13d
              /friday1  expires in  13d
       sunday4   expires in  27d
              /monday2  expires in  13d
              /tuesday2  expires in  13d
              /wednesday2  expires in  13d
              /thursday2  expires in  13d
              /friday2  expires in  13d
```

Writing and Reading Tape Labels

As described in "Dump Names and Tape Names" on page 197 and "Tape Labels, Dump Labels, and EOF Markers" on page 197, you can assign either a permanent name or an AFS tape name to a tape that you use in the Backup System. The names are recorded on the tape's magnetic label, along with an indication of the tape's capacity (size).

You can assign either a permanent name or an AFS tape name, but not both. In general, assigning permanent names rather than AFS tape names simplifies the backup process, because the Backup System does not dictate the format of permanent names. If a tape does not have a permanent name, then by default the Backup System accepts only three strictly defined values in the AFS tape name field, and refuses to write a dump to a tape with an inappropriate AFS tape name. The acceptable values are a name that matches the volume set and dump level of the initial dump, the value `<NULL>`, and no value in the field at all.

If a tape has a permanent name, the Backup System does not check the AFS tape name, and as part of the dump operation constructs the appropriate AFS tape name itself and records it on the label. This means that if you assign a permanent name, the Backup System assigns an AFS tape name itself and the tape has both types of name. In contrast, if a tape has an AFS tape name but not a permanent name, you cannot assign a permanent name without first erasing the AFS tape name.

(You can also suppress the Backup System's check of a tape's AFS tape name, even it does not have a permanent name, by assigning the value **NO** to the **NAME_CHECK** instruction in the *device configuration file*. See "Eliminating the AFS Tape Name Check" on page 235.)

Because the Backup System accepts unlabeled tapes, you do not have to label a tape before using it for the first time. After the first use, there are a couple of cases in which you must relabel a tape in order to write a dump to it:

- The tape does not have a permanent name, and the AFS tape name on it does not match the new initial dump set you want to create (the volume set and dump level names are different, or the index is incorrect).
- You want to recycle a tape before all of the dumps on it have expired. The Backup System does not overwrite a tape with any unexpired dumps. Keep in mind, though, that if you relabel the tape to making recycling possible, you erase all the dump records for the tape from the Backup Database, which makes it impossible to restore any data from the tape.

Note: Labeling a tape that contains dump data makes it impossible to use that data in a restore operation, because the labeling operation removes the dump's records from the Backup Database. If you want to record a permanent name on a tape label, you must do it before dumping any data to the tape.

Recording a Name on the Label

To write a permanent name on a tape's label, include the **-pname** argument to specify a string of up to 32 characters. Check that no other tape used with the Backup System in your cell already has the permanent name you are assigning, because the Backup System does not prevent you from assigning the same name to multiple tapes. The Backup System overwrites the existing AFS tape name, if any, with the value `<NULL>`. When a tape has a permanent name, the Backup System uses it instead of the AFS tape name in most prompts and when referring to the tape in output from **backup** commands. The permanent name persists until you again include the **-pname** argument to the **backup labeltape** command, regardless of the tape's contents and of how often you recycle the tape or use the **backup labeltape** command without the **-pname** argument.

To write an AFS tape name on the label, provide a value for the **-name** argument that matches the volume set name and the final element in the dump level pathname of the initial dump that you plan to write to the tape, and an index that indicates the tape's place in the sequence of tapes for the dump set. The format is as follows:

```
volume_set_name.dump_level_name.tape_index
```

If you omit the **-name** argument, the Backup System sets the AFS tape name to `<NULL>`. The Backup System automatically constructs and records the appropriate name when you later write an initial dump to the tape by using the **backup dump** or **backup savedb** command.

You cannot use the **-name** argument if the tape already has a permanent name. To erase a tape's permanent name, provide a null value to the **-pname** argument by issuing the following command:

```
% backup labeltape -pname ""
```

Recording a Capacity on the Label

To record the tape's capacity on the label, specify a number of kilobytes as the **-size** argument. If you omit this argument the first time you label a tape, the Backup System records the default tape capacity associated with the specified port offset in the **/usr/afs/backup/tapeconfig** file on the Tape Coordinator machine. If the tape's capacity is different (in particular, larger) than the capacity recorded in the **tapeconfig** file, it is best to record a capacity on the label before using the tape. Once set, the value in the label's capacity field persists until you again use the **-size** argument to the **backup labeltape** command. For a discussion of the appropriate capacity to record for tapes, see "Configuring the tapeconfig File" on page 200.

To read a tape's label, use the **backup readlabel** command.

Most tapes also come with an adhesive label you can apply to the exterior casing. To help you easily identify a tape, record at least the tape's permanent and AFS tape names on the adhesive label. Depending on the recycling scheme you use, it can be useful to record other information, such as the dump ID, dump creation date, and expiration date of each dump you write to the tape.

To label a tape

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

   ```
   % butc [<port offset>] [-noautoquery]
   ```

3. Place the tape in the device.
4. **Optional**. Issue the **backup** command to enter interactive mode, if you want to label multiple tapes or issue additional commands after labeling the tape. The interactive prompt appears in the following step.

   ```
   % backup
   ```

5. Issue the **(backup) labeltape** command to label the tape.

   ```
   backup> labeltape [-name <tape name, defaults to NULL>]  \
      [-size <tape size in Kbytes, defaults to size in tapeconfig>]  \
      [-portoffset <TC port offset>] [-pname <permanent tape name>]
   ```

   where

   la

   Is the shortest acceptable abbreviation of **labeltape**.

   -name

   Specifies the AFS tape name to record on the label. Include this argument or the **-pname** argument, but not both. If you omit this argument, the AFS tape name is set to <*NULL*>. If you provide it, it must have the following format.

   ```
   volume_set_name.dump_level_name.tape_index
   ```

for the tape to be acceptable for use in a future **backup dump** operation. The volume_set_name must match the volume set name of the initial dump to be written to the tape, dump_level_name must match the last element of the dump level pathname at which the volume set is to be dumped, and tape_index must correctly indicate the tape's place in the sequence of tapes that house the dump set; indexing begins with the number 1 (one).

-size

Specifies the tape capacity to record on the label. If you are labeling the tape for the first time, you need to include this argument only if the tape's capacity differs from the capacity associated with the specified port offset in the **/usr/afs/backup/tapeconfig** file on the Tape Coordinator machine.

If you provide a value, it is an integer value followed by a letter that indicates units, with no intervening space. A unit value of **k** or **K** indicates kilobytes, **m** or **M** indicates megabytes, and **g** or **G** indicates gigabytes. If you omit the units letter, the default is kilobytes.

-portoffset

Specifies the port offset number of the Tape Coordinator handling the tape or backup data file for this operation. You must provide this argument unless the default value of **0** (zero) is appropriate.

-pname

Specifies the permanent name to record on the label. It can be up to 32 characters in length, and include any alphanumeric characters. Avoid metacharacters that have a special meaning to the shell, to avoid having to mark them as literal in commands issued at the shell prompt.

Include this argument or the **-name** argument, but not both. When you provide this argument, the AFS tape name is set to `<NULL>`. If you omit this argument, any existing permanent name is retained.

6. If you did not include the **-noautoquery** flag when you issued the **butc** command, or if the device's device configuration file includes the instruction **AUTOQUERY YES**, then the Tape Coordinator prompts you to place the tape in the device's drive. You have already done so, but you must now press **<Return>** to indicate that the tape is ready for labeling.

To read the label on a tape

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on

page 536.

```
% bos listusers <machine name>
```

2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

```
% butc [<port offset>] [-noautoquery]
```

3. Place the tape in the device.
4. **Optional**. Issue the **backup** command to enter interactive mode, if you want to label multiple tapes or issue additional commands after labeling the tape. The interactive prompt appears in the following step.

```
% backup
```

5. Issue the **(backup) readlabel** command to read the label on the tape.

```
backup> readlabel [<TC port offset>]
```

where

rea

Is the shortest acceptable abbreviation of **readlabel**.

TC port offset

Specifies the port offset number of Tape Coordinator handling the tape or backup data file for this operation. You must provide this argument unless the default value of **0** (zero) is appropriate.

6. If you did not include the **-noautoquery** flag when you issued the **butc** command, or the device's device configuration file includes the instruction **AUTOQUERY YES** instruction, then the Tape Coordinator prompts you to place the tape in the device's drive. You have already done so, but you must now press **<Return>** to indicate that the tape is ready for reading.

Information from the tape label appears both in the **backup** command window and in the Tape Coordinator window. The output in the command window has the following format:

```
Tape read was labelled: tape_name (initial_dump_ID)
     size: size KBytes
```

where tape_name is the tape's permanent name (if it has one) or AFS tape name, initial_dump_ID is the dump ID of the initial dump on the tape, and size is the capacity recorded on the label, in kilobytes.

The information in the Tape Coordinator window is more extensive. The tape's permanent name appears in the `tape name` field and its AFS tape name in the `AFS tape name` field. If either name is undefined, a value of `<NULL>` appears in the field instead. The capacity recorded on the label appears in the `size` field. Other fields in the output report the creation time, dump level name, and dump ID of the initial dump on the tape (`creationTime`, `dump path`, and `dump id` respectively). The `cell` field reports the cell in which the dump operation was performed, and the `useCount` field reports the number of times the tape has been relabeled, either with the **backup labeltape** command or during a dump operation. For further details, see the command's reference page in the *IBM AFS Administration Reference*.

If the tape has no label, or if the drive is empty, the following message appears at the command shell:

```
Failed to read tape label.
```

The following example illustrates the output in the command shell for a tape in the device with port offset 1:

```
% backup readlabel 1
Tape read was labelled: monthly_guest (917860000)
     size: 2150000 KBytes
```

The following output appears in the Tape Coordinator window at the same time:

```
Tape label
----------
tape name = monthly_guest
AFS tape name = guests.monthly.3
creationTime =  Mon Feb  1 04:06:40 1999
cell = abc.com
size = 2150000 Kbytes
dump path = /monthly
dump id = 917860000
useCount = 44
-- End of tape label --
```

Automating and Increasing the Efficiency of the Backup Process

The Backup System includes several optional features to help you automate the backup process in your cell and make it more efficient. By combining several of the features, you can dump volume data to tape with minimal human intervention in most cases. To take advantage of many of the features, you create a device configuration file in the **/usr/afs/backup** directory for each tape device that participates in automated operations. For general instructions on creating the device configuration file, see "Creating a Device Configuration File" on page 229. The following list refers you to sections that describe each feature in greater detail.

- You can use tape stackers and jukeboxes to perform backup operations. These are tape drives with an attached unit that stores several tapes and can physically insert and remove them from the tape reader (tape drive) without human intervention, meaning that no operator has to be present even for backup operations that require several tapes. To use a stacker or jukebox with the Backup System, include the **MOUNT** and **UNMOUNT** instructions in its device configuration file. See "Invoking a Device's Tape Mounting and Unmounting Routines" on page 230.
- You can suppress the Tape Coordinator's default prompt for the initial tape that it needs for a backup operation, again eliminating the need for a human operator to be present when a backup operation begins. (You must still insert the correct tape in the drive at some point before the operation begins.)

To suppress the initial prompt, include the **-noautoquery** flag on the **butc** command, or assign the value **NO** to the **AUTOQUERY** instruction in the device configuration file. See "Eliminating the Search or Prompt for the Initial Tape" on page 234.

- You can suppress the prompts that the Tape Coordinator otherwise generates when it encounters several types of errors. When you use this feature, the Tape Coordinator instead responds to the errors in a default manner, which generally allows the operation to continue without human intervention. To suppress prompts about error conditions, assign the value **NO** to the **ASK** instruction in the device configuration file. See "Enabling Default Responses to Error Conditions" on page 234.
- You can suppress the Backup System's default verification that the AFS tape name on a tape that has no permanent name matches the name derived from the volume set and dump level names of the initial dump the Backup System is writing to the tape. This enables you to recycle a tape without first relabeling it, as long as all dumps on it are expired. To suppress name checking, assign the value **NO** to the **NAME_CHECK** instruction in the device configuration file. See "Eliminating the AFS Tape Name Check" on page 235.
- You can promote tape streaming (the most efficient way for a tape device to operate) by setting the size of the memory buffer the Tape Coordinator uses when transferring volume data between the file system and the device. To set the buffer size, include the **BUFFERSIZE** instruction in the device configuration file. See "Setting the Memory Buffer Size to Promote Tape Streaming" on page 236.
- You can write dumps to a *backup data file* on the local disk of the Tape Coordinator machine, rather than to tape. You can then transfer the backup data file to a data-archiving system, such as a hierarchical storage management (HSM) system, that you use in conjunction with AFS and the Backup System. Writing a dump to a file is usually more efficient that issuing the equivalent **vos dump** commands individually. To write dumps to a file, include the **FILE** instruction in the device configuration file. See "Dumping Data to a Backup Data File" on page 236.

There are two additional ways to increase backup automation and efficiency that do not involve the device configuration file:

- You can schedule one or more **backup dump** commands to run at specified times. This enables you to create backups at times of low system usage, without requiring a human operator to be present. You can schedule a single dump operation for a future time, or multiple operations to run at various future times. See "Scheduling Dumps" on page 262.
- You can append dumps to a tape that already has other dumps on it. This enables you to use as much of a tape's capacity as possible. The appended dumps do not have be related in any way to one another or to the initial dump on the tape, but grouping dumps appropriately can reduce the number of necessary tape changes during a restore operation. To append a dump, include the **-append** argument to the **backup dump** command. See "Appending Dumps to an Existing Dump Set" on page 260.

Creating a Device Configuration File

To use many of the features that automate backup operations, create a configuration file for each tape device in the **/usr/afs/backup** directory on the local disk of the Tape Coordinator machine that drives the device. The filename has the following form:

CFG_device_name

where device_name represents the name of the tape device or backup data file (see "Dumping Data to a Backup Data File" on page 236 to learn about writing dumps to a file rather than to tape).

For a tape device, construct the device_name portion of the name by stripping off the initial **/dev/** string with which all UNIX device names conventionally begin, and replacing any other slashes in the name with underscores. For example, **CFG_rmt_4m** is the appropriate filename for a device called **/dev/rmt/4m**.

For a backup data file, construct the device_name portion by stripping off the initial slash (**/**) and replacing any other slashes (**/**) in the name with underscores. For example, **CFG_var_tmp_FILE** is the appropriate filename for a backup data file called **/var/tmp/FILE**.

Creating a device configuration file is optional. If you do not want to take advantage of any of the features that the file provides, you do not have to create it.

You can include one of each of the following instructions in any order in a device configuration file. All are optional. Place each instruction on its own line, but do not include any newline (**<Return>**) characters within an instruction.

MOUNT and UNMOUNT

Identify a script of routines for mounting and unmounting tapes in a tape stacker or jukebox's drive as needed. See "Invoking a Device's Tape Mounting and Unmounting Routines" on page 230.

AUTOQUERY

Controls whether the Tape Coordinator prompts for the first tape it needs for a backup operation. See "Eliminating the Search or Prompt for the Initial Tape" on page 234.

ASK

Controls whether the Tape Coordinator asks you how to respond to certain error conditions. See "Enabling Default Responses to Error Conditions" on page 234.

NAME_CHECK

Controls whether the Tape Coordinator verifies that an AFS tape name matches the initial dump you are writing to the tape. See "Eliminating the AFS Tape Name Check" on page 235.

BUFFERSIZE

Sets the size of the memory buffer the Tape Coordinator uses when transferring data between a tape device and a volume. See "Setting the Memory Buffer Size to Promote Tape Streaming" on page 236.

FILE

Controls whether the Tape Coordinator writes dumps to, and restores data from, a tape device or a backup data file. See "Dumping Data to a Backup Data File" on page 236.

Invoking a Device's Tape Mounting and Unmounting Routines

A tape stacker or jukebox helps you automate backup operations because it can switch between multiple tapes during an operation without human intervention. To take advantage of this feature, include the **MOUNT** and optionally **UNMOUNT** instructions in the device configuration file that you write for the stacker or jukebox. The instructions share the same syntax:

```
MOUNT filename
   UNMOUNT filename
```

where filename is the pathname on the local disk of a script or program you have written that invokes the routines defined by the device's manufacturer for mounting or unmounting a tape in the device's tape drive. (For convenience, the following discussion uses the term *script* to refers to both scripts and programs.) The script usually also contains additional logic that handles error conditions or modifies the script's behavior depending on which backup operation is being performed.

You can refer to different scripts with the **MOUNT** or **UNMOUNT** instructions, or to a single script that invokes both mounting and unmounting routines. The scripts inherit the local identity and AFS tokens associated with to the issuer of the **butc** command.

You need to include a **MOUNT** instruction in the device configuration file for all tape devices, but the need for an **UNMOUNT** instruction depends on the tape-handling routines that the device's manufacturer provides. Some devices, usually stackers, have only a single routine for mounting tapes, which also automatically unmounts a tape whose presence prevents insertion of the required new tape. In this case, an **UNMOUNT** instruction is not necessary. For devices that have separate mounting and unmounting routines, you must include an **UNMOUNT** instruction to remove a tape when the Tape Coordinator is finished with it; otherwise, subsequent attempts to run the **MOUNT** instruction fail with an error.

When the device configuration file includes a **MOUNT** instruction, you must stock the stacker or jukebox with the necessary tapes before running a backup operation. Many jukeboxes are able to search for the required tape by reading external labels (such as barcodes) on the tapes, but many stackers can only switch between tapes in sequence and sometimes only in one direction. In the latter case, you must also stock the tapes in the correct order.

To obtain a list of the tapes required for a restore operation so that you can prestock them in the tape device, include the **-n** flag on the appropriate **backup** command (**backup diskrestore**, **backup volrestore**, or **backup volsetrestore**). For a dump operation, it is generally sufficient to stock the device with more tapes than the operation is likely to require. You can prelabel the tapes with permanent names or AFS tape names, or not prelabel them at all. If you prelabel the tapes for a dump operation with AFS tape names, then it is simplest to load them into the stacker in sequential order by tape index. But it is probably simpler still to prelabel tapes with permanent tape names or use unlabeled tapes, in which case the Backup System generates and applies the appropriately indexed AFS tape name itself during the dump operation.

How the Tape Coordinator Uses the MOUNT and UNMOUNT Instructions

When you issue the **butc** command to initialize the Tape Coordinator for a given tape device, the Tape Coordinator looks for the device configuration file called **/usr/afs/backup/CFG_**device_name on its local disk, where device_name has the format described in "Creating a Device Configuration File" on page 229. If the file exists and contains a **MOUNT** instruction, then whenever the Tape Coordinator needs a tape, it executes the script named by the instruction's filename argument.

If the device configuration file does not exist, or does not include a **MOUNT** instruction, then whenever the Tape Coordinator needs a tape, it generates a prompt in its window instructing the operator to insert the necessary tape. The operator must insert the tape and press **<Return>** before the Tape Coordinator continues the backup operation.

Note, however, that you can modify the Tape Coordinator's behavior with respect to the first tape needed for an operation, by setting the **AUTOQUERY** instruction in the device configuration file to **NO**, or including the **-noautoquery** flag to the **butc** command. In this case, the Tape Coordinator does not execute the **MOUNT** instruction or prompt for a tape at the start of an operation, because it expects to find the required first tape in the drive. See "Eliminating the Search or Prompt for the Initial Tape" on page 234.

If there is an **UNMOUNT** instruction in the device configuration file, then whenever the Tape Coordinator closes the tape device, it executes the script named by the instruction's filename argument. It executes the script only once, and regardless of whether the **close** operation on the device succeeded or not. If the device configuration file does not include an **UNMOUNT** instruction, then the Tape Coordinator takes no action.

The Available Parameters and Required Exit Codes

When the Tape Coordinator executes the **MOUNT** script, it passes in five parameters, ordered as follows. You can use the parameters in your script to refine its response to varying circumstances that can arise during a backup operation.

1. The tape device or backup data file's pathname, as recorded in the **/usr/afs/backup/tapeconfig** file.
2. The tape operation, which (except for the exceptions noted in the following list) matches the **backup** command operation code used to initiate the operation:
   - **appenddump** (when a **backup dump** command includes the **-append** flag)
   - **dump** (when a **backup dump** command does not include the **-append** flag)
   - **labeltape**
   - **readlabel**
   - **restore** (for a **backup diskrestore**, **backup volrestore**, or **backup volsetrestore** command)
   - **restoredb**
   - **savedb**
   - **scantape**
3. The number of times the Tape Coordinator has attempted to open the tape device or backup data file. If the open attempt returns an error, the Tape Coordinator increments this value by one and again invokes the **MOUNT** instruction.
4. The tape name. For some operations, the Tape Coordinator passes the string `none`, because it does not know the tape name (when running the **backup scantape** or **backup readlabel**, for example), or

because the tape does not necessarily have a name (when running the **backup labeltape** command, for example).

5. The tape ID recorded in the Backup Database. As with the tape name, the Backup System passes the string `none` for operations where it does not know the tape ID or the tape does not necessarily have an ID.

Your **MOUNT** script must return one of the following exit codes to tell the Tape Coordinator whether or not it mounted the tape successfully:

- Code **0** (zero) indicates a successful mount, and the Tape Coordinator continues the backup operation. If the script or program called by the **MOUNT** instruction does not return this exit code, the Tape Coordinator never calls the **UNMOUNT** instruction.
- Code **1** indicates that mount attempt failed. The Tape Coordinator terminates the backup operation.
- Any other code indicates that the script was unable to access the correct tape. The Tape Coordinator prompts the operator to insert it.

When the Tape Coordinator executes the **UNMOUNT** script, it passes in two parameters in the following order.

1. The tape device's pathname (as specified in the **/usr/afs/backup/tapeconfig** file)
2. The tape operation, which is always **unmount**.

The following example script uses two of the parameters passed to it by the Backup System: `tries` and `operation`. It follows the recommended practice of exiting if the value of the `tries` parameter exceeds one, because that implies that the stacker is out of tapes.

For a **backup dump** or **backup savedb** operation, the routine calls the example **stackerCmd_NextTape** function provided by the stacker's manufacturer. Note that the final lines in the file return the exit code that prompts the operator to insert a tape; these lines are invoked when either the stacker cannot load a tape or a the operation being performed is not one of those explicitly mentioned in the file (is a restore operation, for example).

```
#! /bin/csh -f
set devicefile = $1
set operation = $2
set tries = $3
set tapename = $4
set tapeid = $5
set exit_continue = 0
set exit_abort = 1
set exit_interactive = 2
#--------------------------------------------
if (${tries} > 1) then
   echo "Too many tries"
   exit ${exit_interactive}
endif
```

```
if (${operation} == "unmount") then
   echo "UnMount: Will leave tape in drive"
   exit ${exit_continue}
endif
if ((${operation} == "dump")     |\
    (${operation} == "appenddump")     |\
    (${operation} == "savedb"))  then
    stackerCmd_NextTape ${devicefile}
    if (${status} != 0)exit${exit_interactive}
    echo "Will continue"
    exit ${exit_continue}
endif
if ((${operation} == "labeltape")    |\
    (${operation} == "readlabel")) then
   echo "Will continue"
   exit ${exit_continue}
endif
echo "Prompt for tape"
exit ${exit_interactive}
```

Eliminating the Search or Prompt for the Initial Tape

By default, the Tape Coordinator obtains the first tape it needs for a backup operation by reading the device configuration file for the appropriate tape device. If there is a **MOUNT** instruction in the file, the Tape Coordinator executes the referenced script. If the device configuration file does not exist or does not have a **MOUNT** instruction in it, the Tape Coordinator prompts you to insert the correct tape and press **<Return>**.

If you know in advance that an operation requires a tape, you can increase efficiency by placing the required tape in the drive before issuing the **backup** command and telling the Tape Coordinator's to skip its initial tape-acquisition steps. This both enables the operation to begin more quickly and eliminates that need for you to be present to insert a tape.

There are two ways to bypass the Tape Coordinator's initial tape-acquisition steps:

1. Include the instruction **AUTOQUERY NO** in the device configuration file
2. Include the **-noautoquery** flag to the **butc** command

To avoid any error conditions that require operator attention, be sure that the tape you are placing in the drive does not contain any unexpired dumps and is not write protected. If there is no permanent name on the tape's label and you are creating an initial dump, make sure that the AFS tape name either matches the volume set and dump set names or is `<NULL>`. Alternatively, suppress the Tape Coordinator's name verification step by assigning the value **NO** to the **NAME_CHECK** instruction in the device configuration file, as described in "Eliminating the AFS Tape Name Check" on page 235.

Enabling Default Responses to Error Conditions

By default, the Tape Coordinator asks you how to respond when it encounters certain error conditions. To suppress the prompts and cause the Tape Coordinator to handle the errors in a predetermined manner, include the instruction **ASK NO** in the device configuration file. If you assign the value **YES**, or omit the **ASK** instruction completely, the Tape Coordinator prompts you for direction when it encounters one of the errors.

The following list describes the error conditions and the Tape Coordinator's response to them.

- The Backup System is unable to dump a volume while running the **backup dump** command. When you assign the value **NO**, the Tape Coordinator omits the volume from the dump and continues the operation. When you assign the value **YES**, it prompts to ask if you want to try to dump the volume again immediately, to omit the volume from the dump but continue the operation, or to terminate the operation.
- The Backup System is unable to restore a volume while running the **backup diskrestore**, **backup volrestore**, or **backup volsetrestore** command. When you assign the value **NO**, the Tape Coordinator continues the operation, omitting the problematic volume but restoring the remaining ones. When you assign the value **YES**, it prompts to ask if you want to omit the volume and continue the operation, or to terminate the operation.
- The Backup System cannot determine if the dump set includes any more tapes while running the **backup scantape** command (the command's reference page in the *IBM AFS Administration Reference* discusses possible reasons for this problem). When you assign the value **NO**, the Tape Coordinator proceeds as though there are more tapes and invokes the **MOUNT** script named in the device configuration file, or prompts the operator to insert the next tape. When you assign the value **YES**, it prompts to ask if there are more tapes to scan.
- The Backup System determines that the tape contains an unexpired dump while running the **backup labeltape** command. When you assign the value **NO**, it terminates the operation without relabeling the tape. With a **YES** value, the Tape Coordinator prompts to ask if you want to relabel the tape anyway.

Eliminating the AFS Tape Name Check

If a tape does not have a permanent name and you are writing an initial dump to it, then by default the Backup System verifies that the tape's AFS tape name is acceptable. It accepts three types of values:

- A name that reflects the volume set and dump level of the initial dump and the tape's place in the sequence of tapes for the dump set, as described in "Dump Names and Tape Names" on page 197. If the tape does not already have a permanent name, you can assign the AFS tape name by using the **-name** argument to the **backup labeltape** command.
- A `<NULL>` value, which results when you assign a permanent name, or provide no value for the **backup labeltape** command's **-name** argument.
- No AFS tape name at all, indicating that you have never labeled the tape or written a dump to it.

To bypass the name check, include the **NAME_CHECK NO** instruction in the device configuration file. This enables you to recycle a tape without first relabeling it, as long as all dumps on it are expired. (If a tape has unexpired dumps on it but you want to recycle it anyway, you must use the **backup labeltape** command to relabel it first. For this to work, the **ASK NO** instruction cannot appear in the device configuration file.)

Setting the Memory Buffer Size to Promote Tape Streaming

By default, the Tape Coordinator uses a 16-KB memory buffer during dump operations. As it receives volume data from the Volume Server, the Tape Coordinator gathers 16 KB of data in the buffer before transferring the entire 16 KB to the tape device. Similarly, during a restore operation the Tape Coordinator by default buffers 32 KB of data from the tape device before transferring the entire 32 KB to the Volume Server for restoration into the file system. Buffering makes the volume of data flowing to and from a tape device more even and so promotes tape streaming, which is the most efficient way for a tape device to operate.

In a normal network configuration, the default buffer sizes are usually large enough to promote tape streaming. If the network between the Tape Coordinator machine and file server machines is slow, it can help to increase the buffer size.

To determine if altering the buffer size is helpful for your configuration, observe the tape device in operation to see if it is streaming, or consult the manufacturer. To set the buffer size, include the **BUFFERSIZE** instruction in the device configuration file. It takes an integer value, and optionally units, in the following format:

```
BUFFERSIZE size[{k | K | m | M | g | G}]
```

where size specifies the amount of memory the Tape Coordinator allocates to use as a buffer during both dump and restore operations. The default unit is bytes, but use **k** or **K** to specify kilobytes, **m** or **M** for megabytes, and **g** or **G** for gigabytes. There is no space between the size value and the units letter.

Dumping Data to a Backup Data File

You can write dumps to a *backup data file* rather than to tape. This is useful if, for example, you want to transfer the data to a data-archiving system, such as a hierarchical storage management (HSM) system, that you use in conjunction with AFS and the Backup System. You can restore data from a backup data file into the file system as well. Using a backup data file is usually more efficient than issuing the equivalent **vos dump** and **vos restore** commands individually for multiple volumes.

Writing to a backup data file is simplest if it is on the local disk of the Tape Coordinator machine, but you can also write the file to an NFS-mounted partition that resides on a remote machine. It is even acceptable to write to a file in AFS, provided that the access control list (ACL) on its parent directory grants the necessary permissions, but it is somewhat circular to back up AFS data into AFS itself.

If the backup data file does not already exist when the Tape Coordinator attempts to write a dump to it, the Tape Coordinator creates it. For a restore operation to succeed, the file must exist and contain volume data previously written to it during a **backup dump** operation.

When writing to a backup data file, the Tape Coordinator writes data at 16 KB offsets. If a given block of data (such as the marker that signals the beginning or end of a volume) does not fill the entire 16 KB, the

Tape Coordinator still skips to the next offset before writing the next block. In the output of a **backup dumpinfo** command issued with the **-id** option, the value in the `Pos` column is the ordinal of the 16-KB offset at which the volume data begins, and so is not generally only one higher than the position number on the previous line, as it is for dumps to tape.

Before writing to a backup data file, you need to configure the file as though it were a tape device.

> **Note:** A file pathname, rather than a tape device name, must appear in the third field of the **/usr/afs/backup/tapeconfig** file when the **FILE YES** instruction appears in the device configuration file, and vice versa. If the **tapeconfig** file instead refers to a tape device, dump operations appear to succeed but are inoperative. You cannot restore data that you accidently dumped to a tape device while the **FILE** instruction was set to **YES**. In the same way, if the **FILE** instruction is set to **NO**, the **tapeconfig** entry must refer to an actual tape device.

To configure a backup data file

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

3. **Optional**. Issue the **backup** command to enter interactive mode.

   ```
   # backup
   ```

4. Choose the port offset number to assign to the file. If necessary, display previously assigned port offsets by issuing the **(backup) listhosts** command, which is fully described in "To display the list of configured Tape Coordinators" on page 208.

   ```
   backup> listhosts
   ```

   As for a tape device, acceptable values are the integers **0** (zero) through **58510** (the Backup System can track a maximum of 58,511 port offset numbers). Each port offset must be unique in the cell, but you can associate any number them with a single Tape Coordinator machine. You do not have to assign port offset numbers sequentially.

5. Issue the **(backup) addhost** command to register the backup data file's port offset in the Backup Database.

   ```
   backup> addhost <tape machine name> [<TC port offset>]
   ```

   where

addh

Is the shortest acceptable abbreviation of **addhost**.

tape machine name

Specifies the fully qualified hostname of the Tape Coordinator machine you invoke to write to the backup data file.

TC port offset

Specifies the file's port offset number. You must provide this argument unless the default value of **0** (zero) is appropriate.

6. Using a text editor, create an entry for the backup data file in the local **/usr/afs/backup/tapeconfig** file, using the standard syntax:

```
[capacity  filemark_size]  device_name   port_offset
```

where

capacity

Specifies the amount of space on the partition that houses the backup data file that you want to make available for the file. To avoid the complications that arise from filling up the partition, it is best to provide a value somewhat smaller than the actual amount of space you expect to be available when the dump operation runs, and never larger than the maximum file size allowed by the operating system.

Specify a numerical value followed by a letter that indicates units, with no intervening space. The letter **k** or **K** indicates kilobytes, **m** or **M** indicates megabytes, and **g** or **G** indicates gigabytes. If you omit the units letter, the default is kilobytes. If you leave this field empty, the Tape Coordinator uses the maximum acceptable value (2048 GB or 2 TB). Also leave the filemark_size field empty in that case.

filemark_size

Specify the value **0** (zero) or leave both this field and the capacity field empty. In the latter case, the Tape Coordinator also uses the value zero.

device_name

Specifies the complete pathname of the backup data file. Rather than specifying an actual file pathname, however, the recommended configuration is to create a symbolic link in the **/dev** directory that points to the actual file pathname, and record the symbolic link in this field. This configuration provides these advantages:

- It makes the device_name portion of the **CFG**_device_name, of the **TE**_device_name, and of the **TL**_device_name filenames as short as possible. Because the symbolic link is in the **/dev** directory as though it is a tape device, you strip off the entire **/dev/** prefix when forming the filename, instead of just the initial slash (**/**). If, for example, the symbolic link is called

/dev/FILE, the device configuration file's name is **CFG_FILE**, whereas if the actual pathname **/var/tmp/FILE** appears in the **tapeconfig** file, the configuration file's name must be **CFG_var_tmp_FILE**.

- It provides for a more graceful, and potentially automated, recovery if the Tape Coordinator cannot write a complete dump into the backup data file (for example, because the partition housing the backup data file becomes full). The Tape Coordinator's reaction to this problem is to invoke the **MOUNT** script, or to prompt you if the **MOUNT** instruction does not appear in the configuration file.
  - If there is a **MOUNT** script, you can prepare for this situation by adding a subroutine to the script that changes the symbolic link to point to another backup data file on a partition where there is space available.
  - If there is no **MOUNT** instruction, the prompt enables you manually to change the symbolic link to point to another backup data file and then press **<Return>** to signal that the Tape Coordinator can continue the operation.

If this field names the actual file, there is no way to recover from exhausting the space on the partition. You cannot change the **tapeconfig** file in the middle of an operation.

port_offset

Specifies the port offset number that you chose for the backup data file.

7. Create the device configuration file **CFG**_device_name in the Tape Coordinator machine's **/usr/afs/backup** directory. Include the **FILE YES** instruction in the file.

   Construct the device_name portion of the name based on the device name you recorded in the **tapeconfig** file in Step "6" on page 238. If, as recommended, you recorded a symbolic link name, strip off the **/dev/** string and replace any other slashes (**/**) in the name with underscores (**_**). For example, **CFG_FILE** is the appropriate name if the symbolic link is **/dev/FILE**. If you recorded the name of an actual file, then strip off the initial slash only and replace any other slashes in the name with underscores. For a backup data file called **/var/tmp/FILE**, the appropriate device configuration filename is **CFG_var_tmp_FILE**.

8. If you chose in Step "6" on page 238 to record a symbolic link name in the device_name field of the **tapeconfig** entry, then you must do one of the following:
   - Use the **ln -s** command to create the appropriate symbolic link in the **/dev** directory
   - Write a script that initializes the backup data file in this way, and include a **MOUNT** instruction in the device configuration file to invoke the script. An example script appears following these instructions.

You do not need to create the backup data file itself, because the Tape Coordinator does so if the file does not exist when the dump operation begins.

The following example script illustrates how you can automatically create a symbolic link to the backup data file during the preparation phase for writing to the file. When the Tape Coordinator is executing a **backup dump**, **backup restore**, **backup savedb**, or **backup restoredb** operation, the routine invokes the UNIX **ln -s** command to create a symbolic link from the backup data file named in the **tapeconfig** file to the actual file to use (this is the recommended method). It uses the values of the `tapename` and `tapeid` parameters passed to it by the Backup System when constructing the filename.

The routine makes use of two other parameters as well: `tries` and `operation`. The `tries` parameter tracks how many times the Tape Coordinator has attempted to access the file. A value greater than one indicates that the Tape Coordinator cannot access it, and the routine returns exit code 2 (`exit_interactive`), which results in a prompt for the operator to load a tape. The operator can use this opportunity to change the name of the backup data file specified in the **tapeconfig** file.

```
#! /bin/csh -f
set devicefile = $1
set operation = $2
set tries = $3
set tapename = $4
set tapeid = $5
set exit_continue = 0
set exit_abort = 1
set exit_interactive = 2
#--------------------------------------------
if (${tries} > 1) then
   echo "Too many tries"
   exit ${exit_interactive}
endif
if (${operation} == "labeltape") then
   echo "Won't label a tape/file"
   exit ${exit_abort}
endif
if ((${operation} == "dump")   |\
    (${operation} == "appenddump")   |\
    (${operation} == "restore")   |\
    (${operation} == "savedb")    |\
    (${operation} == "restoredb")) then
   /bin/rm -f ${devicefile}
   /bin/ln -s /hsm/${tapename}_${tapeid} ${devicefile}
   if (${status} != 0) exit ${exit_abort}
endif
exit ${exit_continue}
```

Chapter 7. Backing Up and Restoring AFS Data

The instructions in this chapter explain how to back up and restore AFS data and to administer the Backup Database. They assume that you have already configured all of the Backup System components by following the instructions in "Configuring the AFS Backup System" on page 195.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Enter interactive mode	**backup (interactive)**
Leave interactive mode	**(backup) quit**
List operations in interactive mode	**(backup) jobs**
Cancel operation in interactive mode	**(backup) kill**
Start Tape Coordinator	**butc**
Stop Tape Coordinator	**<Ctrl-c>**
Check status of Tape Coordinator	**backup status**
Back up data	**backup dump**
Display dump records	**backup dumpinfo**
Display volume's dump history	**backup volinfo**
Scan contents of tape	**backup scantape**
Restore volume	**backup volrestore**
Restore partition	**backup diskrestore**
Restore group of volumes	**backup volsetrestore**
Verify integrity of Backup Database	**backup dbverify**
Repair corruption in Backup Database	**backup savedb** and **backup restoredb**
Delete dump set from Backup Database	**backup deletedump**

Using the Backup System's Interfaces

When performing backup operations, you interact with three Backup System components:

- You initiate backup operations by issuing commands from the **backup** suite. You can issue the commands in a command shell (or invoke them in a shell script) on any AFS client or server machine from which you can access the **backup** binary. In the conventional configuration, the binary resides in the **/usr/afs/bin** directory on a server machine and the **/usr/afsws/etc** directory on a client machine.

  The suite provides an *interactive mode*, in which you can issue multiple commands over a persistent connection to the Backup Server and the Volume Location (VL) Server. Interactive mode has several convenient features. For a discussion and instructions, see "Using Interactive and Regular Command Mode" on page 243.

Note that some operating systems include a **backup** command of their own. You must configure machines that run such an operating system to ensure that you are accessing the desired **backup** binary.

- Before you perform a backup operation that involves reading or writing to a tape device or backup data file, you must open a dedicated connection to the appropriate Tape Coordinator machine and start the Tape Coordinator (**butc**) process that handles the device or file. The **butc** process must continue to run over the dedicated connection as long as it is executing an operation or is to be available to execute one. For further discussion and instructions, see "Starting and Stopping the Tape Coordinator Process" on page 247.
- The Backup Server (**buserver**) process must be running on database server machines, because most backup operations require accessing or changing information in the Backup Database. The *IBM AFS Quick Beginnings* explains how to configure the Backup Server.

For consistent Backup System performance, the AFS build level of all three binaries (**backup**, **butc**, and **buserver**) must match. For instructions on displaying the build level, see "Displaying A Binary File's Build Level" on page 87.

Performing Backup Operations as the Local Superuser Root or in a Foreign Cell

By default, the volumes and Backup Database involved in a backup operation must reside on server machines that belong to the cell named in the **/usr/vice/etc/ThisCell** files on both the Tape Coordinator machine and the machine where you issue the **backup** command. Also, to issue most **backup** commands you must have AFS tokens for an identity listed in the local cell's **/usr/afs/etc/UserList** file (which by convention is the same on every server machine in a cell). You can, however, perform backup operations on volumes or the Backup Database from a foreign cell, or perform backup operations while logged in as the local superuser **root** rather than as a privileged AFS identity.

To perform backup operations on volumes that reside in a foreign cell using machines from the local cell, you must designate the foreign cell as the cell of execution for both the Tape Coordinator and the **backup** command interpreter. Use one of the two following methods. For either method, you must also have tokens as an administrator listed in the foreign cell's **/usr/afs/etc/UserList** file.

- Before issuing **backup** commands and the **butc** command, set the AFSCELL environment variable to the foreign cell name in both command shells.
- Include the **-cell** argument to the **butc** and all **backup** commands. If you include the argument on the **backup (interactive)** command, it applies to all commands issued during the interactive session.

To perform backup operations without having administrative AFS tokens, you must log on as the local superuser **root** on both the Tape Coordinator machine and the machine where you issue **backup** commands. Both machines must be server machines, or at least have a **/usr/afs/etc/KeyFile** file that matches the file on other server machines. Then include the **-localauth** argument on both the **butc** command and all **backup** commands (or the **backup (interactive)** command). The Tape Coordinator and **backup** command interpreter construct a server ticket using the server encryption key with the highest key version number in the local **/usr/afs/etc/KeyFile** file, and present it to the Backup Server, Volume Server, and VL Server that belong to the cell named in the local **/usr/afs/etc/ThisCell** file. The ticket never expires.

You cannot combine the **-cell** and **-localauth** options on the same command. Also, each one overrides the local cell setting defined by the AFSCELL environment variable or the **/usr/vice/etc/ThisCell** file.

Using Interactive and Regular Command Mode

The **backup** command suite provides an interactive mode, in which you can issue multiple commands over a persistent connection to the Backup Server and the VL Server. Interactive mode provides the following features:

- The `backup>` prompt replaces the usual command shell prompt.
- You omit the initial **backup** string from command names. Type only the operation code and option names.
- You cannot issue commands that do not belong to the **backup** suite.
- If you assume an administrative AFS identity or specify a foreign cell as you enter interactive mode, it applies to all commands issued during the interactive session. See "Performing Backup Operations as the Local Superuser Root or in a Foreign Cell" on page 242.
- You do not need to enclose shell metacharacters in double quotes.

When you initiate a backup operation in interactive mode, the Backup System assigns it a *job ID number*. You can display the list of current and pending operations with the **(backup) jobs** command, for which instructions appear in "To display pending or running jobs in interactive mode" on page 244. (In both regular and interactive modes, the Tape Coordinator also assigns a *task ID number* to each operation you initiate with a **backup** command. You can track task ID numbers with the **backup status** command. See "Starting and Stopping the Tape Coordinator Process" on page 247.)

You can cancel an operation in interactive mode with the **(backup) kill** command, for which instructions appear in "To cancel operations in interactive mode" on page 246. However, it is best not to interrupt a dump operation because the resulting dump is incomplete, and interrupting a restore operation can leave volumes in an inconsistent state, or even completely remove them from the server machine. For further discussion, see "Backing Up Data" on page 251 and "Restoring and Recovering Data" on page 274.

The **(backup) jobs** and **(backup) kill** commands are available only in interactive mode and there is no equivalent functionality in regular command mode.

To enter interactive mode

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. Entering interactive mode does not itself require privilege, but most other **backup** commands do, and the AFS identity you assume when entering the mode applies to all commands you issue within it. If

necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **backup (interactive)** command at the system prompt. The `backup>` prompt appears. You can include either, but not both, of the **-localauth** and **-cell** options, as discussed in "Performing Backup Operations as the Local Superuser Root or in a Foreign Cell" on page 242.

```
% backup
backup>
```

To exit interactive mode

1. Issue the **quit** command at the `backup>` prompt. The command shell prompt reappears when the command succeeds, which it does only if there are no jobs pending or currently running. To display and cancel pending or running jobs, follow the instructions in "To display pending or running jobs in interactive mode" on page 244 and "To cancel operations in interactive mode" on page 246.

```
backup> quit
%
```

To display pending or running jobs in interactive mode

1. Issue the **jobs** command at the `backup>` prompt.

```
backup> jobs
```

where

j

Is the shortest acceptable abbreviation of **jobs**.

The output always includes the expiration date and time of the tokens that the **backup** command interpreter is using during the current interactive session, in the following format:

```
date   time: TOKEN EXPIRATION
```

If the execution date and time specified for a scheduled dump operation is later than *date time*, then its individual line (as described in the following paragraphs) appears below this line to indicate that the current tokens will not be available to it.

If the issuer of the **backup** command included the **-localauth** flag when entering interactive mode, the line instead reads as follows:

```
:  TOKEN NEVER EXPIRES
```

The entry for a scheduled dump operation has the following format:

```
   Job job_ID:  timestamp:  dump  volume_set  dump_level
```

where

job_ID

Is a job identification number assigned by the Backup System.

timestamp

Indicates the date and time the dump operation is to begin, in the format month/date/year hours:minutes (in 24-hour format)

volume_set

Indicates the volume set to dump.

dump_level

Indicates the dump level at which to perform the dump operation.

The line for a pending or running operation of any other type has the following format:

```
   Job job_ID:  operation  status
```

where

job_ID

Is a job identification number assigned by the Backup System.

operation

Identifies the operation the Tape Coordinator is performing, which is initiated by the indicated command:

`Dump` **(dump name)**

Initiated by the **backup dump** command. The dump name has the following format:

volume_set_name**.**dump_level_name

`Restore`

Initiated by the **backup diskrestore**, **backup volrestore**, or **backup volsetrestore** command.

`Labeltape` **(tape_label)**

Initiated by the **backup labeltape** command. The tape_label is the name specified by the **backup labeltape** command's **-name** or **-pname** argument.

`Scantape`

Initiated by the **backup scantape** command.

`SaveDb`

Initiated by the **backup savedb** command.

`RestoreDb`

Initiated by the **backup restoredb** command.

status

Indicates the job's current status in one of the following messages. If no message appears, the job is either still pending or has finished.

number `Kbytes, volume volume_name`

For a running dump operation, indicates the number of kilobytes copied to tape or a backup data file so far, and the volume currently being dumped.

number `Kbytes, restore.volume`

For a running restore operation, indicates the number of kilobytes copied into AFS from a tape or a backup data file so far.

`[abort requested]`

The **(backup) kill** command was issued, but the termination signal has yet to reach the Tape Coordinator.

`[abort sent]`

The operation is canceled by the **(backup) kill** command. Once the Backup System removes an operation from the queue or stops it from running, it no longer appears at all in the output from the command.

`[butc contact lost]`

The **backup** command interpreter cannot reach the Tape Coordinator. The message can mean either that the Tape Coordinator handling the operation was terminated or failed while the operation was running, or that the connection to the Tape Coordinator timed out.

`[done]`

The Tape Coordinator has finished the operation.

`[drive wait]`

The operation is waiting for the specified tape drive to become free.

`[operator wait]`

The Tape Coordinator is waiting for the backup operator to insert a tape in the drive.

To cancel operations in interactive mode

1. Issue the **jobs** command at the `backup>` prompt, to learn the job ID number of the operation you want to cancel. For details, see "To display pending or running jobs in interactive mode" on page 244.

   ```
   backup> jobs
   ```

2. Issue the **(backup) kill** command to cancel the operation.

   ```
   backup> kill <job ID or dump set name>
   ```

   where

   k

   Is the shortest acceptable abbreviation of **kill**.

   job ID or dump set name

   Specifies either the job ID number of the operation to cancel, as reported by the **jobs** command, or for a dump operation only, the dump name in the format volume_set_name.dump_level_name.

Starting and Stopping the Tape Coordinator Process

Before performing a backup operation that reads from or writes to a tape device or backup data file, you must start the Tape Coordinator (**butc**) process that handles the drive or file. This section explains how to start, stop, and check the status of a Tape Coordinator process. To use these instructions, you must have already configured the Tape Coordinator machine and created a Tape Coordinator entry in the Backup Database, as instructed in "Configuring Tape Coordinator Machines and Tape Devices" on page 205.

The Tape Coordinator assigns a *task ID number* to each operation it performs. The number is distinct from the job ID number assigned by the **backup** command interpreter in interactive mode (which is discussed in "Using Interactive and Regular Command Mode" on page 243). The Tape Coordinator reports the task ID number in its onscreen trace and in the messages that it writes to its log and error files. To view the task ID numbers of a Tape Coordinator's running or pending operations, issue the **backup status** command.

To start a Tape Coordinator process

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file of the cell in which the Tape Coordinator is to access volume data and the Backup Database. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

Alternately, you can log into a file server machine as the local superuser **root** in Step "3" on page 248.

2. Verify that you can write to the Tape Coordinator's log and error files in the local **/usr/afs/backup** directory (the **TE**_device_name and **TL**_device_name files). If the log and error files do not already exist, you must be able to insert and write to files in the **/usr/afs/backup** directory.
3. Open a connection (using a command such as **telnet** or **rlogin**) to the Tape Coordinator machine that drives the tape device, or whose local disk houses the backup data file. The Tape Coordinator uses a devoted connection or window that must remain open for the Tape Coordinator to accept requests and while it is executing them.

   If you plan to include the **-localauth** flag to the **butc** command in the next step, log in as the local superuser **root**.

4. Issue the **butc** command to start the Tape Coordinator. You can include either, but not both, of the **-localauth** and **-cell** options, as discussed in "Performing Backup Operations as the Local Superuser Root or in a Foreign Cell" on page 242.

   ```
   % butc [<port offset>]  [-debuglevel <trace level>]  \
          [-cell <cellname>] [-noautoquery] [-localauth]
   ```

   where

   butc

   Must be typed in full.

   port offset

   Specifies the Tape Coordinator's port offset number. You must provide this argument unless the default value of **0** (zero) is appropriate.

   -debuglevel

   Specifies the type of trace messages that the Tape Coordinator writes to the standard output stream (stdout). Provide one of the following three values, or omit this argument to display the default type of messages (equivalent to setting a value of **0** [zero]):

   - **0**: The Tape Coordinator generates only the minimum number of messages necessary to communicate with the backup operator, including prompts for insertion of additional tapes and messages that indicate errors or the beginning or completion of operations.
   - **1**: In addition to the messages displayed at level **0**, the Tape Coordinator displays the name of each volume being dumped or restored.
   - **2**: In addition to the messages displayed at levels **0** and **1**, the Tape Coordinator displays all of the messages it is also writing to its log file (**/usr/afs/backup/TL**_device_name).

   cellname

   Names the cell in which to perform the backup operations (the cell where the relevant volumes reside and the Backup Server process is running). If you omit this argument, the Tape

Coordinator uses its home cell, as defined in the local **/usr/vice/etc/ThisCell** file. Do not combine this argument with the **-localauth** flag.

-noautoquery

Disables the Tape Coordinator's prompt for the first tape it needs for each operation. For a description of the advantages and consequences of including this flag, see "Eliminating the Search or Prompt for the Initial Tape" on page 234.

-localauth

Constructs a server ticket using a key from the local **/usr/afs/etc/KeyFile** file. The **butc** process presents it to the Backup Server, Volume Server, and VL Server during mutual authentication. You must be logged into a file server machine as the local superuser **root** to include this flag, and cannot combine it with the **-cell** argument.

To stop a Tape Coordinator process

1. Enter an interrupt signal such as **<Ctrl-c>** over the dedicated connection to the Tape Coordinator.

To check the status of a Tape Coordinator process

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **backup status** command.

```
% backup status [<TC port offset>]
```

where

st

Is the shortest acceptable abbreviation of **status**.

TC port offset

Specifies the Tape Coordinator's port offset number. You must provide this argument unless the default value of **0** (zero) is appropriate.

The following message indicates that the Tape Coordinator is not currently performing an operation:

```
Tape coordinator is idle
```

Otherwise, the output includes a message of the following format for each running or pending operation:

```
Task task_ID:  operation:   status
```

where

task_ID

Is a task identification number assigned by the Tape Coordinator. It begins with the Tape Coordinator's port offset number.

operation

Identifies the operation the Tape Coordinator is performing, which is initiated by the indicated command:

- `Dump` (the **backup dump** command)
- `Restore` (the **backup diskrestore**, **backup volrestore**, or **backup volsetrestore** commands)
- `Labeltape` (the **backup labeltape** command)
- `Scantape` (the **backup scantape** command)
- `SaveDb` (the **backup savedb** command)
- `RestoreDb` (the **backup restoredb** command)

status

Indicates the job's current status in one of the following messages.

number `Kbytes transferred, volume` volume_name

For a running dump operation, indicates the number of kilobytes copied to tape or a backup data file so far, and the volume currently being dumped.

number `Kbytes, restore.volume`

For a running restore operation, indicates the number of kilobytes copied into AFS from a tape or a backup data file so far.

`[abort requested]`

The **(backup) kill** command was issued, but the termination signal has yet to reach the Tape Coordinator.

`[abort sent]`

The operation is canceled by the **(backup) kill** command. Once the Backup System removes an operation from the queue or stops it from running, it no longer appears at all in the output from the command.

`[butc contact lost]`

The **backup** command interpreter cannot reach the Tape Coordinator. The message can mean either that the Tape Coordinator handling the operation was terminated or failed while the operation was running, or that the connection to the Tape Coordinator timed out.

`[done]`

The Tape Coordinator has finished the operation.

`[drive wait]`

The operation is waiting for the specified tape drive to become free.

`[operator wait]`

The Tape Coordinator is waiting for the backup operator to insert a tape in the drive.

If the Tape Coordinator is communicating with an XBSA server (a third-party backup utility that implements the Open Group's Backup Service API [XBSA]), the following message appears last in the output:

```
XBSA_program Tape coordinator
```

where XBSA_program is the name of the XBSA-compliant program.

Backing Up Data

This section explains how to use the **backup dump** command to back up AFS data to tape or to a backup data file. The instructions assume that you understand Backup System concepts and have already configured the Backup System according to the instructions in "Configuring the AFS Backup System" on page 195. Specifically, you must already have:

- Decided whether to dump data to tape or to a backup data file, and configured the Tape Coordinator machine and Tape Coordinator process appropriately. See "Configuring Tape Coordinator Machines and Tape Devices" on page 205 and "Dumping Data to a Backup Data File" on page 236.
- Defined a volume set that includes the volumes you want to dump together. See "Defining and Displaying Volume Sets and Volume Entries" on page 209.
- Defined the dump level in the dump hierarchy at which you want to dump the volume set. If it is an incremental dump level, you must have previously created a dump at its parent level. See "Defining and Displaying the Dump Hierarchy" on page 215.
- Created a device configuration file. Such a file is required for each tape stacker, jukebox device, or backup data file. You can also use it to configure the Backup System's automation features. See "Automating and Increasing the Efficiency of the Backup Process" on page 228.

The most basic way to perform a dump operation is to create an initial dump of a single volume set as soon as the appropriate Tape Coordinator is available, by providing only the required arguments to the **backup dump** command. Instructions appear in "To create a dump" on page 263. The command has several optional arguments that you can use to increase the efficiency and flexibility of your backup procedures:

- To append a dump to the end of a set of tapes that already contains other dumps, include the **-append** argument. Otherwise, the Backup System creates an initial dump. Appending dumps enables you to use a tape's full capacity and has other potentially useful features. For a discussion, see "Appending Dumps to an Existing Dump Set" on page 260.
- To schedule one or more dump operations to run at a future time, include the **-at** argument. For a discussion and instructions, see "Scheduling Dumps" on page 262.
- To initiate a number of dump operations with a single **backup dump** command, include the **-file** argument to name a file in which you have listed the commands. For a discussion and instructions, see "Appending Dumps to an Existing Dump Set" on page 260 and "Scheduling Dumps" on page 262.
- To generate a list of the volumes to be included in a dump, without actually dumping them, combine the **-n** flag with the other arguments to be used on the actual command.

Making Backup Operations More Efficient

There are several ways to make dump operations more efficient, less prone to error, and less disruptive to your users. Several of them also simplify the process of restoring data if that becomes necessary.

- It is best not to dump the read/write or read-only version of a volume, because no other users or processes can access a volume while it is being dumped. Instead, shortly before the dump operation begins, create a backup version of each volume to be dumped, and dump the backup version. Creating a Backup version usually makes the source volume unavailable for just a few moments (during which access attempts by other processes are blocked but do not fail). To automate the creation of backup volumes, you can create a **cron** process in the **/usr/afs/local/BosConfig** file on one or more server machines, setting its start time at a sufficient interval before the dump operation is to begin. Include the **-localauth** argument to the **vos backup** or **vos backupsys** command to enable it to run without administrative tokens. For instructions, see "To create and start a new process" on page 117.
- The volume set, dump level, and Tape Coordinator port offset you specify on the **backup dump** command line must be properly defined in the Backup Database. The Backup System checks the database before beginning a dump operation and halts the command immediately if any of the required entities are missing. If necessary, use the indicated commands:
  - To display volume sets, use the **backup listvolsets** command as described in "To display volume sets and volume entries" on page 213.
  - To display dump levels, use the **backup listdumps** command as described in "To display the dump hierarchy" on page 222.
  - To display port offsets, use the **backup listhosts** command as described in "To display the list of configured Tape Coordinators" on page 208.

- Ensure that a valid token corresponding to a privileged administrative identity is available to the Backup System processes both when the **backup dump** command is issued and when the dump operation actually runs (for a complete description or the necessary privileges, see "Granting Administrative Privilege to Backup Operators" on page 204). This is a special concern for scheduled dumps. One alternative is to run **backup** commands (or the script that invokes them) and the **butc** command on server machines, and to include the **-localauth** argument on the command. In this case, the processes use the key with the highest key version number in the local **/usr/afs/etc/KeyFile** file to construct a token that never expires. Otherwise, you must use a method to renew tokens before they expire, or grant tokens with long lifetimes. In either case, you must protect against improper access to the tokens by securing the machines both physically and against unauthorized network access. The protection possibly needs to be even stronger than when a human operator is present during the operations.
- Record tape capacity and filemark size values that are as accurate as possible in the Tape Coordinator's **/usr/afs/backup/tapeconfig** file and on the tape's label. For suggested values and a description of what can happen when they are inaccurate, see "Configuring the tapeconfig File" on page 200.
- If an unattended dump requires multiple tapes, arrange to provide them by properly configuring a tape stacker or jukebox and writing a tape-mounting script to be invoked in the device's **CFG**_device_name file. For instructions, see "Invoking a Device's Tape Mounting and Unmounting Routines" on page 230.
- You can configure any tape device or backup data file's **CFG**_device_name file to take advantage of the Backup System's automation features. See "Automating and Increasing the Efficiency of the Backup Process" on page 228.
- When you issue a **backup** command in regular (noninteractive) mode, the command shell prompt does not return until the operation completes. To avoid having to open additional connections, issue the **backup dump** command in interactive mode, especially when including the **-at** argument to schedule dump operations.
- An incremental dump proceeds most smoothly if there is a dump created at the dump level immediately above the level you are using. If the Backup System does not find a Backup Database record for a dump created at the immediate parent level, it looks for a dump created at one level higher in the hierarchy, continuing up to the full dump level if necessary. It creates an incremental dump at the level one below the lowest valid parent dump that it finds, or even creates a full dump if that is necessary. This algorithm guarantees that the dump captures all data that has changed since the last dump, but has a couple of disadvantages. First, the Backup System's search through the database for a valid parent dump takes extra time. Second, the subsequent pattern of dumps can be confusing to a human operator who needs to restore data from them, because they were not performed at the expected dump levels.

  The easiest way to guarantee that a dump exists at the immediate parent level is always to perform dump operations on the predetermined schedule. To check that the parent dump exists, you can issue the **backup dumpinfo** command (as described in "To display dump records" on page 266) and search for it in the output. Alternatively, issue the **backup volinfo** command (as described in "To display a volume's dump history" on page 271) for a volume that you believe is in the parent dump.

- Always use dump levels from the same hierarchy (levels that are descendants of the same full level) when dumping a given volume set. The result of alternating between levels from different hierarchies can be confusing when you need to restore data or read dump records. It also increases the chance that changed data is not captured in any dump, or is backed up redundantly into more than one dump.

- Use permanent tape names rather than AFS tape names. You can make permanent names more descriptive than is allowed by an AFS tape name's strict format, and also bypass the name-checking step that the Backup System performs by default when a tape has an AFS tape name only. You can also configure the Tape Coordinator always to skip the check, however; for instructions and a description of the acceptable format for AFS tape names, see "Eliminating the AFS Tape Name Check" on page 235.
- If you write dumps to tape, restore operations are simplest if all of your tape devices are compatible (can read the same type of tape, at the same compression ratios, and so on). If you must use incompatible devices, then at least use compatible devices for all dumps performed at dump levels that are at the same depth in their respective hierarchies (compatible devices for all dumps performed at a full dump level, compatible devices for all dumps performed at a level 1 incremental dump level, and so on). The **-portoffset** argument to the **backup diskrestore** and **backup volsetrestore** commands accepts multiple port offset numbers, but uses the first listed port offset when restoring all full dumps, the second port offset when restoring all level 1 dumps, and so on. If you did not use compatible tape devices when creating dumps at the same depth in a hierarchy, you must restore one volume at a time with the **backup volrestore** command.
- In some cases, it makes sense to use a *temporary* volume set, which exists only within the context of the interactive session in which it is created and for which no record is created in the Backup Database. One suitable situation is when dumping a volume to tape in preparation for removing it permanently (perhaps because its owner is leaving the cell). In this case, you can define a volume entry that includes only the volume of interest without cluttering up the Backup Database with a volume set record that you are using only once.
- Do not perform a dump operation when you know that there are network, machine, or server process problems that can prevent the Backup System from accessing volumes or the Volume Location Database (VLDB). Although the Backup System automatically makes a number of repeated attempts to get to an inaccessible volume, the dump operation takes extra time and in some cases stops completely to prompt you for instructions on how to continue. Furthermore, if the Backup System's last access attempt fails and the volume is omitted from the dump, you must take extra steps to have it backed up (namely, the steps described just following for a halted dump operation). For a more complete description of how the Backup System makes repeated access attempts, see "How Your Configuration Choices Influence the Dump Process" on page 255.
- Review the logs created by the Backup System as soon as possible after a dump operation completes, particularly if it ran unattended. They name any volumes that were not successfully backed up, among other problems. The Backup Server writes to the **/usr/afs/logs/BackupLog** file on the local disk of the database server machine, and you can use the **bos getlog** command to read it remotely if you wish; for instructions, see "Displaying Server Process Log Files" on page 128. The Tape Coordinator writes to two files in the local **/usr/afs/backup** directory on the machine where it is running: the **TE_**device_name file records errors, and the **TL_**device_name file records both trace and error messages.
- Avoid halting a dump operation (for instance, by issuing the **(backup) kill** command in interactive mode), both because it introduces the potential for confusion and because recovering from the interruption requires extra effort. When a dump operation is interrupted, the volumes that were backed up before the halt signal is received are complete on the tape or in the backup data file, and are usable in restore operations. The records in the Backup Database about the volumes' dump history accurately

show when and at which dump level they were backed up; to display the records, use the **backup volinfo** command as described in "To display a volume's dump history" on page 271.

However, there is no indication in the dump's Backup Database record that volumes were omitted; to display the record, use the **backup dumpinfo** command as described in "To display dump records" on page 266. You must choose one of the following methods for dealing with the volumes that were not backed up before the dump operation halted. (Actually, you must make the same decision if the dump operation halts for reasons outside your control.)

- You can take no action, waiting until the next regularly scheduled dump operation to back them up. At that time, the Backup System automatically dumps them at the appropriate level to guarantee that the dump captures all of the data that changed since the volume was last dumped. However, you are gambling that restoring the volume is not necessary before the next dump operation. If restoration is necessary, you can restore the volume only to its state at the time it was last included in a dump--you have lost all changes made to the volume since that time.
- You can discard the entire dump and run the dump operation again. To discard the dump, use the **backup labeltape** command to relabel the tapes or backup data file, which automatically removes all associated records from the Backup Database. For instructions, see "Writing and Reading Tape Labels" on page 223. If a long time has passed since the backup version of the volumes was created, some of the source volumes have possibly changed. If that seems likely, reissue the **vos backup** or **vos backupsys** command on them before redoing the dump operation.
- You can create a new volume set that includes the missed volumes and dump it at a full dump level (even if you specify an incremental dump level, the Backup System uses the full dump level at the top of your specified level's hierarchy, because it has never before backed up these volumes as part of the new volume set). The next time you dump the original volume set, the Backup System automatically dumps the missed volumes at the level one below the level it used the last time it dumped the volumes as part of the original volume set.

How Your Configuration Choices Influence the Dump Process

This section provides an overview of the backup process, describing what happens at each stage both by default and as a result of your configuration choices, including the configuration instructions you include in the device-specific **CFG_**device_name file. For the sake of clarity, it tracks the progress of a single **backup dump** command that creates an initial dump. For a discussion of the slight differences in the procedure when you append or schedule dumps, see "Appending Dumps to an Existing Dump Set" on page 260 or "Scheduling Dumps" on page 262.

As a concrete example, the following description traces a dump of the volume set **user** at the **/weekly/mon/tues/wed** dump level. The **user** volume set has one volume entry that matches the backup version of all user volumes:

```
   .*    .*    user.*\.backup
```

The dump level belongs to the following dump hierarchy.

```
/weekly
       /mon
           /tues
                /wed
                    /thurs
                          /fri
```

1. You issue the **butc** command to start a Tape Coordinator to handle the dump operation. The Tape Coordinator does not have to be running when you issue the **backup dump** command, but must be active in time to accept the list of volumes to be included in the dump, when Step "3" on page 256 is completed. To avoid coordination problems, it is best to start the Tape Coordinator before issuing the **backup dump** command.

   As the Tape Coordinator initializes, it reads the entry in its local **/usr/afs/backup/tapeconfig** file for the port offset you specify on the **butc** command line. The entry specifies the name of the device to use, and the Tape Coordinator verifies that it can access it. It also reads the device's configuration file, **/usr/afs/backup/CFG_**device_name, if it exists. See Step "6" on page 258 for a description of how the instructions in the file influence the dump operation.

2. You issue the **backup dump** command, specifying a volume set, dump level, and the same port offset number you specified on the **butc** command in Step "1" on page 256. The Backup System verifies that they have correct Backup Database records and halts the operation with an error message if they do not.

   If you issue the command in interactive mode, the Backup System assigns the operation a job ID number, which you can use to check the operation's status or halt it by using the **(backup) jobs** or **(backup) kill** command, respectively. For instructions, see "To display pending or running jobs in interactive mode" on page 244 and "To cancel operations in interactive mode" on page 246.

3. The Backup System works with the VL Server to generate a list of the volumes in the VLDB that match the name and location criteria defined in the volume set's volume entries. If a volume matches more than one volume entry, the Backup System ignores the duplicates so that the dump includes only one copy of data from the volume.

   To reduce the number of times you need to switch tapes during a restore operation, the Backup System sorts the volumes by server machine and partition, and during the dump operation writes the data from all volumes stored on a specific partition before moving to the next partition.

   As previously mentioned, it is best to back up backup volumes rather than read/write volumes, to avoid blocking users' access to data during the dump. To achieve this, you must explicitly include the **.backup** suffix on the volume names in volume entry definitions. For instructions, and to learn how to define volume entries that match multiple volumes, see "Defining and Displaying Volume Sets and Volume Entries" on page 209.

   In the example, suppose that 50 volumes match the **user** volume set criteria, including three called **user.pat.backup**, **user.terry.backup**, and **user.smith.backup**.

4. The Backup System next scans the dump hierarchy for the dump level you have specified on the **backup dump** command line. If it is a full level, then in the current operation the Backup System backs up all of the data in all of the volumes in the list obtained in Step "3" on page 256.

   If the dump level is incremental, the Backup System reads each volume's dump history in the Backup Database to learn which of the parent levels in its pathname was used when the volume was most recently backed up as part of this volume set. In the usual case, it is the current dump level's immediate parent level.

   An incremental dump of a volume includes only the data that changed since the volume was included in the parent dump. To determine which data are eligible, the Backup System uses the concept of a volume's *clone date*. A read/write volume's clone date is when the Backup System locks the volume before copying its contents into a dump. A backup volume's clone date is the completion time of the operation that created it by cloning its read/write source volume (the operation initiated by a **vos backup** or **vos backupsys** command). A read-only volume's clone date is the time of the release operation (initiated by the **vos release** command) that completed most recently before the dump operation.

   More precisely then, an incremental dump includes only data that have a modification timestamp between the clone date of the volume included in the parent dump (the *parent clone date*) and the clone date of the volume to be included in the current dump (the *current clone date*).

   There are some common exceptions to the general rule that a volume's parent dump is the dump created at the immediate parent level:

   - The volume did not exist at all at the time of the last dump. In this case, the Backup System automatically does a full dump of it.
   - The volume did not match the volume set's name and location criteria at the time of the last dump. In this case, the Backup System automatically does a full dump of it, even if it was backed up recently (fully or incrementally) as part of another volume set. This redundancy is an argument for defining volume entries in terms of names rather than locations, particularly if you move volumes frequently.
   - The volume was not included in the dump at the immediate parent level for some reason (perhaps a process, machine, or network access prevented the Backup System from accessing it). In this case, the Backup System sets the clone date to the time of the last dump operation that included the volume. If the volume was not included in a dump performed at any of the levels in the current level's pathname, the Backup System does a full dump of it.

   In the example, the current dump level is **/weekly/mon/tues/wed**. The **user.pat.backup** and **user.terry.backup** volumes were included in the dump performed yesterday, Tuesday, at the **/weekly/mon/tues** level. The Backup System uses as their parent clone date 3:00 a.m. on Tuesday, which is when backup versions of them were created just before Tuesday's dump operation. However, Tuesday's dump did not include the **user.smith.backup** volume for some reason. The last time it was included in a dump was Monday, at the **/weekly/mon** level. The Backup System uses a parent clone date of Monday at 2:47 a.m., which is when a backup version of the volume was created just before the dump operation on Monday.

5. If performing an incremental dump, the Backup System works with the Volume Server to prepare a list of all of the files in each volume that have changed (have modification timestamps) between the parent clone date and the current clone date. The dump includes the complete contents of every such file. If a file has not changed, the dump includes only a placeholder stub for it. The dump also includes a copy of the complete directory structure in the volume, whether or not it has changed since the previous dump.

   If none of the data in the volume has changed since the last dump, the Backup System omits the volume completely. It generates the following message in the Tape Coordinator window and log files:

   ```
   Volume volume_name (volume_ID) not dumped - has not been modified
      since last dump.
   ```

6. The Tape Coordinator prepares to back up the data. If there is a **CFG**_device_name file, the Tape Coordinator already read it in Step "1" on page 256. The following list describes how the instructions in the file guide the Tape Coordinator's behavior at this point:

   FILE

   If this instruction is set to **YES**, the Tape Coordinator writes data to a backup data file. The device_name field in the **tapeconfig** file must also specify a filename for the dump to work properly. For further discussion and instructions on configuring a backup data file, see "Dumping Data to a Backup Data File" on page 236.

   If it is set to **NO** or does not appear in the file, the Tape Coordinator writes to a tape device.

   MOUNT and UNMOUNT

   If there is a **MOUNT** instruction in the file, each time the Tape Coordinator needs a new tape, it invokes the indicated script or program to mount a tape in the device's tape drive. There must be a **MOUNT** instruction if you want to utilize a tape stacker or jukebox's ability to switch between tapes automatically. If there is no **MOUNT** instruction, the Tape Coordinator prompts the human operator whenever it needs a tape.

   The **AUTOQUERY** instruction, which is described just following, modifies the Tape Coordinator's tape acquisition procedure for the first tape it needs in a dump operation.

   If there is an **UNMOUNT** instruction, then the Tape Coordinator invokes the indicated script or program whenever it closes the tape device. Not all tape devices have a separate tape unmounting routine, in which case the **UNMOUNT** instruction is not necessary. For more details on both instructions, see "Invoking a Device's Tape Mounting and Unmounting Routines" on page 230.

   AUTOQUERY

   If this instruction is set to **NO**, the Tape Coordinator assumes that the first tape needed for the dump operation is already in the tape drive. It does not use its usual tape acquisition procedure as described in the preceding discussion of the **MOUNT** instruction. You can achieve the same effect by including the **-noautoquery** flag to the **butc** command.

If this instruction is absent or set to **YES**, the Tape Coordinator uses its usual tape acquisition procedure even for the first tape. For more details, see "Eliminating the Search or Prompt for the Initial Tape" on page 234.

BUFFERSIZE

If this instruction appears in the file, the Tape Coordinator sets its buffer size to the specified value rather than using the default buffer size of 16 KB. For further discussion, see "Setting the Memory Buffer Size to Promote Tape Streaming" on page 236.

If there is no **CFG**_device_name file, the Tape Coordinator writes data to a tape device and prompts the human operator each time it needs a tape (the only exception being the first tape if you include the **-noautoquery** flag to the **butc** command).

7. The Tape Coordinator opens either a tape drive or backup data file at this point, as directed by the instructions in the **CFG**_device_name file (described in Step "6" on page 258). The instructions also determine whether it invokes a mount script or prompts the operator. In Step "1" on page 256 the Tape Coordinator read in the device's capacity and filemark size from the **tapeconfig** file. It now reads the same values from the tape or backup data file's magnetic label, and overwrites the **tapeconfig** values if there is a difference.

   If creating an initial dump (as in the current example) and there is no permanent name on the label, the Tape Coordinator next checks that the AFS tape name has one of the three acceptable formats. If not, it rejects the tape and you must use the **backup labeltape** command to write an acceptable name. You can bypass this name-checking step by including the **NAME_CHECK NO** instruction in the **CFG**_device_name file. For discussion and a list of the acceptable AFS tape name values, see "Eliminating the AFS Tape Name Check" on page 235.

8. For an initial dump, the Tape Coordinator starts writing at the beginning of the tape or backup dump file, overwriting any existing data. To prevent inappropriate overwriting, the Backup System first checks the Backup Database for any dump records associated with the name (permanent or AFS tape name) on the tape or backup dump file's label. It refuses to write to a backup data file that has unexpired dumps in it, or to a tape that belongs to a dump set with any unexpired dumps. To recycle a file or tape before all dumps have expired, you must use the **backup labeltape** command to relabel it. Doing so removes the Backup Database records of all dumps in the file or on all tapes in the dump set, which makes it impossible to restore data from any of the tapes. For more information on expiration dates, see "Defining Expiration Dates" on page 219.

   The Tape Coordinator also checks for two other types of inappropriate tape reuse. The tape cannot already have data on it that belongs to the dump currently being performed, because that implies that the previous tape is still in the drive, or you have mistakenly reinserted it. The Tape Coordinator generates the following message and attempts to obtain another tape:

   ```
   Can't overwrite tape containing the dump in progress
   ```

   The tape cannot contain data from a parent dump of the current (incremental) dump, because overwriting a parent dump makes it impossible to restore data from the current dump. The Tape Coordinator generates the following message and attempts to obtain another tape:

```
Can't overwrite the parent dump parent_name (parent_dump_ID)
```

9. The Tape Coordinator now writes data to the tape or backup data file. It uses the capacity and filemark size it obtained in Step "7" on page 259 as it tracks how much more space is available, automatically using its tape acquisition procedure if the dump is not finished when it reaches the end of the tape. For a more detailed description, and a discussion of what happens if the Tape Coordinator reaches the physical end-of-tape unexpectedly, see "Configuring the tapeconfig File" on page 200. Similarly, for instructions on configuring a backup data file to optimize recovery from unexpectedly running out of space, see Step "6" on page 238 in the instructions in "Dumping Data to a Backup Data File" on page 236.

   If the Tape Coordinator cannot access a volume during the dump (perhaps because of a server process, machine, or network outage), it skips the volume and continues dumping all volumes that it can access. It generates an error message in the Tape Coordinator window and log file about the omitted volume. It generates a similar message if it discovers that a backup volume has not been recloned since the previous dump operation (that is, that the volume's current clone date is the same as its parent clone date):

   ```
   Volume volume_name (volume_ID) not dumped - has not been re-cloned
       since last dump.
   ```

   After completing a first pass through all of the volumes, it attempts to dump each omitted volume again. It first checks to see if the reason that the volume was inaccessible during the first pass is that it has been moved since the VL Server generated the list of volumes to dump in Step "3" on page 256. If so, it dumps the volume from its new site. If the second attempt to access a volume also fails, the Tape Coordinator it generates the following message, prompting you for instruction on how to proceed:

   ```
   Dump of volume volume_name (volume_ID) failed
   Please select action to be taken for this volume.
      r - retry, try dumping this volume again
      o - omit, this volume from this dump
      a - abort, the entire dump
   ```

   To increase the automation of the dump process, you can include the **ASK NO** instruction in the **CFG_**device_name file to suppress this prompt and have the Tape Coordinator automatically omit the volume from the dump.

   If you are tracking the dump as it happens, the prompt enables you to take corrective action. If the volume has not been recloned, you can issue the **vos backup** command. If the volume is inaccessible, you can investigate and attempt to resolve the cause.

10. If the tape or backup data file does not already have an AFS tape name, the Backup System constructs the appropriate one and records it on the label and in the Backup Database. It also assigns a dump name and ID number to the dump and records them in dump record that it creates in the Backup Database. For details on tape and dump names, see "Dump Names and Tape Names" on page 197. For instructions on displaying dump records or a volume's dump history, or scanning the contents of a tape, see "Displaying Backup Dump Records" on page 266.

Appending Dumps to an Existing Dump Set

The AFS Backup System enables you to append dumps to the end of the final tape in a dump set by including the **-append** flag to the **backup dump** command. Appending dumps improves Backup System automation and efficiency in several ways:

- It maximizes use of a tape's capacity. An initial dump must always start on a new tape, but does not necessarily extend to the end of the final tape in the dump set. You can fill up the unused tape by appending one or more dumps.
- It can reduce the number of tapes and tape changes needed to complete a dump operation. Rather than performing a series of initial dumps first, instead begin with an initial dump and follow it immediately with several appended dumps. In this way you can write all dumps in the series to the same tape (assuming the tape is large enough to accommodate them all). If, in contrast, you perform all of the initial dumps first, each must begin on a new tape and you must switch tapes again if you then want to append dumps.

  You can either issue the appropriate series of **backup dump** commands at the interactive `backup>` prompt, or record them in a file that you then name with the **-file** argument to the **backup dump** command. Appending dumps in this way enables you to run multiple unattended backup operations even without a tape stacker or jukebox, if all of the dumps fit on one tape.
- It can reduce the number of tape changes during a restore operation. For example, if you append all of the incremental dumps of a volume set to tapes in one dump set, then restoring a volume from the volume set requires a minimum number of tape changes. It is best not to append incremental dumps to a tape that contains the parent full dump, however: if the tape is lost or damaged, you lose all of the data from the volume.

  Although it can be efficient to group together appended dumps that are related, the Backup System does not require any relationship between the appended dumps on a tape or in a dump set.

When writing an appended dump, the Backup System performs most of the steps described in "How Your Configuration Choices Influence the Dump Process" on page 255. Appended dumps do not have to be related to one another or the initial dump, so it skips Step "7" on page 259: there is no need to check that the AFS tape name reflects the volume set and dump level names in this case. It also skips Step "8" on page 259. Because it is not overwriting any existing data on the tape, it does not need to check the expiration dates of existing dumps on the tape or in the file. Then in Step "9" on page 259 the Tape Coordinator scans to the end of the last dump on the tape or in the backup data file before it begins writing data.

The Backup System imposes the following conditions on appended dumps:

- If writing to tape, the Tape Coordinator checks that it is the final one in a dump set for which there are complete and valid tape and dump records in the Backup Database. If not, it rejects the tape and requests an acceptable one. If you believe the tape has valid data on it, you can reconstruct the Backup Database dump records for it by using the **-dbadd** argument to the **backup scantape** command as instructed in "To scan the contents of a tape" on page 272.

- The most recent dump on the tape or in the backup data file must have completed successfully.
- The dump set to which the tape or file belongs must begin with an initial dump that is recorded in the Backup Database. If there are no dumps on the current tape, then the Backup System treats the dump operation as an initial dump and imposes the relevant requirements (for example, checks the AFS tape name if appropriate).

As you append dumps, keep in mind that all of a dump set's dump and tape records in the Backup Database are indexed to the initial dump. If you want to delete an appended dump's record, you must delete the initial dump record, and doing so erases the records of all dumps in the dump set. Without those records, you cannot restore any of the data in the dump set.

Similarly, all of the dumps in a dump set must expire before you can recycle (write a new initial dump to) any of the tapes in a dump set. Do not append a dump if its expiration date is later than the date on which you want to recycle any of the tapes in its dump set. To recycle a tape before the last expiration date, you must delete the initial dump's record from the Backup Database. Either use the **backup labeltape** command to relabel the tape as instructed in "To label a tape" on page 225, or use the **backup deletedump** command to delete the record directly as instructed in "To delete dump records from the Backup Database" on page 291.

Although in theory you can append as many dumps as you wish, it generally makes sense to limit the number of tapes in a dump set (for example, to five), for these reasons:

- If an unreadable spot develops on one of the tapes in a dump set, it can prevent the Tape Coordinator from scanning the tape as part of a **backup scantape** operation you use to reconstruct Backup Database records. The Tape Coordinator can almost always scan the tape successfully up to the point of damage and can usually skip past minor damage. A scanning operation can start on any tape in a dump set, so damage on one tape does not prevent scanning of the others in the dump set. However, you can scan only the tapes that precede the damaged one in the dump set or the ones that follow the damaged one, but not both. (For more information on using tapes to reconstruct the information in the Backup Database, see "To scan the contents of a tape" on page 272.)

  An unreadable bad spot can also prevent you from restoring a volume completely, because restore operations must begin with the full dump and continue with each incremental dump in order. If you cannot restore a specific dump, you cannot restore any data from later incremental dumps.

- If you decide in the future to archive one or more dumps, then you must archive the entire set of tapes that constitute the dump set, rather than just the ones that contain the data of interest. This wastes both tape and archive storage space. For more information on archiving, see "Archiving Tapes" on page 218.

Scheduling Dumps

By default, the Backup System starts executing a dump operation as soon as you enter the **backup dump** command, and the Tape Coordinator begins writing data as soon as it is not busy and the list of files to write is available. You can, however, schedule a dump operation to begin at a specific later time:

- To schedule a single dump operation, include the **-at** argument to specify its start time.
- To schedule multiple dump operations, list the operations in a file named by the **-file** argument and use the **-at** argument to specify when the **backup** command interpreter reads the file. If you omit the **-at** argument, the command interpreter reads the file immediately, which does not count as scheduling, but does allow you to initiate multiple dump operations in a single command. Do not combine the **-file** argument with the **-volumeset**, **-dump**, **-portoffset**, **-append**, or **-n** options.

  For file-formatting instructions, see the description of the **-file** argument in Step "7" on page 264 of "To create a dump" on page 263.

The Backup System performs initial and appended dumps in the same manner whether they are scheduled or begin running as soon as you issue the **backup dump** command. The only difference is that the requirements for successful execution hold both at the time you issue the command and when the Backup System actually begins running it. All required Backup Database entries for volume sets, dump levels, and port offsets, and all dump and tape records must exist at both times. Perhaps more importantly, the required administrative tokens must be available at both times. See "Making Backup Operations More Efficient" on page 252.

To create a dump

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

   ```
   % butc [<port offset>] [-noautoquery]
   ```

3. If using a tape device, insert the tape.
4. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

5. Decide which volume set and dump level to use. If necessary, issue the **backup listvolsets** and **backup listdumps** commands to display the existing volume sets and dump levels. For complete instructions and a description of the output, see "To display volume sets and volume entries" on page 213 and "To display the dump hierarchy" on page 222.

   ```
   backup> listvolsets  [<volume set name>]
   backup> listdumps
   ```

If you want to use a temporary volume set, you must create it during the current interactive session. This can be useful if you are dumping a volume to tape in preparation for removing it permanently (perhaps because its owner is leaving the cell). In this case, you can define a volume entry that includes only the volume of interest without cluttering up the Backup Database with a volume set record that you are using only once. Complete instructions appear in "Defining and Displaying Volume Sets and Volume Entries" on page 209.

```
backup>  addvolset <volume set name> -temporary
backup> addvolentry  -name <volume set name>  \
                     -server <machine name>  \
                     -partition <partition name>  \
                     -volumes <volume name (regular expression)>
```

6. If you are creating an initial dump and writing to a tape or backup data file that does not have a permanent name, its AFS tape name must satisfy the Backup System's format requirements as described in "Eliminating the AFS Tape Name Check" on page 235. If necessary, use the **backup readlabel** command to display the label and the **backup labeltape** command to change the names, as instructed in "Writing and Reading Tape Labels" on page 223. You must also relabel a tape if you want to overwrite it and it is part of a dump set that includes any unexpired dumps, though this is not recommended. For a discussion of the appropriate way to recycle tapes, see "Creating a Tape Recycling Schedule" on page 217.
7. Issue the **backup dump** command to dump the volume set.
   - To create one initial dump, provide only the volume set name, dump level name, and port offset (if not zero).
   - To create one appended dump, add the **-append** flag.
   - To schedule a single initial or appended dump, add the **-at** argument.
   - To initiate multiple dump operations, record the appropriate commands in a file and name it with the **-file** argument. Do not combine this argument with options other than the **-at** argument.

```
backup> dump <volume set name> <dump level name> [<TC port offset>]   \
             [-at <Date/time to start dump>+]  \
             [-append]  [-n] [-file <load file>]
```

where

dump

Must be typed in full.

volume set name

Names the volume set to dump.

dump level name

Specifies the complete pathname of the dump level at which to dump the volume set.

TC port offset

Specifies the port offset number of the Tape Coordinator process that is handling the operation. You must provide this argument unless the default value of 0 (zero) is appropriate.

-at

Specifies the date and time in the future at which to run the command, or to read the file named by the **-file** argument. Provide a value in the format mm/dd/yyyy [hh:MM], where the month (mm), day (dd), and year (yyyy) are required. Valid values for the year range from **1970** to **2037**; higher values are not valid because the latest possible date in the standard UNIX representation is in February 2038. The Backup System automatically reduces any later date to the maximum value in 2038.

The hour and minutes (hh:MM) are optional, but if provided must be in 24-hour format (for example, the value **14:36** represents 2:36 p.m.). If you omit them, the time defaults to midnight (00:00 hours).

As an example, the value **04/23/1999 20:20** schedules the command for 8:20 p.m. on 23 April 1999.

Note: A plus sign follows this argument in the command's syntax statement because it accepts a multiword value which does not need to be enclosed in double quotes or other delimiters, not because it accepts multiple dates. Provide only one date (and optionally, time) definition.

-append

Creates an appended dump by scanning to the end of the data from one or more previous dump operations that it finds on the tape or in the backup data file.

-n

Displays the names of all volumes to be included in the indicated dump, without actually writing data to tape or the backup data file. Combine this flag with the arguments you plan to use on the actual command, but not with the **-file** argument.

-file

Specifies the local disk or AFS pathname of a file containing **backup** commands. The Backup System reads the file immediately, or at the time specified by the **-at** argument if it is provided. A partial pathname is interpreted relative to the current working directory.

Place each **backup dump** command on its own line in the indicated file, using the same syntax as for the command line, but without the word **backup** at the start of the line. Each command must include the volume set name and dump level name arguments plus the TC port offset argument if the default value of zero is not appropriate. Commands in the file can also include any of the **backup dump** command's optional arguments, including the **-at** argument (which must specify a date and time later than the date and time at which the Backup System reads the file).

8. If you did not include the **-noautoquery** flag when you issued the **butc** command, or if the device's **CFG_**device_name configuration file includes the instruction **AUTOQUERY YES**, then the Tape Coordinator prompts you to place the tape in the device's drive. You have already done so, but you must now press **<Return>** to indicate that the tape is ready for labeling.

   If more than one tape is required, you must either include the **MOUNT** instruction in the **CFG_**device_name file and stock the corresponding stacker or jukebox with tapes, or remain at the console to respond to the Tape Coordinator's prompts for subsequent tapes.

9. After the dump operation completes, review the Backup System's log files to check for errors. Use the **bos getlog** command as instructed in "Displaying Server Process Log Files" on page 128 to read the **/usr/afs/logs/BackupLog** file, and a text editor on the Tape Coordinator machine to read the **TE_**device_name and **TL_**device_name files in the local **/usr/afs/backup** directory.

   It is also a good idea to record the tape name and dump ID number on the exterior label of each tape.

Displaying Backup Dump Records

The **backup** command suite includes three commands for displaying information about data you have backed up:

- To display information about one or more dump operations, such as the date it was performed and the number of volumes included, use the **backup dumpinfo** command as described in "To display dump records" on page 266. You can display a detailed record of a single dump or more condensed records for a certain number of dumps, starting with the most recent and going back in time. You can specify the number of dumps or accept the default of 10.
- To display a volume's dump history, use the **backup volinfo** command as described in "To display a volume's dump history" on page 271.
- To display information extracted from a tape or backup data file about the volumes it includes, use the **backup scantape** command. To create new dump and tape records in the Backup Database derived from the tape and dump labels, add the **-dbadd** flag. For instructions, see "To scan the contents of a tape" on page 272.

To display dump records

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on

page 536.

```
% bos listusers <machine name>
```

2. Issue the **backup dumpinfo** command to list information about dumps recorded in the Backup Database.

```
% backup dumpinfo [-ndumps <no. of dumps>]  [-id <dump id>]  [-verbose]
```

where

dump

Is the shortest acceptable abbreviation of **dumpinfo**.

-ndumps

Displays the Backup Database record for each of the specified number of dumps, starting with the most recent and going back in time. If the database contains fewer dumps than are requested, the output includes the records for all existing dumps. Do not combine this argument with the **-id** argument or **-verbose** flag; omit all three options to display the records for the last 10 dumps.

-id

Specifies the dump ID number of a single dump for which to display the Backup Database record. You must include the **-id** switch. Do not combine this option with the **-ndumps** or **-verbose** arguments; omit all three arguments to display the records for the last 10 dumps.

-verbose

Provides more detailed information about the dump specified with the **-id** argument, which must be provided along with it. Do not combine this flag with the **-ndumps** option.

If the **-ndumps** argument is provided, the output presents the following information in table form, with a separate line for each dump:

`dumpid`

The dump ID number.

`parentid`

The dump ID number of the dump's parent dump. A value of `0` (zero) identifies a full dump.

`lv`

The depth in the dump hierarchy of the dump level used to create the dump. A value of `0` (zero) identifies a full dump, in which case the value in the `parentid` field is also `0`. A value of `1` or greater indicates an incremental dump made at the corresponding level in the dump hierarchy.

created

The date and time at which the Backup System started the dump operation that created the dump.

nt

The number of tapes that contain the data in the dump. A value of `0` (zero) indicates that the dump operation was terminated or failed. Use the **backup deletedump** command to remove such entries.

nvols

The number of volumes from which the dump includes data. If a volume spans tapes, it is counted twice. A value of `0` (zero) indicates that the dump operation was terminated or failed; the value in the `nt` field is also `0` (zero) in this case.

dump name

The dump name in the form

```
volume_set_name.dump_level_name (initial_dump_ID)
```

where volume_set_name is the name of the volume set, and dump_level_name is the last element in the dump level pathname at which the volume set was dumped.

The initial_dump_ID, if displayed, is the dump ID of the initial dump in the dump set to which this dump belongs. If there is no value in parentheses, the dump is the initial dump in a dump set that has no appended dumps.

If the **-id** argument is provided alone, the first line of output begins with the string `Dump` and reports information for the entire dump in the following fields:

id

The dump ID number.

level

The depth in the dump hierarchy of the dump level used to create the dump. A value of `0` (zero) identifies a full dump. A value of `1` (one) or greater indicates an incremental dump made at the specified level in the dump hierarchy.

volumes

The number of volumes for which the dump includes data.

created

The date and time at which the dump operation began.

If an XBSA server was the backup medium for the dump (rather than a tape device or backup data file), the following line appears next:

```
Backup Service: XBSA_program: Server: hostname
```

where XBSA_program is the name of the XBSA-compliant program and hostname is the name of the machine on which the program runs.

Next the output includes an entry for each tape that houses volume data from the dump. Following the string `Tape`, the first two lines of each entry report information about that tape in the following fields:

`name`

The tape's permanent name if it has one, or its AFS tape name otherwise, and its tape ID number in parentheses.

`nVolumes`

The number of volumes for which this tape includes dump data.

`created`

The date and time at which the Tape Coordinator began writing data to this tape.

Following another blank line, the tape-specific information concludes with a table that includes a line for each volume dump on the tape. The information appears in columns with the following headings:

`Pos`

The relative position of each volume in this tape or file. On a tape, the counter begins at position 2 (the tape label occupies position 1), and increments by one for each volume. For volumes in a backup data file, the position numbers start with 1 and do not usually increment only by one, because each is the ordinal of the 16 KB offset in the file at which the volume's data begins. The difference between the position numbers therefore indicates how many 16 KB blocks each volume's data occupies. For example, if the second volume is at position 5 and the third volume in the list is at position 9, that means that the dump of the second volume occupies 64 KB (four 16-KB blocks) of space in the file.

`Clone time`

For a backup or read-only volume, the time at which it was cloned from its read/write source. For a Read/Write volume, it is the same as the dump creation date reported on the first line of the output.

`Nbytes`

The number of bytes of data in the dump of the volume.

`Volume`

The volume name, complete with `.backup` or `.readonly` extension if appropriate.

If both the **-id** and **-verbose** options are provided, the output is divided into several sections:

- The first section, headed by the underlined string `Dump`, includes information about the entire dump. The fields labeled `id`, `level`, `created`, and `nVolumes` report the same values (though in a different order) as appear on the first line of output when the **-id** argument is provided by itself. Other fields of potential interest to the backup operator are:

`Group id`

The dump's *group ID number*, which is recorded in the dump's Backup Database record if the **GROUPID** instruction appears in the Tape Coordinator's **/usr/afs/backup/CFG_**tcid file when the dump is created.

`maxTapes`

The number of tapes that contain the dump set to which this dump belongs.

`Start Tape Seq`

The ordinal of the tape on which this dump begins in the set of tapes that contain the dump set.

- For each tape that contains data from this dump, there follows a section headed by the underlined string `Tape`. The fields labeled `name`, `written`, and `nVolumes` report the same values (though in a different order) as appear on the second and third lines of output when the **-id** argument is provided by itself. Other fields of potential interest to the backup operator are:

  `expires`

  The date and time when this tape can be recycled, because all dumps it contains have expired.

  `nMBytes Data` and `nBytes Data`

  Summed together, these fields represent the total amount of dumped data actually from volumes (as opposed to labels, filemarks, and other markers).

  `KBytes Tape Used`

  The number of kilobytes of tape (or disk space, for a backup data file) used to store the dump data. It is generally larger than the sum of the values in the `nMBytes Data` and `nBytes Data` fields, because it includes the space required for the label, file marks and other markers, and because the Backup System writes data at 16 KB offsets, even if the data in a given block doesn't fill the entire 16 KB.

- For each volume on a given tape, there follows a section headed by the underlined string `Volume`. The fields labeled `name`, `position`, `clone`, and `nBytes` report the same values (though in a different order) as appear in the table that lists the volumes in each tape when the **-id** argument is provided by itself. Other fields of potential interest to the backup operator are:

  `id`

  The volume ID.

  `tape`

  The name of the tape containing this volume data.

The following example command displays the Backup Database records for the five most recent dump operations.

```
% backup dump 5
```

```
   dumpid   parentid lv created          nt nvols dump name
924424000          0 0  04/18/1999 04:26  1    22 usr.sun (924424000)
924685000  924424000 1  04/21/1999 04:56  1    62 usr.wed (924424000)
924773000  924424000 1  04/22/1999 05:23  1    46 usr.thu (924424000)
924860000  924424000 1  04/23/1999 05:33  1    58 usr.fri (924424000)
925033000          0 0  04/25/1999 05:36  2    73 sys.week
```

To display a volume's dump history

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **backup volinfo** command to display a volume's dump history.

   ```
   % backup volinfo <volume name>
   ```

   where

   voli

   Is the shortest acceptable abbreviation of **volinfo**.

   volume name

   Names the volume for which to display the dump history. If you dumped the backup or read-only version of the volume, include the **.backup** or **.readonly** extension.

The output includes a line for each Backup Database dump record that mentions the specified volume, order from most to least recent. The output for each record appears in a table with six columns:

`dumpID`

The dump ID of the dump that includes the volume.

`lvl`

The depth in the dump hierarchy of the dump level at which the volume was dumped. A value of `0` indicates a full dump. A value of `1` or greater indicates an incremental dump made at the specified depth in the dump hierarchy.

`parentid`

The dump ID of the dump's parent dump. A value of `0` indicates a full dump, which has no parent; in this case, the value in the `lvl` column is also `0`.

`creation date`

The date and time at which the Backup System started the dump operation that created the dump.

`clone date`

For a backup or read-only volume, the time at which it was cloned from its read/write source. For a read/write volume, the same as the value in the `creation date` field.

`tape name`

The name of the tape containing the dump: either the permanent tape name, or an AFS tape name in the format *volume_set_name.dump_level_name.tape_index* where *volume_set_name* is the name of the volume set associated with the initial dump in the dump set of which this tape is a part; *dump_level_name* is the name of the dump level at which the initial dump was backed up; *tape_index* is the ordinal of the tape in the dump set. Either type of name can be followed by a dump ID in parentheses; if it appears, it is the dump ID of the initial dump in the dump set to which this appended dump belongs.

The following example shows part of the dump history of the backup volume **user.smith.backup**:

```
% backup volinfo user.smith.backup
DumpID    lvl parentID  creation   date  clone date       tape name
924600000 1   924427600 04/20/1999 05:20 04/20/1999 05:01 user_incr_2 (924514392)
924514392 1   924427600 04/19/1999 05:33 04/19/1999 05:08 user_incr_2
924427600 0           0 04/18/1999 05:26 04/18/1999 04:58 user_full_6
    .     .      .          .       .        .      .         .
    .     .      .          .       .        .      .         .
```

To scan the contents of a tape

Note: The ability to scan a tape that is corrupted or damaged depends on the extent of the damage and what type of data is corrupted. The Backup System can almost always scan the tape successfully up to the point of damage. If the damage is minor, the Backup System can usually skip over it and scan the rest of the tape, but more major damage can prevent further scanning. A scanning operation does not have to begin with the first tape in a dump set, but the Backup System can process tapes only in sequential order after the initial tape provided. Therefore, damage on one tape does not prevent scanning of the others in the dump set, but it is possible to scan either the tapes that precede the damaged one or the ones that follow it, not both.

If you use the **-dbadd** flag to scan information into the Backup Database and the first tape you provide is not the first tape in the dump set, the following restrictions apply:

- If the first data on the tape is a continuation of a volume that begins on the previous (unscanned) tape in the dump set, the Backup System does not add a record for that volume to the Backup Database.

- The Backup System must read the marker that indicates the start of an appended dump to add database records for the volumes in it. If the first volume on the tape belongs to an appended dump, but is not immediately preceded by the appended-dump marker, the Backup System does not create a Backup Database record for it or any subsequent volumes that belong to that appended dump.

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

   ```
   % butc [<port offset>] [-noautoquery]
   ```

3. If scanning a tape, place it in the drive.
4. **(Optional)** Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

5. Issue the **backup scantape** command to read the contents of the tape.

   ```
   backup> scantape [-dbadd] [-portoffset <TC port offset>]
   ```

   where

   sc

   Is the shortest acceptable abbreviation of **scantape**.

   -dbadd

   Constructs dump and tape records from the tape and dump labels in the dump and writes them into the Backup Database.

   TC port offset

   Specifies the port offset number of the Tape Coordinator process that is handling the operation. You must provide this argument unless the default value of 0 (zero) is appropriate.

6. If you did not include the **-noautoquery** flag when you issued the **butc** command, or the device's **CFG_**device_name configuration file includes the instruction **AUTOQUERY YES** instruction, then the Tape Coordinator prompts you to place the tape in the device's drive. You have already done so, but you must now press **<Return>** to indicate that the tape is ready for reading.

To terminate a tape scanning operation, use a termination signal such as **<Ctrl-c>**, or issue the **(backup) kill** command in interactive mode. It is best not to interrupt the scan if you included the **-dbadd** argument. If the Backup System has already written new records into the Backup Database, then you must remove them before rerunning the scanning operation. If during the repeated scan operation the Backup System finds that a record it needs to create already exists, it halts the operation.

For each dump on the tape, the output in the Tape Coordinator window displays the dump label followed by an entry for each volume. There is no output in the command window. The dump label has the same fields as the tape label displayed by the **backup readlabel** command, as described in "Writing and Reading Tape Labels" on page 223. Or see the *IBM AFS Administration Reference* for a detailed description of the fields in the output.

The following example shows the dump label and first volume entry on the tape in the device that has port offset 2:

```
% backup scantape 2
-- Dump label --
tape name = monthly_guest
AFS tape name = guests.monthly.3
creationTime =  Mon Feb  1 04:06:40 1999
cell = abc.com
size = 2150000 Kbytes
dump path = /monthly
dump id = 917860000
useCount = 44
-- End of dump label --
-- volume --
volume name: user.guest10.backup
volume ID 1937573829
dumpSetName: guests.monthly
dumpID 917860000
level 0
parentID 0
endTime 0
clonedate Mon Feb  1 03:03:23 1999
```

Restoring and Recovering Data

The purpose of making backups is to enable you to recover when data becomes corrupted or is removed accidentally, returning the data to a coherent past state. The AFS Backup System provides three commands that restore varying numbers of volumes:

- To restore one or more volumes to a single site (partition on an AFS file server machine), use the **backup volrestore** command.
- To restore one or more volumes that are defined as a volume set, each to a specified site, use the **backup volsetrestore** command.
- To restore an entire partition (that is, all of the volumes that the VLDB lists as resident on it), use the **backup diskrestore** command.

The commands are suited to different purposes because they vary in the combinations of features they offer and in the requirements they impose. To decide which is appropriate for a specific restore operation, see the subsequent sections of this introduction: "Using the backup volrestore Command" on page 276, "Using the backup diskrestore Command" on page 279, and "Using the backup volsetrestore Command" on page 281.

Making Restore Operations More Efficient

The following comments apply to all types of restore operation:

- The Backup System begins by restoring the most recent full dump of a volume. As it restores subsequent incremental dumps, it alters the data in the full dump appropriately, essentially repeating the volume's change history. The **backup diskrestore** and **backup volsetrestore** commands always restore all incremental dumps, bringing a volume to its state at the time of the most recent incremental dump. You can use the **backup volrestore** command to return a volume to its state at a specified time in the past, by not restoring the data from incremental dumps performed after that time.
- The Backup System sets a restored volume's creation date to the date and time of the restore operation. The creation date appears in the `Creation` field of the output from the **vos examine** and **vos listvol** commands.
- When identifying the volumes to restore, it is best to specify the base (read/write) name. In this case, the Backup System searches the Backup Database for the most recent dump set that includes data from either the read/write or backup version of the volume, and restores dumps of that volume starting with the most recent full dump. If you include the **.backup** or **.readonly** extension on the volume name, the Backup System restores dumps of that version only. If it cannot find data dumped from that version, it does not perform the restoration even if another version was dumped.
- All three restoration commands accept the **-n** option, which generates a list of the volumes to be restored and the tapes or backup data files that contain the necessary dumps, without actually restoring data to AFS server partitions. This enables you to gather together the tapes before beginning the restore operation, even preloading them into a stacker or jukebox if you are using one.
- If you back up AFS data to tape, restoration is simplest if all of your tape devices are compatible, meaning that they can read the same type of tape, at the same compression ratios, and so on. (This suggestion also appears in "Making Backup Operations More Efficient" on page 252, because by the time you need to restore data it is too late to implement it.) You can still restore multiple volumes with a single command even if data was backed up using incompatible devices, because the **-portoffset** argument to all three restoration commands accepts multiple values. However, the Backup System uses the first port offset listed when restoring the full dump of each volume, the next port offset when restoring the level 1 incremental dump of each volume, and so on. If you did not use a compatible tape device when creating the full dump of every volume (and at each incremental level too), you cannot restore multiple volumes with a single command. You must use the **backup volrestore** command to restore one volume at a time, or use the **backup volsetrestore** command after defining volume sets that group volumes according to the tape device used to dump them.
- During a restore operation, the Backup System uses instructions in the relevant **CFG**_device_name configuration file in much the same way as during a dump operation, as described in "How Your Configuration Choices Influence the Dump Process" on page 255. It uses the **MOUNT**, **UNMOUNT**, **AUTOQUERY**, **BUFFERSIZE**, and **FILE** instructions just as for a dump operation. A difference for

the **BUFFERSIZE** instruction is that the default buffer size overridden by the instruction is 32 KB for restore operations rather than the 16 KB used for dump operations. The Backup System does not use the **NAME_CHECK** instruction at all during restore operations. The **ASK** instruction controls whether the Backup System prompts you if it cannot restore a volume for any reason. If the setting is **NO**, it skips the problematic volume and restores as many of the other volumes as possible.

- Do not perform a restore operation when you know that there are network, machine, or server process problems that can prevent the Backup System from accessing volumes or the VLDB. Although the Backup System automatically makes a number of repeated attempts to restore a volume, the restore operation takes extra time and in some cases stops completely to prompt you for instructions on how to continue.
- Avoid halting a restore operation (for instance by issuing the **(backup) kill** command in interactive mode). If a restore operation is interrupted for any reason, including causes outside your control, reissue the same restoration command as soon as is practical; if an outage or other problem caused the operation to halt, do not continue until the system returns to normal.

  Any volume that is completely restored when the operation halts is online and usable, but very few volumes are likely to be in this state. When restoring multiple volumes at once, the Backup System restores the full dump of every volume before beginning the level 1 incremental restore for any of them, and so on, completing the restore of every volume at a specific incremental level before beginning to restore data from the next incremental level. Unless a volume was dumped at fewer incremental levels than others being restored as part of the same operation, it is unlikely to be complete.

  It is even more dangerous to interrupt a restore operation if you are overwriting the current contents of the volume. Depending on how far the restore operation has progressed, it is possible that the volume is in such an inconsistent state that the Backup System removes it entirely. The data being restored is still available on tape or in the backup data file, but you must take extra steps to re-create the volume.

Using the backup volrestore Command

The **backup volrestore** command is most appropriate when you need to restore a few volumes to a single site (partition on a file server machine). By default, it restores the volumes to their state at the time of the most recent dump operation (this is termed a *full restore*). You can also use the command to perform a *date-specific restore*, which restores only the dumps (full and incremental) performed before a specified date and time, leaving the volume in the state it was in at the time of the final relevant incremental dump. The **backup diskrestore** and **backup volsetrestore** commands can only perform full restores.

You can restore data into a new copy of each volume rather than overwriting the current version, by including the **-extension** argument. After mounting the new volume in the filespace, you can compare the contents of the two and decide which to keep permanently.

The following list summarizes how to combine the **backup volrestore** command's arguments to restore a volume in different ways:

- To perform a date-specific restore as described just previously, use the **-date** argument to specify the date and optionally time. The Backup System restores the most recent full dump and each subsequent incremental dump for which the clone date of the volume included in the dump is before the indicated date and time (for a definition of the clone date, see Step "4" on page 256 in "How Your Configuration Choices Influence the Dump Process" on page 255). You can combine this argument with the **-extension** argument to place the date-specific restore in a new volume.

- To move a volume to a new site as you overwrite its contents with the restored data, use the **-server** and **-partition** arguments, singly or in combination, to specify the new site rather than the current site. The Backup System creates a new volume at that site, removes the existing volume, and updates the site information in the volume's VLDB entry. The volume's backup version is not removed automatically from the original site, if it exists. Use the **vos remove** command to remove it and the **vos backup** command to create a backup version at the new site.
- To create a new volume to house the restored data, rather than overwriting an existing volume, use the **-extension** argument. The Backup System creates the new volume on the server and partition named by the **-server** and **-partition** arguments, derives its name by adding the extension to the name specified with the **-volume** argument, and creates a new VLDB entry for it. The command does not affect the existing volume in any way. However, if a volume with the specified extension also already exists, the command overwrites it. To make the contents of the new volume accessible, use the **fs mkmount** command to mount it. You can then compare its contents to those of the existing volume, to see which to retain permanently.
- To restore a volume that no longer exists on an AFS server partition, but for which you have backed up data, specify the name of the new volume with the **-volume** argument and use the **-server** and **-partition** arguments to place it at the desired site. The Backup System creates a new volume and new VLDB entry.

To restore volumes with the backup volrestore command

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

   ```
   % butc [<port offset>] [-noautoquery]
   ```

   Repeat the command for each Tape Coordinator if you are using more than one tape device.

3. If using a tape device, insert the tape.
4. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

5. Issue the **backup volrestore** command with the desired arguments.

```
backup> volrestore <destination machine> <destination partition>  \
                   -volume <volume(s) to restore>+  \
                   [-extension <new volume name extension>]  \
                   [-date <date from which to restore>]  \
                   [-portoffset <TC port offsets>+] [-n]
```

where

volr

Is the shortest acceptable abbreviation of **volrestore**.

destination machine

Names the file server machine on which to restore each volume. It does not have to be a volume's current site.

destination partition

Names the partition on which to restore each volume. It does not have to be a volume's current site.

-volume

Names each volume to restore. It is best to provide the base (read/write) name, for the reasons discussed in "Making Restore Operations More Efficient" on page 274.

-extension

Creates a new volume to house the restored data, with a name derived by appending the specified string to each volume named by the **-volume** extension. The Backup System preserves the contents of the existing volume if it still exists. Do not use either of the **.readonly** or **.backup** extensions, which are reserved. The combination of base volume name and extension cannot exceed 22 characters in length. If you want a period to separate the extension from the name, specify it as the first character of the string (as in **.rst**, for example).

-date

Specifies a date and optionally time; the restored volume includes data from dumps performed before the date only. Provide a value in the format mm/dd/yyyy [hh:MM], where the required mm/dd/yyyy portion indicates the month (mm), day (dd), and year (yyyy), and the optional hh:MM portion indicates the hour and minutes in 24-hour format (for example, the value **14:36** represents 2:36 p.m.). If omitted, the time defaults to 59 seconds after midnight (00:00:59 hours).

Valid values for the year range from **1970** to **2037**; higher values are not valid because the latest possible date in the standard UNIX representation is in February 2038. The command interpreter automatically reduces any later date to the maximum value.

> **Note:** A plus sign follows this argument in the command's syntax statement because it accepts a multiword value which does not need to be enclosed in double quotes or other delimiters, not because it accepts multiple dates. Provide only one date (and optionally, time) definition.

-portoffset

Specifies one or more port offset numbers, each corresponding to a Tape Coordinator to use in the operation. If there is more than one value, the Backup System uses the first one when restoring the full dump of each volume, the second one when restoring the level 1 incremental dump of each volume, and so on. It uses the final value in the list when restoring dumps at the corresponding depth in the dump hierarchy and all dumps at lower levels.

Provide this argument unless the default value of 0 (zero) is appropriate for all dumps. If 0 is just one of the values in the list, provide it explicitly in the appropriate order.

-n

Displays the list of tapes that contain the dumps required by the restore operation, without actually performing the operation.

6. If you did not include the **-noautoquery** flag when you issued the **butc** command, or the device's **CFG_**device_name configuration file includes the instruction **AUTOQUERY YES**, then the Tape Coordinator prompts you to place the tape in the device's drive. You have already done so, but you must now press **<Return>** to indicate that the tape is ready for labeling.

   If more than one tape is required, you must either include the **MOUNT** instruction in the **CFG_**device_name file and stock the corresponding stacker or jukebox with tapes, or remain at the console to respond to the Tape Coordinator's prompts for subsequent tapes.

7. After the restore operation completes, review the Backup System's log files to check for errors. Use the **bos getlog** command as instructed in "Displaying Server Process Log Files" on page 128 to read the **/usr/afs/logs/BackupLog** file, and a text editor on the Tape Coordinator machine to read the **TE_**device_name and **TL_**device_name files in the local **/usr/afs/backup** directory.

Using the backup diskrestore Command

The **backup diskrestore** command is most appropriate when you need to restore all of the volumes on an AFS server partition, perhaps because a hardware failure has corrupted or destroyed all of the data. The command performs a full restore of all of the read/write volumes for which the VLDB lists the specified partition as the current site, using the dumps of either the read/write or backup version of each volume depending on which type was dumped more recently. (You can restore any backup or read-only volumes that resided on the partition by using the **vos backup** and **vos release** commands after the **backup diskrestore** operation is complete.)

By default, the Backup System restores the volumes to the site they previously occupied. To move the partition contents to a different site, use the **-newserver** and **-newpartition** arguments, singly or in combination.

By default, the Backup System overwrites the contents of existing volumes with the restored data. To create a new volume to house the restored data instead, use the **-extension** argument. The Backup System creates the new volume at the site designated by the **-newserver** and **-newpartition** arguments if they are used or the **-server** and **-partition** arguments otherwise. It derives the volume name by adding the extension to the read/write base name listed in the VLDB, and creates a new VLDB entry. The command does not affect the existing volume in any way. However, if a volume with the specified extension also already exists, the command overwrites it.

If a partition seems damaged, be sure not to run the **vos syncserv** command before the **backup diskrestore** command. As noted, the Backup System restores volumes according to VLDB site definitions. The **vos syncserv** command sometimes removes a volume's VLDB entry when the corruption on the partition is so severe that the Volume Server cannot confirm the volume's presence.

To restore a partition with the backup diskrestore command

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

   ```
   % butc [<port offset>] [-noautoquery]
   ```

   Repeat the command for each Tape Coordinator if you are using more than one tape device.

3. If using a tape device, insert the tape.
4. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

5. Issue the **backup diskrestore** command with the desired arguments.

   ```
   backup> diskrestore <machine to restore> <partition to restore>  \
                       [-portoffset <TC port offset>+]  \
                       [-newserver <destination machine>]  \
                       [-newpartition <destination partition>]  \
                       [-extension <new volume name extension>] [-n]
   ```

   where

   di

   Is the shortest acceptable abbreviation of **diskrestore**.

machine to restore

Names the file server machine that the VLDB lists as the site of the volumes that need to be restored.

partition to restore

Names the partition that the VLDB lists as the site of the volumes that need to be restored.

-portoffset

Specifies one or more port offset numbers, each corresponding to a Tape Coordinator to use in the operation. If there is more than one value, the Backup System uses the first one when restoring the full dump of each volume, the second one when restoring the level 1 incremental dump of each volume, and so on. It uses the final value in the list when restoring dumps at the corresponding depth in the dump hierarchy and all dumps at lower levels.

Provide this argument unless the default value of 0 (zero) is appropriate for all dumps. If 0 is just one of the values in the list, provide it explicitly in the appropriate order.

-newserver

Names an alternate file server machine to which to restore the volumes. If you omit this argument, the volumes are restored to the file server machine named by the **-server** argument.

-newpartition

Names an alternate partition to which to restore the data. If you omit this argument, the volumes are restored to the partition named by the **-partition** argument.

-extension

Creates a new volume for each volume being restored, to house the restored data, appending the specified string to the volume's read/write base name as listed in the VLDB. Any string other than **.readonly** or **.backup** is acceptable, but the combination of the base name and extension cannot exceed 22 characters in length. To use a period to separate the extension from the name, specify it as the first character of the string (as in **.rst**, for example).

-n

Displays a list of the tapes necessary to perform the requested restore, without actually performing the operation.

6. If you did not include the **-noautoquery** flag when you issued the **butc** command, or the device's **CFG**_device_name configuration file includes the instruction **AUTOQUERY YES**, then the Tape Coordinator prompts you to place the tape in the device's drive. You have already done so, but you must now press <**Return**> to indicate that the tape is ready for labeling.

   If more than one tape is required, you must either include the **MOUNT** instruction in the **CFG**_device_name file and stock the corresponding stacker or jukebox with tapes, or remain at the console to respond to the Tape Coordinator's prompts for subsequent tapes.

7. After the restore operation completes, review the Backup System's log files to check for errors. Use the **bos getlog** command as instructed in "Displaying Server Process Log Files" on page 128 to read the **/usr/afs/logs/BackupLog** file, and a text editor on the Tape Coordinator machine to read the **TE**_device_name and **TL**_device_name files in the local **/usr/afs/backup** directory.

Using the backup volsetrestore Command

The **backup volsetrestore** command is most appropriate when you need to perform a full restore of several read/write volumes, placing each at a specified site. You specify the volumes to restore either by naming a volume set with the **-name** argument or by listing each volume's name and restoration site in a file named by the **-file** argument, as described in the following sections.

Because the **backup volsetrestore** command enables you to restore a large number of volumes with a single command, the restore operation can potentially take hours to complete. One way to reduce the time is to run multiple instances of the command simultaneously. Either use the **-name** argument to specify disjoint volume sets for each command, or the **-file** argument to name files that list different volumes. You must have several Tape Coordinators available to read the required tapes. Depending on how the volumes to be restored were dumped to tape, specifying disjoint volume sets can also reduce the number of tape changes required.

Restoring a Volume Set with the -name Argument

Use the **-name** argument to restore a group of volumes defined in a volume set. The Backup System creates a list of the volumes in the VLDB that match the server, partition, and volume name criteria defined in the volume set's volume entries, and for which dumps are available. The volumes do not have to exist on the server partition as long as the VLDB still lists them (this can happen when, for instance, a hardware problem destroys the contents of an entire disk).

By default, the Backup System restores, as a read/write volume, each volume that matches the volume set criteria to the site listed in the VLDB. If a volume of the matching name exists at that site, its current contents are overwritten. You can instead create a new volume to house the restored data by including the **-extension** argument. The Backup System creates the new volume at the existing volume's site, derives its name by adding the extension to the existing volume's read/write base name, and creates a new VLDB entry for it. The command does not affect the existing volume in any way. However, if a volume with the specified extension also already exists, the command overwrites it. To make the contents of the new volume accessible, use the **fs mkmount** command to mount it. You can then compare its contents to those of the existing volume, to see which to retain permanently.

It is not required that the volume set was previously used to back up volumes (was used as the **-volumeset** option to the **backup dump** command). It can be defined especially to match the volumes that need to be restored with this command, and that is usually the better choice. Indeed, a *temporary* volume set, created by including the **-temporary** flag to the **backup addvolset** command, can be especially useful in this context (instructions appear in "Defining and Displaying Volume Sets and Volume Entries" on page 209). A temporary volume set is not added to the Backup Database and exists only during the current interactive backup session, which is suitable if the volume set is needed only to complete the single restore operation initialized by this command.

The reason that a specially defined volume set is probably better is that volume sets previously defined for use in dump operations usually match the backup version of volumes, whereas for a restore operation it is best to define volume entries that match the base (read/write) name. In this case, the Backup System searches the Backup Database for the newest dump set that includes a dump of either the read/write or the backup version of the volume. If, in contrast, a volume entry explicitly matches the volume's backup or read-only version, the Backup System uses dumps of that volume version only, restoring them to a read/write volume by stripping off the **.backup** or **.readonly** extension.

If there are VLDB entries that match the volume set criteria, but for which there are no dumps recorded in the Backup Database, the Backup System cannot restore them. It generates an error message on the standard error stream for each one.

Restoring Volumes Listed in a File with the -file Argument

Use the **-file** argument to specify the name and site of each read/write volume to restore. Each volume's entry must appear on its own (unbroken) line in the file, and comply with the following format:

```
machine    partition   volume    [comments...]
```

where

machine

Names the file server machine to which to restore the volume. You can move the volume as you restore it by naming a machine other than the current site.

partition

Names the partition to which to restore the volume. You can move the volume as you restore it by naming a partition other than the current site.

volume

Names the volume to restore. Specify the base (read/write) name to have the Backup System search the Backup Database for the newest dump set that includes a dump of either the read/write or the backup version of the volume. It restores the dumps of that version of the volume, starting with the most recent full dump. If, in contrast, you include the `.backup` or `.readonly` extension, the Backup System restores dumps of that volume version only, but into a read/write volume without the extension. The base name must match the name used in Backup Database dump records rather than in the VLDB, if they differ, because the Backup System does not consult the VLDB when you use the **-file** argument.

comments...

Is any other text. The Backup System ignores any text on each line that appears after the volume name, so you can use this field for helpful notes.

Do not use wildcards (for example, **.***) in the machine, partition, or volume fields. It is acceptable for multiple lines in the file to name the same volume, but the Backup System processes only the first of them.

By default, the Backup System replaces the existing version of each volume with the restored data, placing the volume at the site specified in the machine and partition fields. You can instead create a new volume to house the restored contents by including the **-extension** argument. The Backup System creates a new volume at the site named in the machine and partition fields, derives its name by adding the specified extension to the read/write version of the name in the volume field, and creates a new VLDB entry for it. The command does not affect the existing volume in any way. However, if a volume with the specified extension also already exists, the command overwrites it. To make the contents of the new volume accessible, use the **fs mkmount** command to mount it. You can then compare its contents to those of the existing volume, to see which to retain permanently.

If the file includes entries for volumes that have no dumps recorded in the Backup Database, the Backup System cannot restore them. It generates an error message on the standard error stream for each one.

One way to generate a file to use as input to the **-file** argument is to issue the command with the **-name** and **-n** options and direct the output to a file. The output includes a line like the following for each volume (shown here on two lines only for legibility reasons); the value comes from the source indicated in the following list:

```
machine   partition   volume_dumped  # as   volume_restored;    \
      tape_name   (tape_ID);   pos   position_number;   date
```

where

machine

Names the file server machine that currently houses the volume, as listed in the VLDB.

partition

Names the partition that currently houses the volume, as listed in the VLDB.

volume_dumped

Specifies the version (read/write or backup) of the volume that was dumped, as listed in the Backup Database.

volume_restored

Specifies the name under which the Backup System restores the volume when the **-n** flag is not included. If you include the **-extension** argument with the **-name** and **-n** options, then the extension appears on the name in this field (as in `user.pat.rst`, for example).

tape_name

Names the tape containing the dump of the volume, from the Backup Database. If the tape has a permanent name, it appears here; otherwise, it is the AFS tape name.

tape_ID

The tape ID of the tape containing the dump of the volume, from the Backup Database.

position_number

Specifies the dump's position on the tape (for example, `31` indicates that 30 volume dumps precede the current one on the tape). If the dump was written to a backup data file, this number is the ordinal of the 16 KB-offset at which the volume's data begins.

date

The date and time when the volume was dumped.

To make the entries suitable for use with the **-file** argument, edit them as indicated:

- The Backup System uses only the first three fields on each line of the input file, and so ignores all the fields after the number sign (`#`). You can remove them if it makes it easier for you to read the file, but that is not necessary.
- The volume_dumped (third) field of each line in the output file becomes the volume field in the input file. The Backup System restores data to read/write volumes only, so remove the `.backup` or `.readonly` extension if it appears on the name in the volume_dumped field.
- The output file includes a line for every dump operation in which a specific volume was included (the full dump and any incremental dumps), but the Backup System only processes the first line in the input file that mentions a specific volume. You can remove the repeated lines if it makes the file easier for you to read.
- The *machine* and *partition* fields on an output line designate the volume's current site. To move the volume to another location as you restore it, change the values.

To restore a group of volumes with the backup volsetrestore command

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

   ```
   % butc [<port offset>] [-noautoquery]
   ```

   Repeat the command for each Tape Coordinator if you are using more than one tape device.

3. If using a tape device, insert the tape.
4. Issue the **backup** command to enter interactive mode.

   ```
   % backup
   ```

5. **(Optional)** If appropriate, issue the **(backup) addvolset** command to create a new volume set expressly for this restore operation. Include the **-temporary** flag if you do not need to add the

volume set to the Backup Database. Then issue one or more **(backup) addvolentry** commands to create volume entries that include only the volumes to be restored. Complete instructions appear in "Defining and Displaying Volume Sets and Volume Entries" on page 209.

```
backup> addvolset <volume set name>  [-temporary]
backup> addvolentry  -name <volume set name>  \
                     -server <machine name>  \
                     -partition <partition name>  \
                     -volumes <volume name (regular expression)>
```

6. Issue the **backup volsetrestore** command with the desired arguments.

```
backup> volsetrestore [-name <volume set name>]  \
              [-file <file name>]  \
              [-portoffset <TC port offset>+]  \
              [-extension <new volume name extension>] [-n]
```

where

-name

Names a volume set to restore. The Backup System restores all of the volumes listed in the VLDB that match the volume set's volume entries, as described in "Restoring a Volume Set with the -name Argument" on page 282. Provide this argument or the **-file** argument, but not both.

-file

Specifies the full pathname of a file that lists one or more volumes and the site (file server machine and partition) to which to restore each. The input file has the format described in "Restoring Volumes Listed in a File with the -file Argument" on page 283. Use either this argument or the **-name** argument, but not both.

-portoffset

Specifies one or more port offset numbers, each corresponding to a Tape Coordinator to use in the operation. If there is more than one value, the Backup System uses the first one when restoring the full dump of each volume, the second one when restoring the level 1 incremental dump of each volume, and so on. It uses the final value in the list when restoring dumps at the corresponding depth in the dump hierarchy and all dumps at lower levels.

Provide this argument unless the default value of 0 (zero) is appropriate for all dumps. If 0 is just one of the values in the list, provide it explicitly in the appropriate order.

-extension

Creates a new volume for each volume being restored, to house the restored data, appending the specified string to the volume's read/write base name as listed in the VLDB. Any string other than **.readonly** or **.backup** is acceptable, but the combination of the base name and extension cannot exceed 22 characters in length. To use a period to separate the extension from the name, specify it as the first character of the string (as in **.rst**, for example).

-n

Displays a list of the volumes to be restored when the flag is not included, without actually restoring them. The **Output** section of this reference page details the format of the output. When combined with the **-name** argument, its output is easily edited for use as input to the **-file** argument on a subsequent **backup volsetrestore** command.

7. If you did not include the **-noautoquery** flag when you issued the **butc** command, or the device's **CFG**_device_name configuration file includes the instruction **AUTOQUERY YES**, then the Tape Coordinator prompts you to place the tape in the device's drive. You have already done so, but you must now press <**Return**> to indicate that the tape is ready for labeling.

   If more than one tape is required, you must either include the **MOUNT** instruction in the **CFG**_device_name file and stock the corresponding stacker or jukebox with tapes, or remain at the console to respond to the Tape Coordinator's prompts for subsequent tapes.

8. After the restore operation completes, review the Backup System's log files to check for errors. Use the **bos getlog** command as instructed in "Displaying Server Process Log Files" on page 128 to read the **/usr/afs/logs/BackupLog** file, and a text editor on the Tape Coordinator machine to read the **TE**_device_name and **TL**_device_name files in the local **/usr/afs/backup** directory.

Maintaining the Backup Database

The Backup Database stores all of the configuration and tracking information that the Backup System uses when dumping and restoring data. If a hardware failure or other problem on a database server machine corrupts or damages the database, it is relatively easy to recreate the configuration information (the dump hierarchy and lists of volume sets and Tape Coordinator port offset numbers). However, restoring the dump tracking information (dump records) is more complicated and time-consuming. To protect yourself against loss of data, back up the Backup Database itself to tape on a regular schedule.

Another potential concern is that the Backup Database can grow large rather quickly, because the Backup System keeps very detailed and cross-referenced records of dump operations. Backup operations become less efficient if the Backup Server has to navigate through a large number of obsolete records to find the data it needs. To keep the database to a manageable size, use the **backup deletedump** command to delete obsolete records, as described in "Removing Obsolete Records from the Backup Database" on page 291. If you later find that you have removed records that you still need, you can use the **backup scantape** command to read the information from the dump and tape labels on the corresponding tapes back into the database, as instructed in "To scan the contents of a tape" on page 272.

Backing Up and Restoring the Backup Database

Because of the importance of the information in the Backup Database, it is best to back it up to tape or other permanent media on a regular basis. As for the other AFS, administrative databases, the recommended method is to use a utility designed to back up a machine's local disk, such as the UNIX **tar** command. For instructions, see "Backing Up and Restoring the Administrative Databases" on page 78.

In the rare event that the Backup Database seems damaged or corrupted, you can use the **backup dbverify** command to check its status. If it is corrupted, use the **backup savedb** command to repair some types of damage. Then use the **backup restoredb** to return the corrected database to the local disks of the database server machines. For instructions, see "Checking for and Repairing Corruption in the Backup Database" on page 287.

Checking for and Repairing Corruption in the Backup Database

In rare cases, the Backup Database can become damaged or corrupted, perhaps because of disk or other hardware errors. Use the **backup dbverify** command to check the integrity of the database. If it is corrupted, the most efficient way to repair it is to use the **backup savedb** command to copy the database to tape. The command automatically repairs several types of corruption, and you can then use the **backup restoredb** command to transfer the repaired copy of the database back to the local disks of the database server machines.

The **backup savedb** command also removes *orphan blocks*, which are ranges of memory that the Backup Server preallocated in the database but cannot use. Orphan blocks do not interfere with database access, but do waste disk space. The **backup dbverify** command reports the existence of orphan blocks if you include the **-detail** flag.

To verify the integrity of the Backup Database

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **backup dbverify** command to check the integrity of the Backup Database.

   ```
   % backup dbverify [-detail]
   ```

   where

   db

   Is the shortest acceptable abbreviation of **dbverify**.

   -detail

   Reports the existence of orphan blocks and other information about the database, as described on the **backup dbverify** reference page in the *IBM AFS Administration Reference*.

   The output reports one of the following messages:

   - `Database OK` indicates that the Backup Database is undamaged.
   - `Database not OK` indicates that the Backup Database is damaged. To recover from the problem, use the instructions in "To repair corruption in the Backup Database" on page 288.

To repair corruption in the Backup Database

1. Log in as the local superuser **root** on each database server machine in the cell.
2. If the Tape Coordinator for the tape device that is to perform the operation is not already running, open a connection to the appropriate Tape Coordinator machine and issue the **butc** command, for which complete instructions appear in "To start a Tape Coordinator process" on page 247.

   ```
   % butc [<port offset>] [-noautoquery]
   ```

3. If writing to tape, place a tape in the appropriate device.
4. Working on one of the machines, issue the **backup** command to enter interactive mode.

   ```
   # backup -localauth
   ```

   where **-localauth** constructs a server ticket from the local **/usr/afs/etc/KeyFile** file. This flag enables you to issue a privileged command while logged in as the local superuser **root** but without AFS administrative tokens.

5. Verify that no backup operations are actively running. If necessary, issue the **(backup) status** command as described in "To check the status of a Tape Coordinator process" on page 249. Repeat for each Tape Coordinator port offset in turn.

   ```
   backup> status -portoffset <TC port offset>
   ```

6. Issue the **(backup) savedb** command to repair corruption in the database as it is written to tape or a file.

   ```
   backup> savedb [-portoffset <TC port offset>]
   ```

   where

   sa

   Is the shortest acceptable abbreviation of **savedb**.

   -portoffset

   Specifies the port offset number of the Tape Coordinator handling the tape or backup data file for this operation. You must provide this argument unless the default value of 0 (zero) is appropriate.

7. Exit interactive mode.

   ```
   backup>  quit
   ```

8. On each machine in turn, issue the **bos shutdown** command to shut down the Backup Server process. Include the **-localauth** flag because you are logged in as the local superuser root, but do not necessarily have administrative tokens. For complete command syntax, see "To stop processes temporarily" on page 122.

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

9. On each machine in turn, issue the following commands to remove the Backup Database.

   ```
   # cd /usr/afs/db
   # rm bdb.DB0
   # rm bdb.DBSYS1
   ```

10. On each machine in turn, starting with the machine with the lowest IP address, issue the **bos start** command to restart the Backup Server process, which creates a zero-length copy of the Backup Database as it starts. For complete command syntax, see "To start processes by changing their status flags to Run" on page 121.

    ```
    # /usr/afs/bin/bos start <machine name> buserver  -localauth
    ```

11. Working on one of the machines, issue the **backup** command to enter interactive mode.

    ```
    # backup -localauth
    ```

    where **-localauth** constructs a server ticket from the local **/usr/afs/etc/KeyFile** file.

12. Issue the **(backup) addhost** command to create an entry in the new, empty database for the Tape Coordinator process handling the tape or file from which you are reading the repaired copy of the database (presumably the process you started in Step "2" on page 289 and which performed the **backup savedb** operation in Step "6" on page 289). For complete syntax, see Step "8" on page 206 in "To configure a Tape Coordinator machine" on page 205.

    ```
    backup>  addhost <tape machine name> [<TC port offset>]
    ```

13. Issue the **(backup) restoredb** command to copy the repaired database to the database server machines.

    ```
    backup> restoredb  [-portoffset <TC port offset>]
    ```

    where

    res

    Is the shortest acceptable abbreviation of **restoredb**.

    -portoffset

    Specifies the port offset number of the Tape Coordinator handling the tape or backup data file for this operation. You must provide this argument unless the default value of **0** (zero) is appropriate.

14. **(Optional)** Exit interactive mode if you do not plan to issue any additional **backup** commands.

```
backup> quit
```

15. **(Optional)** If desired, enter **Ctrl-d** or another interrupt signal to exit the **root** shell on each database server machine. You can also issue the **Ctrl-c** signal on the Tape Coordinator machine to stop the process.

Removing Obsolete Records from the Backup Database

Whenever you recycle or relabel a tape using the **backup dump** or **backup labeltape** command, the Backup System automatically removes all of the dump records for the dumps contained on the tape and all other tapes in the dump set. However, obsolete records can still accumulate in the Backup Database over time. For example, when you discard a backup tape after using it the maximum number of times recommended by the manufacturer, the records for dumps on it remain in the database. Similarly, the Backup System does not automatically remove a dump's record when the dump reaches its expiration date, but only if you then recycle or relabel the tape that contains the dump. Finally, if a backup operation halts in the middle, the records for any volumes successfully written to tape before the halt remain in the database.

A very large Backup Database can make backup operations less efficient because the Backup Server has to navigate through a large number of records to find the ones it needs. To remove obsolete records, use the **backup deletedump** command. Either identify individual dumps by dump ID number, or specify the removal of all dumps created during a certain time period. Keep in mind that you cannot remove the record of an appended dump except by removing the record of its initial dump, which removes the records of all associated appended dumps. Removing records of a dump makes it impossible to restore data from the corresponding tapes or from any dump that refers to the deleted dump as its parent, directly or indirectly. That is, restore operations must begin with the full dump and continue with each incremental dump in order. If you have removed the records for a specific dump, you cannot restore any data from later incremental dumps.

Another way to truncate the Backup Database is to include the **-archive** argument to the **backup savedb** command. After a copy of the database is written to tape or to a backup data file, the Backup Server deletes the dump records for all dump operations with timestamps prior to the date and time you specify. However, issuing the **backup deletedump** command with only the **-to** argument is equivalent in effect and is simpler because it does not require starting a Tape Coordinator process as the **backup savedb** command does. For further information on the **-archive** argument to the **backup savedb** command, see the command's reference page in the *IBM AFS Administration Reference*.

If you later need to access deleted dump records, and the corresponding tapes still exist, you can use the **-dbadd** argument to the **backup scantape** command to scan their contents into the database, as instructed in "To scan the contents of a tape" on page 272.

To delete dump records from the Backup Database

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. **(Optional)** Issue the **backup** command to enter interactive mode, if you want to delete multiple records or issue additional commands. The interactive prompt appears in the following step.

   ```
   % backup
   ```

3. **(Optional)** Issue the **backup dumpinfo** command to list information from the Backup Database that can help you decide which records to delete. For detailed instructions, see "To display dump records" on page 266.

   ```
   backup> dumpinfo [<no. of dumps>]  [-id <dump id>]  [-verbose]
   ```

4. Issue the **backup deletedump** command to delete one or more dump sets.

   ```
   backup> deletedump [-dumpid <dumpid>+] [-from <date time>]  \
                      [-to <date time>]
   ```

   where

   dele

   Is the shortest acceptable abbreviation of **deletedump**.

   -dumpid

   Specifies the dump ID of each initial dump to delete from the Backup Database. The records for all associated appended dumps are also deleted. Provide either this argument or the **-to** (and optionally, **-from**) argument.

   -from

   Specifies the beginning of a range of dates; the record for any dump created during the indicated period of time is deleted.

   To omit all records before the time indicated with the **-to** argument, omit this argument. Otherwise provide a value in the following format

   mm/dd/yyyy [hh:MM]

   where the month (mm), day (dd), and year (yyyy) are required. You can omit the hour and minutes (hh:MM) to indicate the default of midnight (00:00 hours). If you provide them, use 24-hour format (for example, the value **14:36** represents 2:36 p.m.).

   You must provide the **-to** argument along with this one.

   Note: A plus sign follows this argument in the command's syntax statement because it accepts a multiword value which does not need to be enclosed in double quotes or other delimiters, not because it accepts multiple dates. Provide only one date (and optionally, time) definition.

   -to

   Specifies the end of a range of dates; the record of any dump created during the range is deleted from the Backup Database.

To delete all records created after the date you specify with the **-from** argument, specify the value **NOW**. To delete every dump record in the Backup Database, provide the value **NOW** and omit the **-from** argument. Otherwise, provide a date value in the same format as described for the **-from** argument. Valid values for the year (yyyy) range from **1970** to **2037**; higher values are not valid because the latest possible date in the standard UNIX representation is in early 2038. The command interpreter automatically reduces any later date to the maximum value in 2038.

If you omit the time portion (hh:MM), it defaults to 59 seconds after midnight (00:00:59 hours). Similarly, the **backup** command interpreter automatically adds 59 seconds to any time value you provide. In both cases, adding 59 seconds compensates for how the Backup Database and **backup dumpinfo** command represent dump creation times in hours and minutes only. For example, the Database records a creation timestamp of `20:55` for any dump operation that begins between 20:55:00 and 20:55:59. Automatically adding 59 seconds to a time thus includes the records for all dumps created during that minute.

Provide either this argument, or the **-dumpid** argument. This argument is required if the **-from** argument is provided.

> **Note:** A plus sign follows this argument in the command's syntax statement because it accepts a multiword value which does not need to be enclosed in double quotes or other delimiters, not because it accepts multiple dates. Provide only one date (and optionally, time) definition.

Chapter 8. Monitoring and Auditing AFS Performance

AFS comes with three main monitoring tools:

- The **scout** program, which monitors and gathers statistics on File Server performance.
- The **fstrace** command suite, which traces Cache Manager operations in detail.
- The **afsmonitor** program, which monitors and gathers statistics on both the File Server and the Cache Manager.

AFS also provides a tool for auditing AFS events on file server machines running AIX.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Initialize the **scout** program	**scout**
Display information about a trace log	**fstrace lslog**
Display information about an event set	**fstrace lsset**
Change the size of a trace log	**fstrace setlog**
Set the state of an event set	**fstrace setset**
Dump contents of a trace log	**fstrace dump**
Clear a trace log	**fstrace clear**
Initialize the **afsmonitor** program	**afsmonitor**

Using the scout Program

The **scout** program monitors the status of the File Server process running on file server machines. It periodically collects statistics from a specified set of File Server processes, displays them in a graphical format, and alerts you if any of the statistics exceed a configurable threshold.

More specifically, the **scout** program includes the following features.

- You can monitor, from a single location, the File Server process on any number of server machines from the local and foreign cells. The number is limited only by the size of the display window, which must be large enough to display the statistics.
- You can set a threshold for many of the statistics. When the value of a statistic exceeds the threshold, the **scout** program highlights it (displays it in reverse video) to draw your attention to it. If the value goes back under the threshold, the highlighting is deactivated. You control the thresholds, so highlighting reflects what you consider to be a noteworthy situation. See "Highlighting Significant Statistics" on page 299.
- The **scout** program alerts you to File Server process, machine, and network outages by highlighting the name of each machine that does not respond to its probe, enabling you to respond more quickly.

- You can set how often the **scout** program collects statistics from the File Server processes.

System Requirements

The **scout** program runs on any AFS client machine that has access to the **curses** graphics package, which most UNIX distributions include as a standard utility. It can run on both dumb terminals and under windowing systems that emulate terminals, but the output looks best on machines that support reverse video and cursor addressing. For best results, set the TERM environment variable to the correct terminal type, or one with characteristics similar to the actual ones. For machines running AIX, the recommended TERM setting is **vt100**, assuming the terminal is similar to that. For other operating systems, the wider range of acceptable values includes **xterm**, **xterms**, **vt100**, **vt200**, and **wyse85**.

No privilege is required to run the **scout** program, so any user who can access the directory where its binary resides (the **/usr/afsws/bin** directory in the conventional configuration) can use it. The program's probes for collecting statistics do not impose a significant burden on the File Server process, but you can restrict its use by placing the binary file in a directory with a more restrictive access control list (ACL).

Multiple instances of the **scout** program can run on a single client machine, each over its own dedicated connection (in its own window). It must run in the foreground, so the window in which it runs does not accept further input except for an interrupt signal.

You can also run the **scout** program on several machines and view its output on a single machine, by opening telnet connections to the other machines from the central one and initializing the program in each remote window. In this case, you can include the **-host** flag to the **scout** command to make the name of each remote machine appear in the *banner line* at the top of the window displaying its output. See "The Banner Line" on page 297.

Using the -basename argument to Specify a Domain Name

As previously mentioned, the **scout** program can monitor the File Server process on any number of file server machines. If all of the machines belong to the same cell, then their hostnames probably all have the same domain name suffix, such as **abc.com** in the ABC Corporation cell. In this case, you can use the **-basename** argument to the **scout** command, which has several advantages:

- You can omit the domain name suffix as you enter each file server machine's name on the command line. The **scout** program automatically appends the domain name to each machine's name, resulting in a fully-qualified hostname. You can omit the domain name suffix even when you don't include the **-basename** argument, but in that case correct resolution of the name depends on the state of your cell's naming service at the time of connection.
- The machine names are more likely to fit in the appropriate column of the display without having to be truncated (for more on truncating names in the display column, see "The Statistics Display Region" on page 297).
- The domain name appears in the banner line at the top of the display window to indicate the name of the cell you are monitoring.

The Layout of the scout Display

The **scout** program can display statistics either in a dedicated window or on a plain screen if a windowing environment is not available. For best results, use a window or screen that can print in reverse video and do cursor addressing.

The **scout** program screen has three main regions: the *banner line*, the *statistics display region* and the *probe/message* line. This section describes their contents, and graphic examples appear in "Example Commands and Displays" on page 302.

The Banner Line

By default, the string `scout` appears in the banner line at the top of the window or screen, to indicate that the **scout** program is running. You can display two additional types of information by include the appropriate option on the command line:

- Include the **-host** flag to display the local machine's name in the banner line. This is particularly useful when you are running the **scout** program on several machines but displaying the results on a single machine.

  For example, the following banner line appears when you run the **scout** program on the machine **client1.abc.com** and use the**-host** flag:

  ```
  [client1.abc.com] scout
  ```

- Include the **-basename** argument to display the specified cell domain name in the banner line. For further discussion, see "Using the -basename argument to Specify a Domain Name" on page 296.

  For example, if you specify a value of **abc.com** for the **-basename** argument, the banner line reads:

  ```
  scout for abc.com
  ```

The Statistics Display Region

The statistics display region occupies most of the window and is divided into six columns. The following list describes them as they appear from left to right in the window.

`Conn`

Displays the number of RPC connections open between the File Server process and client machines. This number normally equals or exceeds the number in the fourth `Ws` column. It can exceed the number in that column because each user on the machine can have more than one connection open at once, and one client machine can handle several users.

`Fetch`

Displays the number of fetch-type RPCs (fetch data, fetch access list, and fetch status) that the File Server process has received from client machines since it started. It resets to zero when the File Server process restarts.

`Store`

Displays the number of store-type RPCs (store data, store access list, and store status) that the File Server process has received from client machines since it started. It resets to zero when the File Server process restarts.

`Ws`

Displays the number of client machines (workstations) that have communicated with the File Server process within the last 15 minutes (such machines are termed *active*). This number is likely to be smaller than the number in the `Conn`) column because a single client machine can have several connections open to one File Server process.

[Unlabeled column]

Displays the name of the file server machine on which the File Server process is running. It is 12 characters wide. Longer names are truncated and an asterisk (`*`) appears as the last character in the name. If all machines have the same domain name suffix, you can use the **-basename** argument to decrease the need for truncation; see "Using the -basename argument to Specify a Domain Name" on page 296.

`Disk attn`

Displays the number of kilobyte blocks available on up to 26 of the file server machine's AFS server (**/vicep**) partitions. The display for each partition has the following format:

```
partition_letter:free_blocks
```

For example, `a:8949` indicates that partition **/vicepa** has 8,949 KB free. If the window is not wide enough for all partition entries to appear on a single line, the **scout** program automatically stacks the partition entries into subcolumns within the sixth column.

The label on the `Disk attn` column indicates the threshold value at which entries in the column become highlighted. By default, the **scout** program highlights a partition that is over 95% full, in which case the label is as follows:

```
Disk attn: > 95% used
```

For more on this threshold and its effect on highlighting, see "Highlighting Significant Statistics" on page 299.

For all columns except the fifth (file server machine name), you can use the **-attention** argument to set a threshold value above which the **scout** program highlights the statistic. By default, only values in the fifth and sixth columns ever become highlighted. For instructions on using the **-attention** argument, see "Highlighting Significant Statistics" on page 299.

The Probe Reporting Line

The bottom line of the display indicates how many times the **scout** program has probed the File Server processes for statistics. The statistics gathered in the latest probe appear in the statistics display region. By default, the **scout** program probes the File Servers every 60 seconds, but you can use the **-frequency** argument to specify a different probe frequency.

Highlighting Significant Statistics

To draw your attention to a statistic that currently exceed a threshold value, the **scout** program displays it in reverse video (highlights it). You can set the threshold value for most statistics, and so determine which values are worthy of special attention and which are normal.

Highlighting Server Outages

The only column in which you cannot control highlighting is the fifth, which identifies the file server machine for which statistics are displayed in the other columns. The **scout** program uses highlighting in this column to indicate that the File Server process on a machine fails to respond to its probe, and automatically blanks out the other columns. Failure to respond to the probe can indicate a File Server process, file server machine, or network outage, so the highlighting draws your attention to a situation that is probably interrupting service to users.

When the File Server process once again responds to the probes, its name appears normally and statistics reappear in the other columns. If all machine names become highlighted at once, a possible network outage has disrupted the connection between the file server machines and the client machine running the **scout** program.

Highlighting for Extreme Statistic Values

To set the threshold value for one or more of the five statistics-displaying columns, use the **-attention** argument. The threshold value applies to all File Server processes you are monitoring (you cannot set different thresholds for different machines). For details, see the syntax description in "To start the scout program" on page 300.

It is not possible to change the threshold values for a running **scout** program. Stop the current program and start a new one. Also, the **scout** program does not retain threshold values across restarts, so you must specify all thresholds every time you start the program.

Resizing the scout Display

Do not resize the display window while the **scout** program is running. Increasing the size does no harm, but the **scout** program does not necessarily adjust to the new dimensions. Decreasing the display's width can disturb column alignment, making the display harder to read. With any type of resizing, the **scout** program does not adjust the display in any way until it displays the results of the next probe.

To resize the display effectively, stop the **scout** program, resize the window and then restart the program. Even in this case, the **scout** program's response depends on the accuracy of the information it receives from the display environment. Testing during development has shown that the display environment does not reliably provide information about window resizing. If you use the X windowing system, issuing the following sequence of commands before starting the **scout** program (or placing them in the shell initialization file) sometimes makes it adjust properly to resizing.

```
% set noglob
% eval '/usr/bin/X11/resize'
% unset noglob
```

To start the scout program

1. Open a dedicated command shell. If necessary, adjust it to the appropriate size.
2. Issue the **scout** command to start the program.

```
% scout  [initcmd]  -server <FileServer name(s) to monitor>+  \
         [-basename <base server name>]  \
         [-frequency <poll frequency, in seconds>] [-host]  \
         [-attention <specify attention (highlighting) level>+]  \
         [-debug <turn debugging output on to the named file>]
```

where

initcmd

Is an optional string that accommodates the command's use of the AFS command parser. It can be omitted and ignored.

-server

Identifies each File Server process to monitor, by naming the file server machine it is running on. Provide fully-qualified hostnames unless the **-basename** argument is used. In that case, specify only the initial part of each machine name, omitting the domain name suffix common to all the machine names.

-basename

Specifies the domain name suffix common to all of the file server machines named by the **-server** argument. For discussion of this argument's effects, see "Using the -basename argument to Specify a Domain Name" on page 296.

Do not include the period that separates the domain suffix from the initial part of the machine name, but do include any periods that occur within the suffix itself. (For example, in the ABC Corporation cell, the proper value is **abc.com**, not **.abc.com**.)

-frequency

Sets the frequency, in seconds, of the **scout** program's probes to File Server processes. Specify an integer greater than 0 (zero). The default is 60 seconds.

-host

Displays the name of the machine that is running the **scout** program in the display window's banner line. By default, no machine name is displayed.

-attention

Defines the threshold value at which to highlight one or more statistics. You can provide the pairs of statistic and threshold in any order, separating each pair and the parts of each pair with one or more spaces. The following list defines the syntax for each statistic.

conn connections

Highlights the value in the `Conn` (first) column when the number of connections that the File Server has open to client machines exceeds the connections value. The highlighting deactivates when the value goes back below the threshold. There is no default threshold.

fetch fetch_RPCs

Highlights the value in the `Fetch` (second) column when the number of fetch RPCs that clients have made to the File Server process exceeds the fetch_RPCs value. The highlighting deactivates only when the File Server process restarts, at which time the value returns to zero. There is no default threshold.

store store_RPCs

Highlights the value in the `Store` (third) column when the number of store RPCs that clients have made to the File Server process exceeds the store_RPCs value. The highlighting deactivates only when the File Server process restarts, at which time the value returns to zero. There is no default threshold.

ws active_clients

Highlights the value in the `Ws` (fourth) column when the number of active client machines (those that have contacted the File Server in the last 15 minutes) exceeds the active_clients value. The highlighting deactivates when the value goes back below the threshold. There is no default threshold.

disk percent_full % or disk min_blocks

Highlights the value for a partition in the `Disk attn` (sixth) column when either the amount of disk space used exceeds the percentage indicated by thepercent_full value, or the number of free KB blocks is less than the min_blocks value. The highlighting deactivates when the value goes back below the percent_full threshold or above the min_blocks threshold.

The value you specify appears in the header of the sixth column following the string `Disk attn`. The default threshold is 95% full.

Acceptable values for percent_full are the integers from the range **0** (zero) to **99**, and you must include the percent sign to distinguish this statistic from a min_blocks value..

The following example sets the threshold for the `Conn` column to 100, for the `Ws` column to 50, and for the `Disk attn` column to 75%. There is no threshold for the `Fetch` and `Store` columns.

-attention conn 100 ws 50 disk 75%

The following example has the same affect as the previous one except that it sets the threshold for the Disk attn column to 5000 free KB blocks:

-attention disk 5000 ws 50 conn 100

-debug

Enables debugging output and directs it into the specified file. Partial pathnames are interpreted relative to the current working directory. By default, no debugging output is produced.

To stop the scout program

1. Enter **Ctrl-c** in the display window. This is the proper interrupt signal even if the general interrupt signal in your environment is different.

Example Commands and Displays

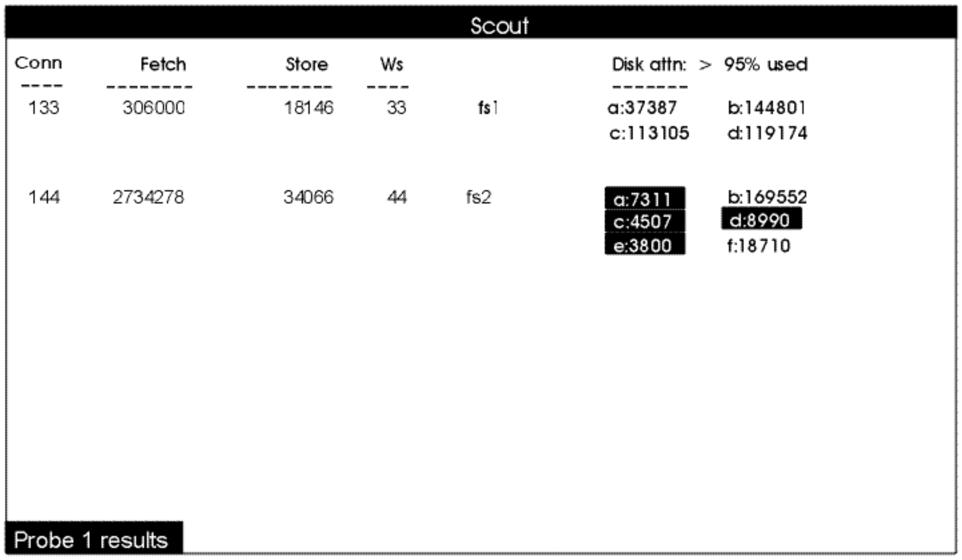
This section presents examples of the **scout** program, combining different arguments and illustrating the screen displays that result.

In the first example, an administrator in the ABC Corporation issues the **scout** command without providing any optional arguments or flags. She includes the **-server** argument because she is providing multiple machine names. She chooses to specify on the initial part of each machine's name even though she has not used the **-basename** argument, relying on the cell's name service to obtain the fully-qualified name that the **scout** program requires for establishing a connection.

```
% scout -server fs1 fs2
```

"Figure 2" on page 303 depicts the resulting display. Notice first that the machine names in the fifth (unlabeled) column appear in the format the administrator used on the command line. Now consider the second line in the display region, where the machine name `fs2` appears in the fifth column. The `Conn` and `Ws` columns together show that machine **fs2** has 144 RPC connections open to 44 client machines, demonstrating that multiple connections per client machine are possible. The `Fetch` column shows that client machines have made 2,734,278 fetch RPCs to machine **fs2** since the File Server process last started and the `Store` column shows that they have made 34,066 store RPCs.

Six partition entries appear in the `Disk attn` column, marked `a` through `f` (for **/vicepa** through **/vicepf**). They appear on three lines in two subcolumns because of the width of the window; if the window is wider, there are more subcolumns. Four of the partition entries (`a`, `c`, `d`, and `e`) appear in reverse video to indicate that they are more than 95% full (the threshold value that appears in the `Disk attn` header).



```
                                     Scout
Conn      Fetch        Store     Ws                Disk attn:  >  95% used
----    --------     --------   ----               -------
 133     306000        18146     33        fs1     a:37387      b:144801
                                                   c:113105     d:119174

 144     2734278       34066     44        fs2     a:7311       b:169552
                                                   c:4507       d:8990
                                                   e:3800       f:18710

Probe 1 results
```

Figure 2. First example scout display

In the second example, the administrator uses more of the **scout** program's optional arguments.

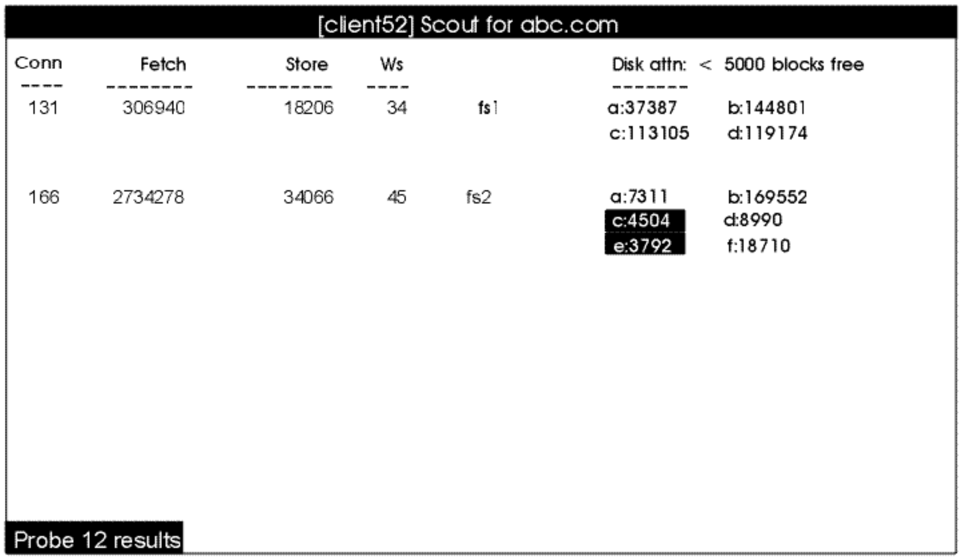
- She provides the machine names in the same form as in Example 1, but this time she also uses the **-basename** argument to specify their domain name suffix, **abc.com**. This implies that the **scout** program does not need the name service to expand the names to fully-qualified hostnames, but the name service still converts the hostnames to IP addresses.
- She uses the **-host** flag to display in the banner line the name of the client machine where the **scout** program is running.
- She uses the **-frequency** argument to changes the probing frequency from its default of once per minute to once every five seconds.
- She uses the **-attention** argument to changes the highlighting threshold for partitions to a 5000 KB minimum rather than the default of 95% full.

```
% scout -server fs1 fs2 -basename abc.com -host -frequency 5 -attention disk 5000
```

The use of optional arguments results in several differences between "Figure 3" on page 304 and "Figure 2" on page 303. First, because the **-host** flag is included, the banner line displays the name of the machine running the **scout** process as `[client52]` along with the basename `abc.com` specified with the **-basename** argument.

Another difference is that two rather than four of machine **fs2**'s partitions appear in reverse video, even though their values are almost the same as in "Figure 2" on page 303. This is because the administrator changed the highlight threshold to a 5000 block minimum, as also reflected in the `Disk attn` column's header. And while machine **fs2**'s partitions **/vicepa** and **/vicepd** are still 95% full, they have more than 5000 free blocks left; partitions **/vicepc** and **/vicepe** are highlighted because they have fewer than 5000 blocks free.

Note also the result of changing the probe frequency, reflected in the probe reporting line at the bottom left corner of the display. Both this example and the previous one represent a time lapse of one minute after the administrator issues the **scout** command. In this example, however, the **scout** program has probed the File Server processes 12 times as opposed to once

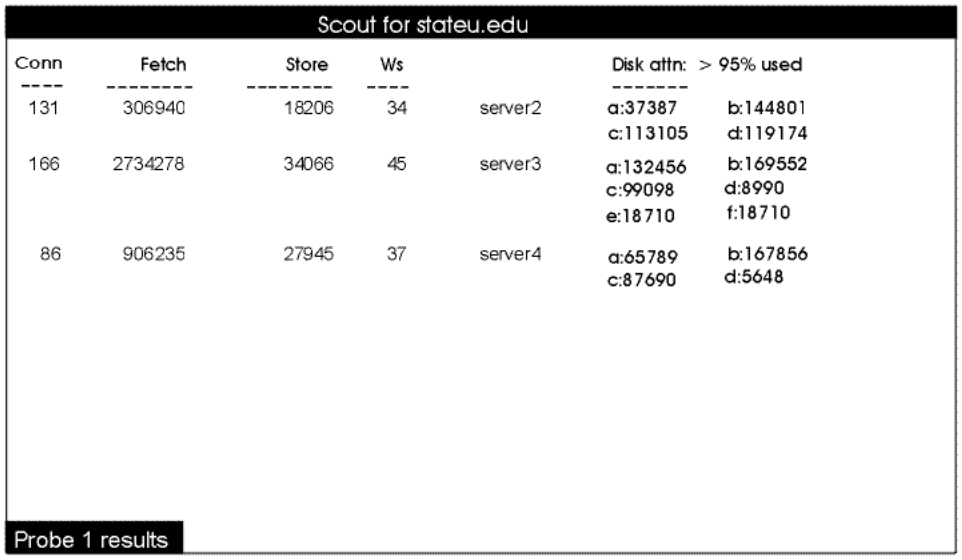


```
                         [client52] Scout for abc.com
Conn      Fetch       Store     Ws                 Disk attn:  <  5000 blocks free
----    --------    --------   ----                -------
 131     306940       18206     34       fs1       a:37387      b:144801
                                                   c:113105     d:119174

 166    2734278       34066     45       fs2       a:7311       b:169552
                                                   c:4504       d:8990
                                                   e:3792       f:18710







Probe 12 results
```

Figure 3. Second example scout display

In "Figure 4" on page 304, an administrator in the State University cell monitors three of that cell's file server machines. He uses the **-basename** argument to specify the **stateu.edu** domain name.

```
% scout -server server2 server3 server4 -basename stateu.edu
```



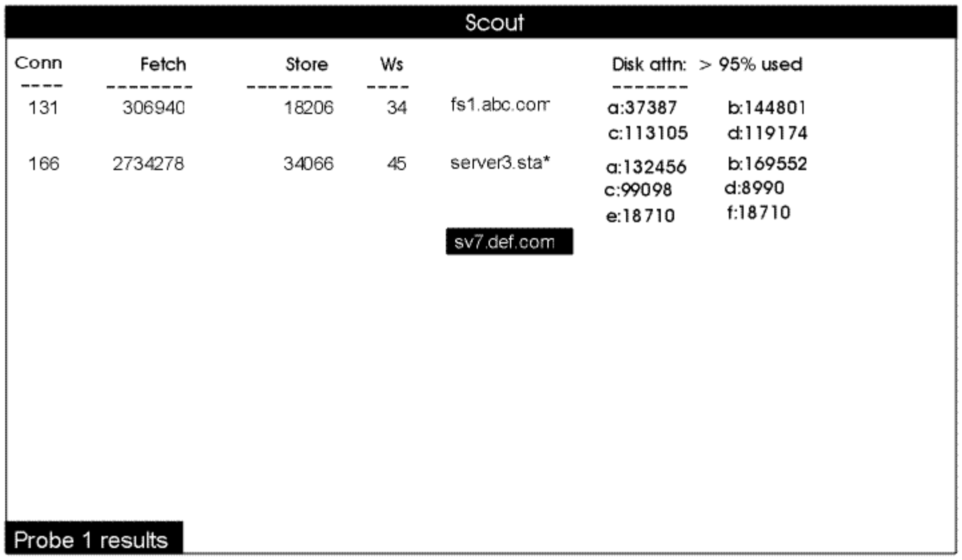
```
                              Scout for stateu.edu
Conn      Fetch       Store     Ws                    Disk attn:  > 95% used
----    --------    --------   ----                   -------
 131      306940       18206     34       server2     a:37387      b:144801
                                                      c:113105     d:119174
 166     2734278       34066     45       server3     a:132456     b:169552
                                                      c:99098      d:8990
                                                      e:18710      f:18710

  86      906235       27945     37       server4     a:65789      b:167856
                                                      c:87690      d:5648

Probe 1 results
```

Figure 4. Third example scout display

"Figure 5" on page 305 illustrates three of the **scout** program's features. First, you can monitor file server machines from different cells in a single display: **fs1.abc.com**, **server3.stateu.edu**, and **sv7.def.com**. Because the machines belong to different cells, it is not possible to provide the **-basename** argument.

Second, it illustrates how the display must truncate machine names that do not fit in the fifth column, using an asterisk at the end of the name to show that it is shortened.

Third, it illustrates what happens when the **scout** process cannot reach a File Server process, in this case the one on the machine **sv7.def.com**: it highlights the machine name and blanks out the values in the other columns.



```
                                     Scout
Conn      Fetch       Store     Ws                    Disk attn:  > 95% used
----    --------    --------   ----                   -------
 131      306940       18206     34   fs1.abc.com     a:37387      b:144801
                                                      c:113105     d:119174
 166     2734278       34066     45   server3.sta*    a:132456     b:169552
                                                      c:99098      d:8990
                                                      e:18710      f:18710
                                      sv7.def.com

Probe 1 results
```

Figure 5. Fourth example scout display

Using the fstrace Command Suite

This section describes the **fstrace** commands that system administrators employ to trace Cache Manager activity for debugging purposes. It assumes the reader is familiar with the Cache Manager concepts described in "Administering Client Machines and the Cache Manager" on page 351.

The **fstrace** command suite monitors the internal activity of the Cache Manager and enables you to record, or trace, its operations in detail. The operations, which are termed *events*, comprise the **cm** *event set*. Examples of **cm** events are fetching files and looking up information for a listing of files and subdirectories using the UNIX **ls** command.

Following are the **fstrace** commands and their respective functions:

- The **fstrace apropos** command provides a short description of commands.
- The **fstrace clear** command clears the trace log.
- The **fstrace dump** command dumps the contents of the trace log.
- The **fstrace help** command provides a description and syntax for commands.
- The **fstrace lslog** command lists information about the trace log.
- The **fstrace lsset** command lists information about the event set.
- The **fstrace setlog** command changes the size of the trace log.
- The **fstrace setset** command sets the state of the event set.

About the fstrace Command Suite

The **fstrace** command suite replaces and greatly expands the functionality formerly provided by the **fs debug** command. Its intended use is to aid in diagnosis of specific Cache Manager problems, such as client machine hangs, cache consistency problems, clock synchronization errors, and failures to access a volume or AFS file. Therefore, it is best not to keep **fstrace** logging enabled at all times, unlike the logging for AFS server processes.

Most of the messages in the trace log correspond to low-level Cache Manager operations. It is likely that only personnel familiar with the AFS source code can interpret them. If you have an AFS source license, you can attempt to interpret the trace yourself, or work with the AFS Product Support group to resolve the underlying problems. If you do not have an AFS source license, it is probably more efficient to contact the AFS Product Support group immediately in case of problems. They can instruct you to activate **fstrace** tracing if appropriate.

The log can grow in size very quickly; this can use valuable disk space if you are writing to a file in the local file space. Additionally, if the size of the log becomes too large, it can become difficult to parse the results for pertinent information.

When AFS tracing is enabled, each time a **cm** event occurs, a message is written to the trace log, **cmfx**. To diagnose a problem, read the output of the trace log and analyze the operations executed by the Cache Manager. The default size of the trace log is 60 KB, but you can increase or decrease it.

To use the **fstrace** command suite, you must first enable tracing and reserve, or allocate, space for the trace log with the **fstrace setset** command. With this command, you can set the **cm** event set to one of three states to enable or disable tracing for the event set and to allocate or deallocate space for the trace log in the kernel:

`active`

Enables tracing for the event set and allocates space for the trace log.

`inactive`

Temporarily disables tracing for the event set; however, the event set continues to allocate space occupied by the log to which it sends data.

`dormant`

Disables tracing for the event set; furthermore, the event set releases the space occupied by the log to which it sends data. When the **cm** event set that sends data to the **cmfx** trace log is in this state, the space allocated for that log is freed or deallocated.

Both event sets and trace logs can be designated as *persistent*, which prevents accidental resetting of an event set's state or clearing of a trace log. The designation is made as the kernel is compiled and cannot be changed.

If an event set such as **cm** is persistent, you can change its state only by including the **-set** argument to the **fstrace setset** command. (That is, you cannot change its state along with the state of all other event sets by issuing the **fstrace setset** command with no arguments.) Similarly, if a trace log such as **cmfx** is persistent, you can clear it only by including either the **-set** or **-log** argument to the **fstrace clear** command (you cannot clear it along with all other trace logs by issuing the **fstrace clear** command with no arguments.)

When a problem occurs, set the **cm** event set to active using the **fstrace setset** command. When tracing is enabled on a busy AFS client, the volume of events being recorded is significant; therefore, when you are diagnosing problems, restrict AFS activity as much as possible to minimize the amount of extraneous tracing in the log. Because tracing can have a negative impact on system performance, leave **cm** tracing in the dormant state when you are not diagnosing problems.

If a problem is reproducible, clear the **cmfx** trace log with the **fstrace clear** command and reproduce the problem. If the problem is not easily reproduced, keep the state of the event set active until the problem recurs.

To view the contents of the trace log and analyze the **cm** events, use the **fstrace dump** command to copy the content lines of the trace log to standard output (stdout) or to a file.

Note: If a particular command or process is causing problems, determine its process id (PID). Search the output of the **fstrace dump** command for the PID to find only those lines associated with the problem.

Requirements for Using the fstrace Command Suite

Except for the **fstrace help** and **fstrace apropos** commands, which require no privilege, issuing the **fstrace** commands requires that the issuer be logged in as the local superuser **root** on the local client machine. Before issuing an **fstrace** command, verify that you have the necessary privilege.

The Cache Manager catalog must be in place so that logging can occur. The **fstrace** command suite uses the standard UNIX catalog utilities. The default location is **/usr/vice/etc/C/afszcm.cat**. It can be placed in another directory by placing the file elsewhere and using the proper NLSPATH and LANG environment variables.

Using fstrace Commands Effectively

To use **fstrace** commands most effectively, configure them as indicated:

- Store the **fstrace** binary in a local disk directory.
- When you dump the **fstrace** log to a file, direct it to one on the local disk.
- The trace can grow large in just a few minutes. Before attempting to dump the log to a local file, verify that you have enough room. Be particularly careful if you are using disk quotas on partitions in the local file system.
- Attempt to limit Cache Manager activity on the AFS client machine other than the problem operation. This reduces the amount of extraneous data in the trace.
- Activate the **fstrace** log for the shortest possibly period of time. If possible activate the trace immediately before performing the problem operation, deactivate it as soon as the operation completes, and dump the trace log to a file immediately.
- If possible, obtain UNIX process ID (PID) of the command or program that initiates the problematic operation. This enables the person analyzing the trace log to search it for messages associated with the PID.

Activating the Trace Log

To start Cache Manager tracing on an AFS client machine, you must first configure

- The **cmfx** kernel trace log using the **fstrace setlog** command
- The **cm** event set using the **fstrace setset** command

The **fstrace setlog** command sets the size of the **cmfx** kernel trace log in kilobytes. The trace log occupies 60 kilobytes of kernel by default. If the trace log already exists, it is cleared when this command is issued and a new log of the given size is created. Otherwise, a new log of the desired size is created.

The **fstrace setset** command sets the state of the **cm** kernel event set. The state of the **cm** event set determines whether information on the events in that event set is logged.

After establishing kernel tracing on the AFS client machine, you can check the state of the event set and the size of the kernel buffer allocated for the trace log. To display information about the state of the **cm** event set, issue the **fstrace lsset** command. To display information about the **cmfx** trace log, use the **fstrace lslog** command. See the instructions in "Displaying the State of a Trace Log or Event Set" on page 309.

To configure the trace log

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fstrace setlog** command to set the size of the **cmfx** kernel trace log.

   ```
   # fstrace setlog  [-log <log_name>+]  -buffersize <1-kilobyte_units>
   ```

The following example sets the size of the **cmfx** trace log to 80 KB.

```
# fstrace setlog cmfx 80
```

To set the event set

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fstrace setset** command to set the state of event sets.

   ```
   % fstrace setset [-set <set_name>+] [-active] [-inactive]  \
                    [-dormant]
   ```

The following example activates the **cm** event set.

```
# fstrace setset cm -active
```

Displaying the State of a Trace Log or Event Set

An event set must be in the *active state* to be included in the trace log. To display an event set's state, use the **fstrace lsset** command. To set its state, issue the **fstrace setset** command as described in "To set the event set" on page 309.

To display size and allocation information for the trace log, issue the **fstrace lslog**command with the **-long** argument.

To display the state of an event set

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fstrace lsset** command to display the available event set and its state.

   ```
   # fstrace lsset  [-set <set_name>+]
   ```

The following example displays the event set and its state on the local machine.

```
# fstrace lsset cm
Available sets:
cm active
```

The output from this command lists the event set and its states. The three event states for the **cm** event set are:

active

Tracing is enabled.

inactive

Tracing is disabled, but space is still allocated for the corresponding trace log (**cmfx**).

dormant

Tracing is disabled, and space is no longer allocated for the corresponding trace log (**cmfx**).Disables tracing for the event set.

To display the log size

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fstrace lslog** command to display information about the kernel trace log.

   ```
   # fstrace lslog  [-set <set_name>+]  [-log <log_name>]  [-long]
   ```

The following example uses the **-long** flag to display additional information about the **cmfx** trace log.

```
# fstrace lslog cmfx -long
Available logs:
cmfx : 60 kbytes (allocated)
```

The output from this command lists information on the trace log. When issued without the **-long** flag, the **fstrace lslog** command lists only the name of the log. When issued with the **-long** flag, the **fstrace lslog** command lists the log, the size of the log in kilobytes, and the allocation state of the log.

There are two allocation states for the kernel trace log:

`allocated`

Space is reserved for the log in the kernel. This indicates that the event set that writes to this log is either *active* (tracing is enabled for the event set) or *inactive* (tracing is temporarily disabled for the event set); however, the event set continues to reserve space occupied by the log to which it sends data.

`unallocated`

Space is not reserved for the log in the kernel. This indicates that the event set that writes to this log is *dormant* (tracing is disabled for the event set); furthermore, the event set releases the space occupied by the log to which it sends data.

Dumping and Clearing the Trace Log

After the Cache Manager operation you want to trace is complete, use the **fstrace dump** command to dump the trace log to the standard output stream or to the file named by the **-file** argument. Or, to dump the trace log continuously, use the **-follow** argument (combine it with the **-file** argument if desired). To halt continuous dumping, press an interrupt signal such as **<Ctrl-c>**.

To clear a trace log when you no longer need the data in it, issue the **fstrace clear** command. (The **fstrace setlog** command also clears an existing trace log automatically when you use it to change the log's size.)

To dump the contents of a trace log

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fstrace dump** command to dump trace logs.

```
# fstrace dump [-set <set_name>+]  [-follow <log_name>]  \
               [-file <output_filename>]  \
               [-sleep <seconds_between_reads>]
```

At the beginning of the output of each dump is a header specifying the date and time at which the dump began. The number of logs being dumped is also displayed if the **-follow** argument is not specified. The header appears as follows:

```
AFS Trace Dump --
Date: date time
Found n logs.
```

where *date* is the starting date of the trace log dump, *time* is the starting time of the trace log dump, and *n* specifies the number of logs found by the **fstrace dump** command.

The following is an example of trace log dump header:

```
AFS Trace Dump --
Date: Fri Apr 16 10:44:38 1999
Found 1 logs.
```

The contents of the log follow the header and are comprised of messages written to the log from an active event set. The messages written to the log contain the following three components:

- The timestamp associated with the message (number of seconds from an arbitrary start point)
- The process ID or thread ID associated with the message
- The message itself

A trace log message is formatted as follows:

```
time timestamp, pid pid:event message
```

where *timestamp* is the number of seconds from an arbitrary start point, *pid* is the process ID number of the Cache Manager event, and *event message* is the Cache Manager event which corresponds with a function in the AFS source code.

The following is an example of a dumped trace log message:

```
time 749.641274, pid 3002:Returning code 2 from 19
```

For the messages in the trace log to be most readable, the Cache Manager catalog file needs to be installed on the local disk of the client machine; the conventional location is **/usr/vice/etc/C/afszcm.cat**. Log messages that begin with the string `raw op`, like the following, indicate that the catalog is not installed.

```
raw op 232c, time 511.916288, pid 0
p0:Fri Apr 16 10:36:31 1999
```

Every 1024 seconds, a current time message is written to each log. This message has the following format:

```
time timestamp, pid pid: Current time: unix_time
```

where timestamp is the number of seconds from an arbitrary start point, pid is the process ID number, and unix_time is the standard time format since January 1, 1970.

The current time message can be used to determine the actual time associated with each log message. Determine the actual time as follows:

1. Locate the log message whose actual time you want to determine.
2. Search backward through the dump record until you come to a current time message.

3. If the current time message's *timestamp* is smaller than the log message's *timestamp*, subtract the former from the latter. If the current time message's *timestamp* is larger than the log message's *timestamp*, add 1024 to the latter and subtract the former from the result.
4. Add the resulting number to the current time message's *unix_time* to determine the log message's actual time.

Because log data is stored in a finite, circular buffer, some of the data can be overwritten before being read. If this happens, the following message appears at the appropriate place in the dump:

```
Log wrapped; data missing.
```

> **Note:** If this message appears in the middle of a dump, which can happen under a heavy work load, it indicates that not all of the log data is being written to the log or some data is being overwritten. Increasing the size of the log with the **fstrace setlog** command can alleviate this problem.

To clear the contents of a trace log

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fstrace clear** command to clear logs by log name or by event set.

   ```
   # fstrace clear  [-set <set_name>+]  [-log <log_name>+]
   ```

The following example clears the **cmfx** log used by the **cm** event set on the local machine.

```
# fstrace clear cm
```

The following example also clears the **cmfx** log on the local machine.

```
# fstrace clear cmfx
```

Examples of fstrace Commands

This section contains an extensive example of the use of the **fstrace** command suite, which is useful for gathering a detailed trace of Cache Manager activity when you are working with AFS Product Support to diagnose a problem. The Product Support representative can guide you in choosing appropriate parameter settings for the trace.

Before starting the kernel trace log, try to isolate the Cache Manager on the AFS client machine that is experiencing the problem accessing the file. If necessary, instruct users to move to another machine so as to minimize the Cache Manager activity on this machine. To minimize the amount of unrelated AFS activity recorded in the trace log, place both the **fstrace** binary and the dump file must reside on the local disk, not in AFS. You must be logged in as the local superuser **root** to issue **fstrace** commands.

Before starting a kernel trace, issue the **fstrace lsset** command to check the state of the **cm** event set.

```
# fstrace lsset cm
```

If tracing has not been enabled previously or if tracing has been turned off on the client machine, the following output is displayed:

```
Available sets:
cm inactive
```

If tracing has been turned off and kernel memory is not allocated for the trace log on the client machine, the following output is displayed:

```
Available sets:
cm inactive (dormant)
```

If the current state of the **cm** event set is `inactive` or `inactive (dormant)`, turn on kernel tracing by issuing the **fstrace setset** command with the **-active** flag.

```
# fstrace setset cm -active
```

If tracing is enabled currently on the client machine, the following output is displayed:

```
Available sets:
cm active
```

If tracing is enabled currently, you do not need to use the **fstrace setset** command. Do issue the **fstrace clear** command to clear the contents of any existing trace log, removing prior traces that are not related to the current problem.

```
# fstrace clear cm
```

After checking on the state of the event set, issue the **fstrace lslog** command with the **-long** flag to check the current state and size of the kernel trace log .

```
# fstrace lslog cmfx -long
```

If tracing has not been enabled previously or the **cm** event set was set to `active` or `inactive` previously, output similar to the following is displayed:

```
Available logs:
cmfx : 60 kbytes (allocated)
```

The **fstrace** tracing utility allocates 60 kilobytes of memory to the trace log by default. You can increase or decrease the amount of memory allocated to the kernel trace log by setting it with the **fstrace setlog** command. The number specified with the **-buffersize** argument represents the number of kilobytes allocated to the kernel trace log. If you increase the size of the kernel trace log to 100 kilobytes, issue the following command.

```
# fstrace setlog cmfx 100
```

After ensuring that the kernel trace log is configured for your needs, you can set up a file into which you can dump the kernel trace log. For example, create a dump file with the name **cmfx.dump.file.1** using the following **fstrace dump** command. Issue the command as a continuous process by adding the **-follow** and **-sleep** arguments. Setting the **-sleep** argument to *10* dumps output from the kernel trace log to the file every 10 seconds.

```
# fstrace dump -follow cmfx -file cmfx.dump.file.1 -sleep 10
AFS Trace Dump -
   Date: Fri Apr 16 10:54:57 1999
Found 1 logs.
time 32.965783, pid 0: Fri Apr 16 10:45:52 1999
time 32.965783, pid 33657: Close 0x5c39ed8 flags 0x20
time 32.965897, pid 33657: Gn_close vp 0x5c39ed8 flags 0x20 (returns
0x0)
time 35.159854, pid 10891: Breaking callback for 5bd95e4 states 1024
(volume 0)
time 35.407081, pid 10891: Breaking callback for 5c0fadc states 1024
(volume 0)
    .                          .
    .                          .
    .                          .
time 71.440456, pid 33658: Lookup adp 0x5bbdcf0 name g3oCKs fid (756
4fb7e:588d240.2ff978a8.6)
time 71.440569, pid 33658: Returning code 2 from 19
time 71.440619, pid 33658: Gn_lookup vp 0x5bbdcf0 name g3oCKs (returns
0x2)
time 71.464989, pid 38267: Gn_open vp 0x5bbd000 flags 0x0 (returns 0x
0)
AFS Trace Dump - Completed
```

Using the afsmonitor Program

The **afsmonitor** program enables you to monitor the status and performance of specified File Server and Cache Manager processes by gathering statistical information. Among its other uses, the **afsmonitor** program can be used to fine-tune Cache Manager configuration and load balance File Servers.

The **afsmonitor** program enables you to perform the following tasks.

- Monitor any number of File Server and Cache Manager processes on any number of machines (in both local and foreign cells) from a single location.
- Set threshold values for any monitored statistic. When the value of a statistic exceeds the threshold, the **afsmonitor** program highlights it to draw your attention. You can set threshold levels that apply to every machine or only some.
- Invoke programs or scripts automatically when a statistic exceeds its threshold.

Requirements for running the afsmonitor program

The following software must be accessible to a machine where the **afsmonitor** program is running:

- The AFS **xstat** libraries, which the **afsmonitor** program uses to gather data
- The **curses** graphics package, which most UNIX distributions provide as a standard utility

The **afsmonitor** screens format successfully both on so-called dumb terminals and in windowing systems that emulate terminals. For the output to looks its best, the display environment needs to support reverse video and cursor addressing. Set the TERM environment variable to the correct terminal type, or to a value that has characteristics similar to the actual terminal type. The display window or terminal must be at least 80 columns wide and 12 lines long.

The **afsmonitor** program must run in the foreground, and in its own separate, dedicated window or terminal. The window or terminal is unavailable for any other activity as long as the **afsmonitor** program is running. Any number of instances of the **afsmonitor** program can run on a single machine, as long as each instance runs in its own dedicated window or terminal. Note that it can take up to three minutes to start an additional instance.

No privilege is required to run the **afsmonitor** program. By convention, it is installed in the **/usr/afsws/bin** directory, and anyone who can access the directory can monitor File Servers and Cache Managers. The probes through which the **afsmonitor** program collects statistics do not constitute a significant burden on the File Server or Cache Manager unless hundreds of people are running the program. If you wish to restrict its use, place the binary file in a directory available only to authorized users.

The afsmonitor Output Screens

The **afsmonitor** program displays its data on three screens:

- `System Overview`: This screen appears automatically when the **afsmonitor** program initializes. It summarizes separately for File Servers and Cache Managers the number of machines being monitored and how many of them have *alerts* (statistics that have exceeded their thresholds). It then lists the hostname and number of alerts for each machine being monitored, indicating if appropriate that a process failed to respond to the last probe.
- `File Server`: This screen displays File Server statistics for each file server machine being monitored. It highlights statistics that have exceeded their thresholds, and identifies machines that failed to respond to the last probe.
- `Cache Managers`: This screen displays Cache Manager statistics for each client machine being monitored. It highlights statistics that have exceeded their thresholds, and identifies machines that failed to respond to the last probe.

Fields at the corners of every screen display the following information:

- In the top left corner, the program name and version number.

- In the top right corner, the screen name, current and total page numbers, and current and total column numbers. The page number (for example, `p. 1 of 3`) indicates the index of the current page and the total number of (vertical) pages over which data is displayed. The column number (for example, `c. 1 of 235`) indicates the index of the current leftmost column and the total number of columns in which data appears. (The symbol >>> indicates that there is additional data to the right; the symbol <<< indicates that there is additional data to the left.)
- In the bottom left corner, a list of the available commands. Enter the first letter in the command name to run that command. Only the currently possible options appear; for example, if there is only one page of data, the `next` and `prev` commands, which scroll the screen up and down respectively, do not appear. For descriptions of the commands, see the following section about navigating the display screens.
- In the bottom right corner, the `probes` field reports how many times the program has probed File Servers (`fs`), Cache Managers (`cm`), or both. The counts for File Servers and Cache Managers can differ. The `freq` field reports how often the program sends probes.

Navigating the afsmonitor Display Screens

As noted, the lower left hand corner of every display screen displays the names of the commands currently available for moving to alternate screens, which can either be a different type or display more statistics or machines of the current type. To execute a command, press the lowercase version of the first letter in its name. Some commands also have an uppercase version that has a somewhat different effect, as indicated in the following list.

`cm`

Switches to the `Cache Managers` screen. Available only on the `System Overview` and `File Servers` screens.

`fs`

Switches to the `File Servers` screen. Available only on the `System Overview` and the `Cache Managers` screens.

`left`

Scrolls horizontally to the left, to access the data columns situated to the left of the current set. Available when the <<< symbol appears at the top left of the screen. Press uppercase **L** to scroll horizontally all the way to the left (to display the first set of data columns).

`next`

Scrolls down vertically to the next page of machine names. Available when there are two or more pages of machines and the final page is not currently displayed. Press uppercase **N** to scroll to the final page.

`oview`

Switches to the `System Overview` screen. Available only on the `Cache Managers` and `File Servers` screens.

`prev`

Scrolls up vertically to the previous page of machine names. Available when there are two or more pages of machines and the first page is not currently displayed. Press uppercase **N** to scroll to the first page.

`right`

Scrolls horizontally to the right, to access the data columns situated to the right of the current set. This command is available when the `>>>` symbol appears at the upper right of the screen. Press uppercase **R** to scroll horizontally all the way to the right (to display the final set of data columns).

The System Overview Screen

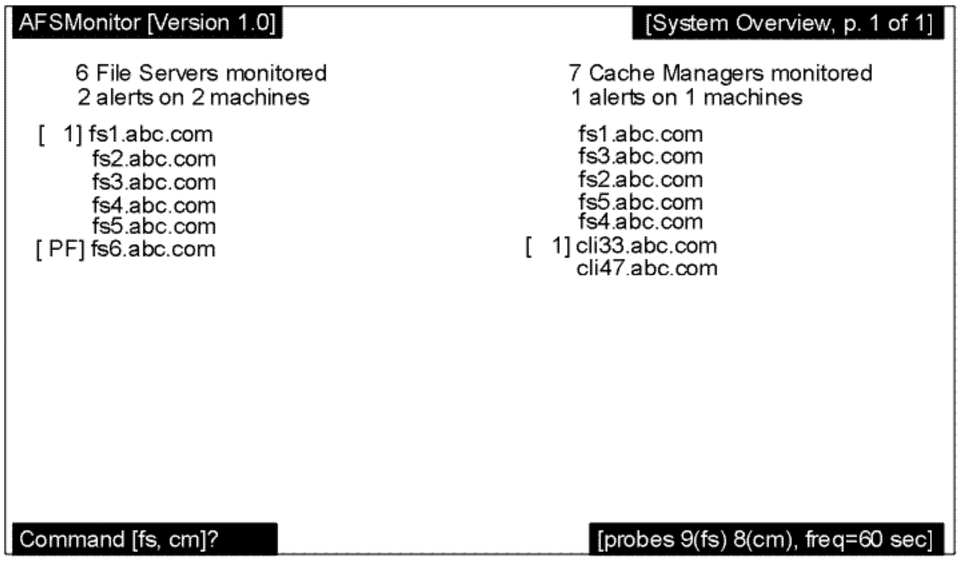
The `System Overview` screen appears automatically as the **afsmonitor** program initializes. This screen displays the status of as many File Server and Cache Manager processes as can fit in the current window; scroll down to access additional information.

The information on this screen is split into File Server information on the left and Cache Manager information on the right. The header for each grouping reports two pieces of information:

- The number of machines on which the program is monitoring the indicated process
- The number of alerts and the number of machines affected by them (an *alert* means that a statistic has exceeded its threshold or a process failed to respond to the last probe)

A list of the machines being monitored follows. If there are any alerts on a machine, the number of them appears in square brackets to the left of the hostname. If a process failed to respond to the last probe, the letters `PF` (probe failure) appear in square brackets to the left of the hostname.

The following graphic is an example `System Overview` screen. The **afsmonitor** program is monitoring six File Servers and seven Cache Managers. The File Server process on host **fs1.abc.com** and the Cache Manager on host **cli33.abc.com** are each marked `[ 1]` to indicate that one threshold value is exceeded. The `[PF]` marker on host **fs6.abc.com** indicates that its File Server process did not respond to the last probe.



```
AFSMonitor [Version 1.0]                                    [System Overview, p. 1 of 1]

      6 File Servers monitored                          7 Cache Managers monitored
      2 alerts on 2 machines                            1 alerts on 1 machines

  [   1] fs1.abc.com                                     fs1.abc.com
         fs2.abc.com                                     fs3.abc.com
         fs3.abc.com                                     fs2.abc.com
         fs4.abc.com                                     fs5.abc.com
         fs5.abc.com                                     fs4.abc.com
  [ PF] fs6.abc.com                                 [   1] cli33.abc.com
                                                         cli47.abc.com

Command [fs, cm]?                                  [probes 9(fs) 8(cm), freq=60 sec]
```

Figure 6. The afsmonitor System Overview Screen

The File Servers Screen

The `File Servers` screen displays the values collected at the most recent probe for File Server statistics.

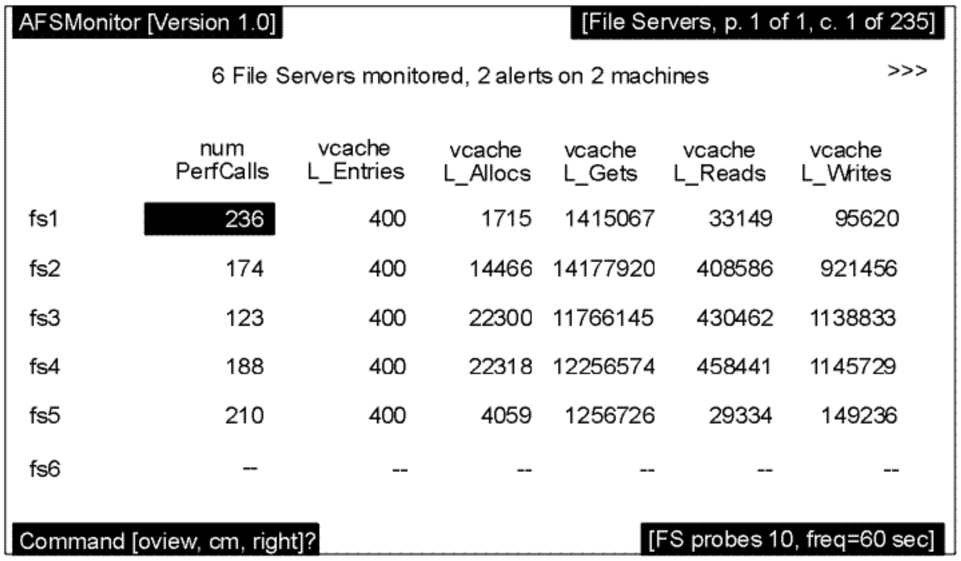
A summary line at the top of the screen (just below the standard program version and screen title blocks) specifies the number of monitored File Servers, the number of alerts, and the number of machines affected by the alerts.

The first column always displays the hostnames of the machines running the monitored File Servers.

To the right of the hostname column appear as many columns of statistics as can fit within the current width of the display screen or window; each column requires space for 10 characters. The name of the statistic appears at the top of each column. If the File Server on a machine did not respond to the most recent probe, a pair of dashes (`--`) appears in each column. If a value exceeds its configured threshold, it is highlighted in reverse video. If a value is too large to fit into the allotted column width, it overflows into the next row in the same column.

For a list of the available File Server statistics, see "Appendix C, The afsmonitor Program Statistics" on page 563.

The following graphic depicts the `File Servers` screen that follows the System Overview Screen example previously discussed; however, one additional server probe has been completed. In this example, the File Server process on **fs1** has exceeded the configured threshold for the number of performance calls received (the **numPerfCalls** statistic), and that field appears in reverse video. Host **fs6** did not respond to Probe 10, so dashes appear in all fields.

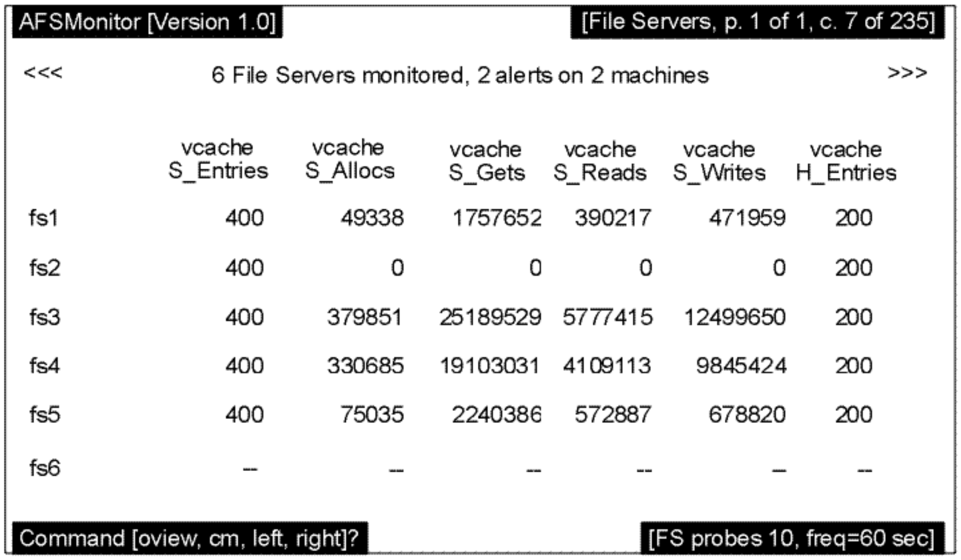


```
AFSMonitor [Version 1.0]                        [File Servers, p. 1 of 1, c. 1 of 235]

                 6 File Servers monitored, 2 alerts on 2 machines                  >>>

                num      vcache     vcache     vcache     vcache     vcache
             PerfCalls  L_Entries  L_Allocs    L_Gets    L_Reads   L_Writes
fs1               236        400      1715    1415067     33149      95620
fs2               174        400     14466   14177920    408586     921456
fs3               123        400     22300   11766145    430462    1138833
fs4               188        400     22318   12256574    458441    1145729
fs5               210        400      4059    1256726     29334     149236
fs6                --         --        --         --        --         --

Command [oview, cm, right]?                           [FS probes 10, freq=60 sec]
```

Figure 7. The afsmonitor File Servers Screen

Both the File Servers and Cache Managers screen (discussed in the following section) can display hundreds of columns of data and are therefore designed to scroll left and right. In the preceding graphic, the screen displays the leftmost screen and the screen title block shows that column 1 of 235 is displayed. The appearance of the >>> symbol in the upper right hand corner of the screen and the **right** command in the command block indicate that additional data is available by scrolling right. (For information on the available statistics, see "Appendix C, The afsmonitor Program Statistics" on page 563.)

If the **right** command is executed, the screen looks something like the following example. Note that the horizontal scroll symbols now point both to the left (<<<) and to the right (>>>) and both the **left** and **right** commands appear, indicating that additional data is available by scrolling both left and right.



```
AFSMonitor [Version 1.0]                        [File Servers, p. 1 of 1, c. 7 of 235]

<<<              6 File Servers monitored, 2 alerts on 2 machines                  >>>

               vcache     vcache      vcache     vcache     vcache     vcache
             S_Entries   S_Allocs     S_Gets    S_Reads   S_Writes  H_Entries
fs1               400      49338     1757652    390217     471959       200
fs2               400          0           0         0          0       200
fs3               400     379851    25189529   5777415   12499650       200
fs4               400     330685    19103031   4109113    9845424       200
fs5               400      75035     2240386    572887     678820       200
fs6                --         --          --        --         --        --

Command [oview, cm, left, right]?                     [FS probes 10, freq=60 sec]
```

Figure 8. The afsmonitor File Servers Screen Shifted One Page to the Right

The Cache Managers Screen

The `Cache Managers` screen displays the values collected at the most recent probe for Cache Manager statistics.

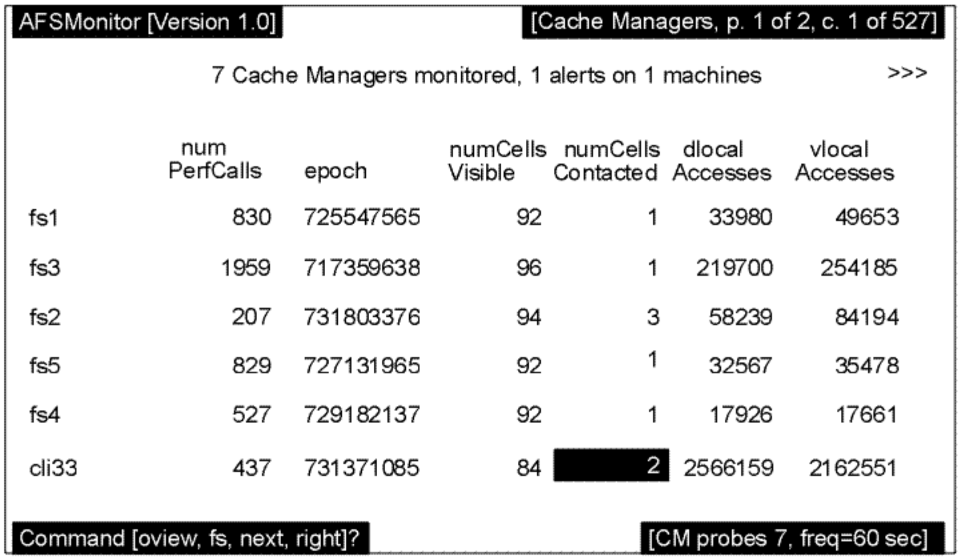
A summary line at the top of the screen (just below the standard program version and screen title blocks) specifies the number of monitored Cache Managers, the number of alerts, and the number of machines affected by the alerts.

The first column always displays the hostnames of the machines running the monitored Cache Managers.

To the right of the hostname column appear as many columns of statistics as can fit within the current width of the display screen or window; each column requires space for 10 characters. The name of the statistic appears at the top of each column. If the Cache Manager on a machine did not respond to the most recent probe, a pair of dashes (--) appears in each column. If a value exceeds its configured threshold, it is highlighted in reverse video. If a value is too large to fit into the allotted column width, it overflows into the next row in the same column.

For a list of the available Cache Manager statistics, see "Appendix C, The afsmonitor Program Statistics" on page 563.

The following graphic depicts a Cache Managers screen that follows the System Overview Screen previously discussed. In the example, the Cache Manager process on host **cli33** has exceeded the configured threshold for the number of cells it can contact (the **numCellsContacted** statistic), so that field appears in reverse video.



```
AFSMonitor [Version 1.0]                    [Cache Managers, p. 1 of 2, c. 1 of 527]

                 7 Cache Managers monitored, 1 alerts on 1 machines              >>>

              num                     numCells  numCells   dlocal     vlocal
              PerfCalls   epoch       Visible   Contacted  Accesses   Accesses

fs1             830     725547565        92         1      33980      49653

fs3            1959     717359638        96         1     219700     254185

fs2             207     731803376        94         3      58239      84194

fs5             829     727131965        92         1      32567      35478

fs4             527     729182137        92         1      17926      17661

cli33           437     731371085        84         2    2566159    2162551

Command [oview, fs, next, right]?                          [CM probes 7, freq=60 sec]
```

Figure 9. The afsmonitor Cache Managers Screen

Configuring the afsmonitor Program

To customize the **afsmonitor** program, create an ASCII-format configuration file and use the **-config** argument to name it. You can specify the following in the configuration file:

- The File Servers, Cache Managers, or both to monitor.
- The statistics to display. By default, the display includes 271 statistics for File Servers and 570 statistics for Cache Managers. For information on the available statistics, see "Appendix C, The afsmonitor Program Statistics" on page 563.
- The threshold values to set for statistics and a script or program to execute if a threshold is exceeded. By default, no threshold values are defined and no scripts or programs are executed.

The following list describes the instructions that can appear in the configuration file:

`cm` *`hostname`*

Names a client machine for which to display Cache Manager statistics. The order of **cm** lines in the file determines the order in which client machines appear from top to bottom on the `System Overview` and `Cache Managers` output screens.

`fs` *`hostname`*

Names a file server machine for which to display File Server statistics. The order of **fs** lines in the file determines the order in which file server machines appear from top to bottom on the `System Overview` and `File Servers` output screens.

`thresh fs | cm` *`field_name thresh_val`* `[`*`cmd_to_run`*`] [`*`arg1`*`] . . . [`*`argn`*`]`

Assigns the threshold value thresh_val to the statistic field_name, for either a File Server statistic (**fs**) or a Cache Manager statistic (**cm**). The optional cmd_to_execute field names a binary or script to execute each time the value of the statistic changes from being below thresh_val to being at or above thresh_val. A change between two values that both exceed thresh_val does not retrigger the binary or script. The optional arg1 through argn fields are additional values that the **afsmonitor** program passes as arguments to the cmd_to_execute command. If any of them include one or more spaces, enclose the entire field in double quotes.

The parameters **fs**, **cm**, field_name, threshold_val, and arg1 through argn correspond to the values with the same name on the **thresh** line. The host_name parameter identifies the file server or client machine where the statistic has crossed the threshold, and the actual_val parameter is the actual value of field_name that equals or exceeds the threshold value.

Use the **thresh** line to set either a global threshold, which applies to all file server machines listed on **fs** lines or client machines listed on **cm** lines in the configuration file, or a machine-specific threshold, which applies to only one file server or client machine.

- To set a global threshold, place the **thresh** line before any of the **fs** or **cm** lines in the file.
- To set a machine-specific threshold, place the **thresh** line below the corresponding **fs** or **cm** line, and above any other **fs** or **cm** lines. A machine-specific threshold value always overrides the corresponding global threshold, if set. Do not place a **thresh fs** line directly after a **cm** line or a **thresh cm** line directly after a **fs** line.

`show fs | cm` *`field/group/section`*

Specifies which individual statistic, group of statistics, or section of statistics to display on the `File Servers` screen (**fs**) or `Cache Managers` screen (**cm**) and the order in which to display them. The appendix of **afsmonitor** statistics in the *IBM AFS Administration Guide* specifies the group and section to which each statistic belongs. Include as many **show** lines as necessary to customize the screen display as desired, and place them anywhere in the file. The top-to-bottom order of the **show** lines in the configuration file determines the left-to-right order in which the statistics appear on the corresponding screen.

If there are no **show** lines in the configuration file, then the screens display all statistics for both Cache Managers and File Servers. Similarly, if there are no **show fs** lines, the `File Servers` screen displays all file server statistics, and if there are no **show cm** lines, the `Cache Managers` screen displays all client statistics.

# comments

Precedes a line of text that the **afsmonitor** program ignores because of the initial number (**#**) sign, which must appear in the very first column of the line.

For a list of the values that can appear in the field/group/section field of a **show** instruction, see "Appendix C, The afsmonitor Program Statistics" on page 563.)

The following example illustrates a possible configuration file:

```
thresh cm dlocalAccesses  1000000
thresh cm dremoteAccesses  500000 handleDRemote
thresh fs rx_maxRtt_Usec     1000
cm client5
cm client33
cm client14
thresh cm dlocalAccesses  2000000
thresh cm vcacheMisses      10000
cm client2
fs fs3
fs fs9
fs fs5
fs fs10
show cm numCellsContacted
show cm dlocalAccesses
show cm dremoteAccesses
show cm vcacheMisses
show cm Auth_Stats_group
```

Since the first three **thresh** instructions appear before any **fs** or **cm** instructions, they set global threshold values:

- All Cache Manager process in this file use **1000000** as the threshold for the **dlocalAccesses** statistic (except for the machine **client2** which uses an overriding value of **2000000**.)
- All Cache Manager processes in this file use **500000** as the threshold value for the **dremoteAccesses** statistic; if that value is exceeded, the script **handleDRemote** is invoked.
- All File Server processes in this file use **1000** as the threshold value for the **rx_maxRtt_Usec** statistic.

The four **cm** instructions monitor the Cache Manager on the machines **client5**, **client33**, **client14**, and **client2**. The first three use all of the global threshold values.

The Cache Manager on **client2** uses the global threshold value for the **dremoteAccesses** statistic, but a different one for the **dlocalAccesses** statistic. Furthermore, **client22** is the only Cache Manager that uses the threshold set for the **vcacheMisses** statistic.

The **fs** instructions monitor the File Server on the machines **fs3**, **fs9**, **fs5**, and **fs10**. They all use the global threshold for the**rx_maxRtt_Usec** statistic.

Because there are no **show fs** instructions, the File Servers screen displays all File Server statistics. The Cache Managers screen displays only the statistics named in **show cm** instructions, ordering them from left to right. The **Auth_Stats_group** includes several statistics, all of which are displayed (**curr_PAGs**, **curr_Records**, **curr_AuthRecords**, **curr_UnauthRecords**, **curr_MaxRecordsInPAG**, **curr_LongestChain**, **PAGCreations**, **TicketUpdates**, **HWM_PAGS**, **HWM_Records**, **HWM_MaxRecordsInPAG**, and **HWM_LongestChain**).

Writing afsmonitor Statistics to a File

All of the statistical information collected and displayed by the **afsmonitor** program can be preserved by writing it to an output file. You can create an output file by using the **-output** argument when you startup the **afsmonitor** process. You can use the output file to track process performance over long periods of time and to apply post-processing techniques to further analyze system trends.

The **afsmonitor** program output file is a simple ASCII file that records the information reported by the File Server and Cache Manager screens. The output file has the following format:

```
time   host_name CM|FS   list_of_measured_values
```

and specifies the *time* at which the *list_of_measured_values* were gathered from the Cache Manager (**CM**) or File Server (**FS**) process housed on host_name. On those occasion where probes fail, the value `-1` is reported instead of the *list_of_measured_values*.

This file format provides several advantages:

- It can be viewed using a standard editor. If you intend to view this file frequently, use the **-detailed** flag with the **-output** argument. It formats the output file in a way that is easier to read.
- It can be passed through filters to extract desired information using the standard set of UNIX tools.
- It is suitable for long term storage of the **afsmonitor** program output.

To start the afsmonitor Program

1. Open a separate command shell window or use a dedicated terminal for each instance of the **afsmonitor** program. This window or terminal must be devoted to the exclusive use of the **afsmonitor** process because the command cannot be run in the background.
2. Initialize the **afsmonitor** program. The message `afsmonitor Collecting Statistics...`, followed by the appearance of the `System Overview` screen, confirms a successful start.

```
% afsmonitor [initcmd]  [-config <configuration file>]  \
             [-frequency <poll frequency, in seconds>]  \
             [-output <storage file name>] [-detailed]  \
             [-debug <turn debugging output on to the named file>] \
             [-fshosts <list of file servers to monitor>+]  \
             [-cmhosts <list of cache managers to monitor>+]
afsmonitor Collecting Statistics...
```

where

initcmd

Is an optional string that accommodates the command's use of the AFS command parser. It can be omitted and ignored.

-config

Specifies the pathname of an **afsmonitor** configuration file, which lists the machines and statistics to monitor. Partial pathnames are interpreted relative to the current working directory. Provide either this argument or one or both of the **-fshosts** and **-cmhosts** arguments. You must use a configuration file to set thresholds or customize the screen display. For instructions on creating the configuration file, see "Configuring the afsmonitor Program" on page 321.

-frequency

Specifies how often to probe the File Server and Cache Manager processes, as a number of seconds. Acceptable values range from **1** and **86400**; the default value is **60**. This frequency applies to both File Server and Cache Manager probes; however, File Server and Cache Manager probes are initiated and processed independent of each other. The actual interval between probes to a host is the probe frequency plus the time needed by all hosts to respond to the probe.

-output

Specifies the name of an output file to which to write all of the statistical data. By default, no output file is created. For information on this file, see "Writing afsmonitor Statistics to a File" on page 324.

-detailed

Formats the output file named by the **-output** argument to be more easily readable. The **-output** argument must be provided along with this flag.

-fshosts

Identifies each File Server process to monitor by specifying the host it is running on. You can identify a host using either its complete Internet-style host name or an abbreviation acceptable to the cell's naming service. Combine this argument with the **-cmhosts** if you wish, but not the **-config** argument.

-cmhosts

Identifies each Cache Manager process to monitor by specifying the host it is running on. You can identify a host using either its complete Internet-style host name or an abbreviation acceptable to the cell's naming service. Combine this argument with the **-fshosts** if you wish, but not the **-config** argument.

To stop the afsmonitor program

To exit an **afsmonitor** program session, Enter the **<Ctrl-c>** interrupt signal or an uppercase **Q**.

The xstat Data Collection Facility

The **afsmonitor** program uses the **xstat** data collection facility to gather and calculate the data that it (the **afsmonitor** program) then uses to perform its function. You can also use the **xstat** facility to create your own data display programs. If you do, keep the following in mind. The File Server considers any program calling its RPC routines to be a Cache Manager; therefore, any program calling the File Server interface directly must export the Cache Manager's callback interface. The calling program must be capable of emulating the necessary callback state, and it must respond to periodic keep-alive messages from the File Server. In addition, a calling program must be able to gather the collected data.

The **xstat** facility consists of two C language libraries available to user-level applications:

- **/usr/afsws/lib/afs/libxstat_fs.a** exports calls that gather information from one or more running File Server processes.
- **/usr/afsws/lib/afs/libxstat_cm.a** exports calls that collect information from one or more running Cache Managers.

The libraries allow the caller to register

- A set of File Servers or Cache Managers to be examined.
- The frequency with which the File Servers or Cache Managers are to be probed for data.
- A user-specified routine to be called each time data is collected.

The libraries handle all of the lightweight processes, callback interactions, and timing issues associated with the data collection. The user needs only to process the data as it arrives.

The libxstat Libraries

The **libxstat_fs.a** and **libxstat_cm.a** libraries handle the callback requirements and other complications associated with the collection of data from File Servers and Cache Managers. The user provides only the means of accumulating the desired data. Each **xstat** library implements three routines:

- Initialization (**xstat_fs_Init** and **xstat_cm_Init**) arranges the periodic collection and handling of data.
- Immediate probe (**xstat_fs_ForceProbeNow** and **xstat_cm_ForceProbeNow**) forces the immediate collection of data, after which collection returns to its normal probe schedule.
- Cleanup (**xstat_fs_Cleanup** and **xstat_cm_Cleanup**) terminates all connections and removes all traces of the data collection from memory.

The File Server and Cache Manager each define data collections that clients can fetch. A data collection is simply a related set of numbers that can be collected as a unit. For example, the File Server and Cache Manager each define profiling and performance data collections. The profiling collections maintain counts of the number of times internal functions are called within servers, allowing bottleneck analysis to be performed. The performance collections record, among other things, internal disk I/O statistics for a File Server and cache effectiveness figures for a Cache Manager, allowing for performance analysis.

For a copy of the detailed specification which provides much additional usage information about the **xstat** facility, its libraries, and the routines in the libraries, contact AFS Product Support.

Example xstat Commands

AFS comes with two low-level, example commands: **xstat_fs_test** and **xstat_cm_test**. The commands allow you to experiment with the **xstat** facility. They gather information and display the available data collections for a File Server or Cache Manager. They are intended merely to provide examples of the types of data that can be collected via **xstat**; they are not intended for use in the actual collection of data.

To use the example xstat_fs_test command

1. Issue the example **xstat_fs_test** command to test the routines in the **libxstat_fs.a** library and display the data collections associated with the File Server process. The command executes in the foreground.

   ```
   % xstat_fs_test [initcmd]  \
                   -fsname <File Server name(s) to monitor>+  \
                   -collID <Collection(s) to fetch>+  [-onceonly]  \
                   [-frequency <poll frequency, in seconds>]  \
                   [-period <data collection time, in minutes>] [-debug]
   ```

   where

   xstat_fs_test

   Must be typed in full.

initcmd

Is an optional string that accommodates the command's use of the AFS command parser. It can be omitted and ignored.

-fsname

Is the Internet host name of each file server machine on which to monitor the File Server process.

-collID

Specifies each data collection to return. The indicated data collection defines the type and amount of data the command is to gather about the File Server. Data is returned in the form of a predefined data structure (refer to the specification documents referenced previously for more information about the data structures).

There are two acceptable values:

- **1** reports various internal performance statistics related to the File Server (for example, vnode cache entries and **Rx** protocol activity).
- **2** reports all of the internal performance statistics provided by the **1** setting, plus some additional, detailed performance figures about the File Server (for example, minimum, maximum, and cumulative statistics regarding File Server RPCs, how long they take to complete, and how many succeed).

-onceonly

Directs the command to gather statistics just one time. Omit this option to have the command continue to probe the File Server for statistics every 30 seconds. If you omit this option, you can use the **<Ctrl-c>** interrupt signal to halt the command at any time.

-frequency

Sets the frequency in seconds at which the program initiates probes to the File Server. If you omit this argument, the default is 30 seconds.

-period

Sets how long the utility runs before exiting, as a number of minutes. If you omit this argument, the default is 10 minutes.

-debug

Displays additional information as the command runs.

To use the example xstat_cm_test command

1. Issue the example **xstat_cm_test** command to test the routines in the **libxstat_cm.a** library and display the data collections associated with the Cache Manager. The command executes in the foreground.

   ```
   % xstat_cm_test [initcmd]  \
                   -cmname <Cache Manager name(s) to monitor>+  \
                   -collID <Collection(s) to fetch>+ \
                   [-onceonly] [-frequency <poll frequency, in seconds>]  \
                   [-period <data collection time, in minutes>] [-debug]
   ```

   where

   xstat_cm_test

   Must be typed in full.

   initcmd

   Is an optional string that accommodates the command's use of the AFS command parser. It can be omitted and ignored.

   -cmname

   Is the host name of each client machine on which to monitor the Cache Manager.

   -collID

   Specifies each data collection to return. The indicated data collection defines the type and amount of data the command is to gather about the Cache Manager. Data is returned in the form of a predefined data structure (refer to the specification documents referenced previously for more information about the data structures).

   There are two acceptable values:

   - **0** provides profiling information about the numbers of times different internal Cache Manager routines were called since the Cache manager was started.
   - **1** reports various internal performance statistics related to the Cache manager (for example, statistics about how effectively the cache is being used and the quantity of intracell and intercell data access).
   - **2** reports all of the internal performance statistics provided by the **1** setting, plus some additional, detailed performance figures about the Cache Manager (for example, statistics about the number of RPCs sent by the Cache Manager and how long they take to complete; and statistics regarding things such as authentication, access, and PAG information associated with data access).

-onceonly

Directs the command to gather statistics just one time. Omit this option to have the command continue to probe the Cache Manager for statistics every 30 seconds. If you omit this option, you can use the **<Ctrl-c>** interrupt signal to halt the command at any time.

-frequency

Sets the frequency in seconds at which the program initiates probes to the Cache Manager. If you omit this argument, the default is 30 seconds.

-period

Sets how long the utility runs before exiting, as a number of minutes. If you omit this argument, the default is 10 minutes.

-debug

Displays additional information as the command runs.

Auditing AFS Events on AIX File Servers

You can audit AFS events on AIX File Servers using an AFS mechanism that transfers audit information from AFS to the AIX auditing system. The following general classes of AFS events can be audited. For a complete list of specific AFS audit events, see "Appendix D, AIX Audit Events" on page 591.

- Authentication and Identification Events
- Security Events
- Privilege Required Events
- Object Creation and Deletion Events
- Attribute Modification Events
- Process Control Events

Note: This section assumes familiarity with the AIX auditing system. For more information, see the *AIX System Management Guide* for the version of AIX you are using.

Configuring AFS Auditing on AIX File Servers

The directory **/usr/afs/local/audit** contains three files that contain the information needed to configure AIX File Servers to audit AFS events:

- The **events.sample** file contains information on auditable AFS events. The contents of this file are integrated into the corresponding AIX events file (**/etc/security/audit/events**).
- The **config.sample** file defines the six classes of AFS audit events and the events that make up each class. It also defines the classes of AFS audit events to audit for the File Server, which runs as the local superuser **root**. The contents of this file must be integrated into the corresponding AIX config file (**/etc/security/audit/config**).
- The **objects.sample** file contains a list of information about audited files. You must only audit files in the local file space. The contents of this file must be integrated into the corresponding AIX objects file (**/etc/security/audit/objects**).

Once you have properly configured these files to include the AFS-relevant information, use the AIX auditing system to start up and shut down the auditing.

To enable AFS auditing

1. Create the following string in the file **/usr/afs/local/Audit** on each File Server on which you plan to audit AFS events:

   ```
   AFS_AUDIT_AllEvents
   ```

2. Issue the **bos restart** command (with the **-all** flag) to stop and restart all server processes on each File Server. For instructions on using this command, see "Stopping and Immediately Restarting Processes" on page 124.

To disable AFS auditing

1. Remove the contents of the file **/usr/afs/local/Audit** on each File Server for which you are no longer interested in auditing AFS events.
2. Issue the **bos restart** command (with the **-all** flag) to stop and restart all server processes on each File Server. For instructions on using this command, see "Stopping and Immediately Restarting Processes" on page 124.

Chapter 9. Managing Server Encryption Keys

This chapter explains how to maintain your cell's server encryption keys, which are vital for secure communications in AFS.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Add a new server encryption key	**bos addkey** and **kas setpassword**
Inspect key checksums in the Authentication Database	**kas examine**
Inspect key checksums in the **KeyFile**	**bos listkeys**
Remove an old server encryption key	**bos removekey**

About Server Encryption Keys

An encryption key is a string of octal numbers used to encrypt and decrypt packets of information. In AFS, a server encryption key is the key used to protect information being transferred between AFS server processes and between them and their clients. A server encryption key is essentially a password for a server process and like a user password is stored in the Authentication Database.

Maintaining your cell's server encryption keys properly is the most basic way to protect the information in your AFS filespace from access by unauthorized users.

Keys and Mutual Authentication: A Review

Server encryption keys play a central role in the mutual authentication between client and server processes in AFS. For a more detailed description of mutual authentication, see "A More Detailed Look at Mutual Authentication" on page 51.

When a client wants to contact an AFS server, it first contacts the Ticket Granting Service (TGS) module of the Authentication Server. After verifying the client's identity (based indirectly on the password of the human user whom the client represents), the TGS gives the client a server ticket. This ticket is encrypted with the server's encryption key. (The TGS also invents a second encryption key, called the session key, to be used only for a single episode of communication between server and client. The server ticket and session key, together with other pieces of information, are collectively referred to as a token.)

The client cannot read the server ticket or token because it does not know the server encryption key. However, the client sends it to the AFS server along with service requests, because the ticket proves to the AFS server processes that it has already authenticated with the TGS. AFS servers trust the TGS to grant tickets only to valid clients. The fact that the client possesses a ticket encrypted with the server's encryption key proves to the server that the client is valid. On the other hand, the client assumes that only a genuine AFS server knows the server encryption key needed to decrypt the ticket. The server's ability to decrypt the ticket and understand its contents proves to the client that the server is legitimate.

Maintaining AFS Server Encryption Keys

As you maintain your cell's server encryption keys, keep the following in mind.

- Change the key frequently to enhance your cell's security. Changing the key at least once a month is strongly recommended.
- The AFS server encryption key currently in use is stored in two places. When you add a new key, you must make changes in both places and make them in the correct order, as instructed in "Adding Server Encryption Keys" on page 337. Failure to follow the instructions can seriously impair cell functioning, as clients and servers become unable to communicate. The two storage sites for the current server encryption key are the following:

  1. The file **/usr/afs/etc/KeyFile** on the local disk of every file server machine. The file can list more than one key, each with an associated numerical identifier, the key version number or kvno. A client token records the key version number of the key used to seal it, and the server process retrieves the appropriate key from this file when the client presents the token.
  2. The **afs** entry in the Authentication Database. The current server encryption key is in the entry's password field, just like an individual user's scrambled password. The Authentication Server's Ticket Granting Service (TGS) uses this key to encrypt the tokens it gives to clients. There is only a single key in the entry, because the TGS never needs to read existing tokens, but only to generate new ones by using the current key.

  For instructions on creating the initial **afs** entry and **KeyFile** files as you install your cell's first server machine, see the IBM AFS Quick Beginnings.

- At any specific time, the tokens that the Authentication Server's Ticket Granting Service gives to clients are sealed with only one of the server encryption keys, namely the one stored in the **afs** entry in the Authentication Database.
- When you add a new server encryption key, you cannot immediately remove the former key from the **/usr/afs/etc/KeyFile** file on the local disk of every AFS server machine. Any time that you add a new key, it is likely that some clients still have valid, unexpired tokens sealed with the previous key. The more frequently you change the server encryption key, the more such tickets there are likely to be. To be able to grant service appropriately to clients with such tokens, an AFS server process must still be able to access the server encryption key used to seal it.

  You can safely delete an old server encryption key only when it is certain that no clients have tokens sealed with that key. In general, wait a period of time at least as long as the maximum token lifetime in your cell. By default, the maximum token lifetime for users is 25 hours (except for users whose Authentication Database entries were created by using the 3.0 version of AFS, for whom the default is 100 hours). You can use the **-lifetime** argument to the **kas setfields** command to change this default.

  Instructions for removing obsolete keys appear in "Removing Server Encryption Keys" on page 341.

- You create a new AFS server encryption key in much the same way regular users change their passwords, by providing a character string that is converted into an encryption key automatically. See "Adding Server Encryption Keys" on page 337.

- In addition to using server encryption keys when communicating with clients, the server processes use them to protect communications with other server processes. Therefore, all server machines in your cell must have the same version of the **KeyFile** file. The easiest way to maintain consistency (if you run the United States edition of AFS) is to use the Update Server to distribute the contents of the system control machine's **/usr/afs/etc** directory to all of the other server machines. There are two implications:
  - You must run the **upserver** process on the system control machine and an **upclientetc** process on all other server machines that references the system control machine. The IBM AFS Quick Beginnings explains how to install both processes. For instructions on verifying that the Update Server processes are running, see "Displaying Process Status and Information from the BosConfig File" on page 113.
  - Change the **KeyFile** file only on the system control machine (except in the types of emergencies discussed in "Handling Server Encryption Key Emergencies" on page 343). Any changes you make on other server machines are overwritten the next time the **upclientetc** process retrieves the contents of the system control machine's **/usr/afs/etc** directory. By default, this happens every five minutes.

  If you run the international edition of AFS, do not use the Update Server to distribute the contents of the **/usr/afs/etc** directory, particularly the **KeyFile** file. The data in the file is too sensitive for transfer in unencrypted form, and because of United States government exports regulations the international edition of AFS does not include the necessary encryption routines in a form that the Update Server can use. You must instead modify the file on each server machine individually, taking care to enter the same key on every server machine.

- Never edit the **KeyFile** directly with a text editor. Instead, always use the appropriate **bos** commands as instructed in "Adding Server Encryption Keys" on page 337 and "Removing Server Encryption Keys" on page 341.

Displaying Server Encryption Keys

To display the server encryption keys in the **/usr/afs/etc/KeyFile** file on any file server machine, use the **bos listkeys** command. Use the **kas examine** command to display the key in the Authentication Database's **afs** entry.

By default the commands do not display the actual string of octal digits that constitute a key, but rather a checksum, a decimal number derived by encrypting a constant with the key. This prevents unauthorized users from easily accessing the actual key, which they can then use to falsify or eavesdrop on protected communications. The **bos listkeys** and **kas examine** commands generate the same checksum for a given key, so displaying checksums rather than actual keys is generally sufficient. If you suspect that the keys differ in a way that the checksums are not revealing, then you are probably experiencing authentication problems throughout your cell. The easiest solution is to create a new server encryption key following the instructions in "Adding Server Encryption Keys" on page 337 or "Handling Server Encryption Key Emergencies" on page 343. Another common reason to issue the **bos listkeys** command is to display the key version numbers currently in use, in preparation for choosing the next one; here, the checksum is sufficient because the key itself is irrelevant.

If it is important to display the actual octal digits, include the **-showkey** argument to both the **bos listkeys** and **kas examine** commands.

To display the KeyFile file

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos listkeys** command to display the contents of one machine's **/usr/afs/etc/KeyFile** file.

   ```
   % bos listkeys <machine name> [-showkey]
   ```

   where

   listk

   Is the shortest acceptable abbreviation of **listkeys**.

   machine name

   Names a file server machine. In the normal case, it is acceptable to name any machine, because correct cell functioning requires that the **KeyFile** file be the same on all of them.

   -showkey

   Displays the octal digits that constitute each key.

In the following example, the output displays a checksum for each server encryption key rather than the actual octal digits. The penultimate line indicates when an administrator last changed the file, and the final line confirms that the output is complete.

```
% bos listkeys fs1.abc.com
key 0 has cksum 972037177
key 1 has cksum 2825165022
Keys last changed on Wed Jan 13 11:20:29 1999.
All done.
```

To display the afs key from the Authentication Database

1. Issue the **kas examine** command to display the **afs** entry in the Authentication Database.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the

ADMIN flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

```
% kas examine afs [-showkey]  \
             -admin  <admin principal to use for authentication>
Administrator's (admin_user) password: <admin_password>
```

where

e

Is the shortest acceptable abbreviation of **examine**.

afs

Designates the **afs** entry.

-showkey

Displays the octal digits that constitute the key.

-admin

Names an administrative account with the ADMIN flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

In the following example, the **admin** user displays the **afs** entry without using the **-showkey** flag. The second line shows the key version number in parentheses and the key's checksum. The line that begins with the string last mod reports the date on which the indicated administrator changed the key. There is no necessary relationship between this date and the date reported by the **bos listkeys** command, because the latter date changes for any type of change to the **KeyFile** file, not just a key addition. For a description of the other lines in the output from the **kas examine** command, see its reference page in the IBM AFS Administration Reference.

```
% kas examine afs  -admin admin
Administrator's (admin) password: <admin_password>
User data for afs
 key (1) cksum is 2825165022, last cpw: no date
 password will never expire.
 An unlimited number of unsuccessful authentications is permitted.
 entry expires on never. Max ticket lifetime 100.00 hours.
 last mod on Wed Jan 13 11:21:36 1999 by admin
 permit password reuse
```

Adding Server Encryption Keys

As noted, AFS records server encryption keys in two separate places:

1. In the file **/usr/afs/etc/KeyFile** on the local disk of each server machine, for use by the AFS server processes running on the machine
2. In the **afs** entry in the Authentication Database, for use by the Ticket Granting Service (TGS) when creating tokens

To ensure that server processes and the TGS share the same AFS server encryption key, execute all the steps in this section without interruption.

The following instructions include a step in which you restart the database server processes (the Authentication, Backup, Protection, and Volume Location Server processes) on all database server machines. As a database server process starts, it reads in the server encryption key that has the highest key version number in the **KeyFile** file and uses it to protect the messages that it sends for synchronizing the database and maintaining quorum. It uses the same key throughout its lifetime, which can be for an extended period, even if you remove the key from the **KeyFile** file. However, if one of the peer database server processes restarts and the others do not, quorum and database synchronization break down because the processes are no longer using the same key: the restarted process is using the key that currently has the highest key version number, and the other processes are still using the key they read in when they originally started. To avoid this problem, it is safest to restart all of the database server processes when adding a new key.

After adding a new key, you can remove obsolete keys from the **KeyFile** file to prevent it from becoming cluttered. However, you must take care not to remove keys that client or server processes are still using. For discussion and instructions, see "Removing Server Encryption Keys" on page 341.

To add a new server encryption key

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos listkeys** command to display the key version numbers that are already in use, as a first step in choosing the key version number for the new key.

   ```
   % bos listkeys <machine name>
   ```

   where

   listk

   Is the shortest acceptable abbreviation of **listkeys**.

   machine name

   Names any file server machine.

3. Choose a key version number for the new key, based on the output from Step "2" on page 338 and the following requirements:

   - A key version number must be an integer between 0 (zero) and 255 to comply with Kerberos standards. It is simplest if you keep your key version numbers in sequence by choosing a key version number one greater than the largest existing one.
   - Do not reuse a key version number currently found in the **KeyFile** file, particularly if it is also the one in the Authentication Database **afs** entry. Client processes possibly still have tickets sealed with the key that originally had that key version number, but the server processes start using the new key marked with that key version number. Because the keys do not match, the server processes refuse requests from clients who hold legitimate tokens.

4. Issue the **bos addkey** command to create a new AFS server encryption key in the **KeyFile** file.

   If you run the United States edition of AFS and use the Update Server to distribute the contents of the system control machine's **/usr/afs/etc** directory, substitute the system control machine for the machine name argument. (If you have forgotten which machine is the system control machine, see "To locate the system control machine" on page 72.)

   If you run the international edition of AFS or do not use the Update Server, repeat the **bos addkey** command, substituting each server machine in your cell for the machine name argument in turn.

   To avoid visible echoing of the string that corresponds to the new key, omit the **-key** argument from the command line; instead enter the string at the prompts that appear when you omit it, as shown in the following syntax specification.

   ```
   % bos addkey  -server <machine name> -kvno <key version number>
   input key: <afs_password>
   Retype input key: <afs_password>
   ```

   where

   addk

   Is the shortest acceptable abbreviation of **addkey**.

   -server

   Names the cell's system control machine if you are using the Update Server, or each server machine in turn if you are not.

   -kvno

   Specifies the new key's key version number as an integer from the range 0 (zero) through 255.

   Remember the number. You need to use it again in Step "6" on page 340. If you are using the international edition of AFS, be sure to type the same number each time you issue this command.

afs_password

Is a character string similar to a user password, of any length from one to about 1,000 characters. To improve security, include nonalphabetic characters and make the string as long as is practical (you need to type it only in this step and in Step "6" on page 340). If you are using the international edition of AFS, be sure to type the same string each time you issue this command.

Do not enter an octal string directly. The BOS Server scrambles the character string into an octal string appropriate for use as an encryption key before recording it in the **KeyFile** file.

5. If you are using the Update Server, wait for a few minutes while the Update Server distributes the new **KeyFile** file to all server machines. The maximum necessary waiting period is the largest value provided for the **-t** argument to the **upclientetc** process's initialization command used on any of the server machines; the default time is five minutes.

   To be certain that all machines have the same **KeyFile** file, issue the **bos listkeys** command for every file server machine and verify that the checksum for the new key is the same on all machines.

   ```
   % bos listkeys <machine name>
   ```

   If you are not using the Update Server, try to complete Step "4" on page 339 within five minutes.

6. Issue the **kas setpassword** command to enter the same key in the **afs** entry in the Authentication Database.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas setpassword -name afs -kvno <kvno>  \
                      -admin  <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   new_password: afs_password
   Verifying, please re-enter new_password: <admin_password>
   ```

   where

sp

Is an acceptable alias for **setpassword** (**setp** is the shortest acceptable abbreviation).

-name afs

Creates the new key in the **afs** entry.

-kvno

Specifies the same key version number as in Step "4" on page 339.

-admin

Names an administrative account with the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

afs_password

Is the same character string you entered in Step "4" on page 339.

7. **(Optional.)** If you want to verify that the keys you just created in the **KeyFile** file and the Authentication Database **afs** entry are identical and have the same key version number, follow the instructions in "Displaying Server Encryption Keys" on page 335.
8. Issue the **bos restart** command to restart the database server processes on all database server machines. This forces them to start using the key in the **KeyFile** file that currently has the highest key version number.

   Repeat this command in quick succession for each database server machine, starting with the machine that has the lowest IP address.

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

   where

   res

   Is the shortest acceptable abbreviation of **restart**.

   machine name

   Names each database server machine in turn.

   buserver kaserver ptserver vlserver

   Designates the Backup Server, Authentication Server, Protection Server, and Volume Location (VL) Server, respectively.

Removing Server Encryption Keys

You can periodically remove old keys from the **/usr/afs/etc/KeyFile** file to keep it to a reasonable size. To avoid disturbing cell functioning, do not remove an old key until all tokens sealed with the key and held by users or client processes have expired. After adding a new key, wait to remove old keys at least as long as the longest token lifetime you use in your cell. For Authentication Database user entries created under AFS version 3.1 or higher, the default token lifetime is 25 hours; for entries created under AFS version 3.0, it is 100 hours.

There is no command for removing the key from the **afs** entry in the Authentication Database, because the key field in that entry must never be empty. Use the **kas setpassword** command to replace the **afs** key, but only as part of the complete procedure detailed in "To add a new server encryption key" on page 338.

Never remove from the **KeyFile** file the key that is currently in the **afs** entry in the Authentication Database. AFS server processes become unable to decrypt the tickets that clients present to them.

To remove a key from the KeyFile file

1. Verify that you are authenticated as a user listed in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

   ```
   % bos listusers <machine name>
   ```

2. Issue the **bos listkeys** command to display the key version number of each key you want to remove. The output also reveals whether it has been at least 25 hours since a new key was placed in the **KeyFile** file. For complete instructions for the **bos listkeys** command, see "To display the KeyFile file" on page 336.

   ```
   % bos listkeys <machine name>
   ```

3. Issue the **kas examine** command to verify that the key currently in the Authentication Database's **afs** entry does not have the same key version number as any of the keys you are removing from the **KeyFile** file. For detailed instructions for the **kas examine** command, see "To display the afs key from the Authentication Database" on page 336.

   ```
   % kas examine afs  -admin <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ```

4. Issue the **bos removekey** command to remove one or more server encryption keys from the **KeyFile** file.

   If you run the United States edition of AFS and use the Update Server to distribute the contents of the system control machine's **/usr/afs/etc** directory, substitute the system control machine for the machine name argument. (If you have forgotten which machine is the system control machine, see "To locate the system control machine" on page 72.)

   If you run the international edition of AFS or do not use the Update Server, repeat the **bos removekey** command, substituting each server machine in your cell for the machine name argument in turn.

   ```
   % bos removekey <machine name> <key version number>
   ```

   where

   removek

   Is the shortest acceptable abbreviation of **removekey**.

   machine name

   Names the cell's system control machine if you are using the Update Server, or each server machine in turn if you are not.

key version number

Specifies the key version number of each key to remove.

Handling Server Encryption Key Emergencies

In rare circumstances, the AFS server processes can become unable to decrypt the server tickets that clients or peer server processes are presenting. Activity in your cell can come to a halt, because the server processes believe that the tickets are forged or expired, and refuse to execute any actions. This can happen on one machine or several; the effect is more serious when more machines are involved.

One common cause of server encryption key problems is that the client's ticket is encrypted with a key that the server process does not know. Usually this means that the **/usr/afs/etc/KeyFile** on the server machine does not include the key in the **afs** Authentication Database entry, which the Authentication Server's Ticket Granting Service (TGS) module is using to encrypt server tickets.

Another possibility is that the **KeyFile** files on different machines do not contain the same keys. In this case, communications among server processes themselves become impossible. For instance, AFS's replicated database mechanism (Ubik) breaks down if the instances of a database server process on the different database server machines are not using the same key.

The appearance of the following error message when you direct a **bos** command to a file server machine in the local cell is one possible symptom of server encryption key mismatch. (Note, however, that you can also get this message if you forget to include the **-cell** argument when directing the **bos** command to a file server machine in a foreign cell.)

```
bos: failed to contact host's bosserver (security object was passed a bad ticket).
```

The solution to server encryption key emergencies is to put a new AFS server encryption key in both the Authentication Database and the **KeyFile** file on every server machine, so that the TGS and all server processes again share the same key.

Handling key emergencies requires some unusual actions. The reasons for these actions are explained in the following sections; the actual procedures appear in the subsequent instructions.

Prevent Mutual Authentication

It is necessary to prevent the server processes from trying to mutually authenticate with you as you deal with a key emergency, because they possibly cannot decrypt your token. When you do not mutually authenticate, the server processes assign you the identity **anonymous**. To prevent mutual authentication, use the **unlog** command to discard your tokens and include the **-noauth** flag on every command where it is available.

Disable Authorization Checking by Hand

Because the server processes recognize you as the user **anonymous** when you do not mutually authenticate, you must turn off authorization checking. Only with authorization checking disabled do the server processes allow the **anonymous** user to perform privileged actions such as key creation.

In an emergency, disable authorization checking by creating the file **/usr/afs/local/NoAuth** by hand. In normal circumstances, use the **bos setauth** command instead.

Work Quickly on Each Machine

Disabling authorization checking is a serious security exposure, because server processes on the affected machine perform any action for anyone. Disable authorization checking only for as long as necessary, completing all steps in an uninterrupted session and as quickly as possible.

Work at the Console

Working at the console of each server machine on which you disable authorization checking ensures that no one else logs onto the console while you are working there. It does not prevent others from connecting to the machine remotely (using the **telnet** program, for example), which is why it is important to work quickly. The only way to ensure complete security is to disable network traffic, which is not a viable option in many environments. You can improve security in general by limiting the number of people who can connect remotely to your server machines at any time, as recommended in "Improving Security in Your Cell" on page 50.

Change Individual KeyFile Files

If you use the Update Server to distribute the contents of the **/usr/afs/etc** directory, an emergency is the only time when it is appropriate to change the **KeyFile** file on individual machines instead. Updating each machine's file is necessary because mismatched keys can prevent the system control machine's **upserver** process from mutually authenticating with **upclientetc** processes on other server machines, in which case the **upserver** process refuses to distribute its **KeyFile** file to them.

Even if it appears that the Update Server is working correctly, the only way to verify that is to change the key on the system control machine and wait the standard delay period to see if the **upclientetc** processes retrieve the key. During an emergency, it does not usually make sense to wait the standard delay period. It is more efficient simply to update the file on each server machine separately. Also, even if the Update Server can distribute the file correctly, other processes can have trouble because of mismatched keys. The following instructions add the new key file on the system control machine first. If the Update Server is working, then it is distributing the same change as you are making on each server machine individually.

If your cell does not use the Update Server, or uses the international edition of AFS, you always change keys on server machines individually. The following instructions are also appropriate for you.

Two Component Procedures

There are two subprocedures used frequently in the following instructions: disabling authorization checking and reenabling it. For the sake of clarity, the procedures are detailed here; the instructions refer to them as necessary.

Disabling Authorization Checking in an Emergency

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Create the file **/usr/afs/local/NoAuth** to disable authorization checking.

   ```
   # touch /usr/afs/local/NoAuth
   ```

3. Discard your tokens, in case they were sealed with an incompatible key, which can prevent some commands from executing.

   ```
   # unlog
   ```

Reenabling Authorization Checking in an Emergency

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Remove the **/usr/afs/local/NoAuth** file.

   ```
   # rm /usr/afs/local/NoAuth
   ```

3. Authenticate as an administrative identity that belongs to the **system:administrators** group and is listed in the **/usr/afs/etc/UserList** file.

   ```
   # klog <admin_user>
   Password: <admin_password>
   ```

4. If appropriate, log out from the console (or close the remote connection you are using), after issuing the **unlog** command to destroy your tokens.

To create a new server encryption key in emergencies

1. **On the system control machine**, disable authorization checking as instructed in "Disabling Authorization Checking in an Emergency" on page 345.
2. Issue the **bos listkeys** command to display the key version numbers already in use in the **KeyFile** file, as a first step in choosing the new key's key version number.

   ```
   # bos listkeys <machine name> -noauth
   ```

   where

listk

Is the shortest acceptable abbreviation of **listkeys**.

machine name

Specifies a file server machine.

-noauth

Bypasses mutual authentication with the BOS Server. Include it in case the key emergency is preventing successful mutual authentication.

3. Choose a key version number for the new key, based on what you learned in Step "2" on page 345 plus the following requirements:
   - It is best to keep your key version numbers in sequence by choosing a key version number one greater than the largest existing one.
   - Key version numbers must be integers between 0 and 255 to comply with Kerberos standards.
   - Do not reuse a key version number currently listed in the **KeyFile** file.
4. **On the system control machine**, issue the **bos addkey** command to create a new AFS server encryption key in the **KeyFile** file.

```
# bos addkey <machine name> -kvno <key version number> -noauth
input key: <afs_password>
Retype input key: <afs_password>
```

where

addk

Is the shortest acceptable abbreviation of **addkey**.

machine name

Names the file server machine on which to define the new key in the **KeyFile** file.

-kvno

Specifies the key version number you chose in Step "3" on page 346, an integer in the range 0 (zero) through 255. You must specify the same number in Steps "7" on page 347, "8" on page 348, and "13" on page 348.

-noauth

Bypasses mutual authentication with the BOS Server. Include it in case the key emergency is preventing successful mutual authentication.

afs_password

Is a character string similar to a user password, of any length from one to about 1,000 characters. To improve security, make the string as long as is practical, and include nonalphabetic characters.

Do not type an octal string directly. The BOS Server scrambles the character string into an octal string appropriate for use as an encryption key before recording it in the **KeyFile** file.

Remember the string. You need to use it again in Steps "7" on page 347, "8" on page 348, and "13" on page 348.

5. **On every database server machine in your cell** (other than the system control machine), disable authorization checking as instructed in "Disabling Authorization Checking in an Emergency" on page 345. Do not repeat the procedure on the system control machine, if it is a database server machine, because you already disabled authorization checking in Step "1" on page 345. (If you need to learn which machines are database server machines, use the **bos listhosts** command as described in "To locate database server machines" on page 71.)
6. Wait at least 90 seconds after finishing Step "5" on page 347, to allow each of the database server processes (the Authentication, Backup, Protection and Volume Location Servers) to finish electing a new sync site. Then issue the **udebug** command to verify that the election worked properly. Issue the following commands, substituting each database server machine's name for server machine in turn. Include the system control machine if it is a database server machine.

   ```
   # udebug <server machine> buserver
   # udebug <server machine> kaserver
   # udebug <server machine> ptserver
   # udebug <server machine> vlserver
   ```

   For each process, the output from all of the database server machines must agree on which one is the sync site for the process. It is not, however, necessary that the same machine serves as the sync site for each of the four processes. For each process, the output from only one machine must include the following string:

   ```
   I am sync site ...
   ```

   The output on the other machines instead includes the following line

   ```
   I am not sync site
   ```

   and a subsequent line that begins with the string `Sync host` and specifies the IP address of the machine claiming to be the sync site.

   If the output does not meet these requirements or seems abnormal in another way, contact AFS Product Support for assistance.

7. **On every database server machine in your cell** (other than the system control machine), issue the **bos addkey** command described in Step "4" on page 346. Be sure to use the same values for afs_password and kvno as you used in that step.

8. Issue the **kas setpassword** command to define the new key in the Authentication Database's **afs** entry. It must match the key you created in Step "4" on page 346 and Step "7" on page 347.

   ```
   # kas setpassword  -name afs  -kvno <key version number> -noauth
   new_password: <afs_password>
   Verifying, please re-enter new_password: <afs_password>
   ```

   where

   sp

   Is an acceptable alias for **setpassword** (**setp** is the shortest acceptable abbreviation).

   -kvno

   Is the same key version number you specified in Step "4" on page 346.

   afs_password

   Is the same character string you specified as afs_password in Step "4" on page 346. It does not echo visibly.

9. **On every database server machine in your cell** (including the system control machine if it is a database server machine), reenable authorization checking as instructed in "Reenabling Authorization Checking in an Emergency" on page 345. If the system control machine is not a database server machine, do not perform this procedure until Step "11" on page 348.
10. Repeat Step "6" on page 347 to verify that each database server process has properly elected a sync site after being restarted in Step "9" on page 348.
11. **On the system control machine** (if it is not a database server machine), reenable authorization checking as instructed in "Reenabling Authorization Checking in an Emergency" on page 345. If it is a database server machine, you already performed the procedure in Step "9" on page 348.
12. **On all remaining (simple) file server machines**, disable authorization checking as instructed in "Disabling Authorization Checking in an Emergency" on page 345.
13. **On all remaining (simple) file server machines**, issue the **bos addkey** command described in Step "4" on page 346. Be sure to use the same values for afs_password and kvno as you used in that step.
14. **On all remaining (simple) file server machines**, reenable authorization checking as instructed in "Reenabling Authorization Checking in an Emergency" on page 345.

III. Managing Client Machines

Chapter 10. Administering Client Machines and the Cache Manager

This chapter describes how to administer an AFS client machine, which is any machine from which users can access the AFS filespace and communicate with AFS server processes. (A client machine can simultaneously function as an AFS server machine if appropriately configured.) An AFS client machine has the following characteristics:

- The kernel includes the set of modifications, commonly referred to as the *Cache Manager*, that enable access to AFS files and directories. You can configure many of the Cache Manager's features to suit your users' needs. See "Overview of Cache Manager Customization" on page 352.
- The **/usr/vice/etc** directory on the local disk stores several configuration files. See "Configuration Files in the /usr/vice/etc Directory" on page 354.
- A cache stores temporary copies of data fetched from AFS file server machines, either in machine memory or on a devoted local disk partition. See "Determining the Cache Type, Size, and Location" on page 356 and "Setting Other Cache Parameters with the afsd program" on page 360.

To learn how to install the client functionality on a machine, see the *IBM AFS Quick Beginnings*.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Display cache size set at reboot	**cat /usr/vice/etc/cacheinfo**
Display current cache size and usage	**fs getcacheparms**
Change disk cache size without rebooting	**fs setcachesize**
Initialize Cache Manager	**afsd**
Display contents of **CellServDB** file	**cat /usr/vice/etc/CellServDB**
Display list of database server machines from kernel memory	**fs listcells**
Change list of database server machines in kernel memory	**fs newcell**
Check cell's status regarding setuid	**fs getcellstatus**
Set cell's status regarding setuid	**fs setcell**
Set server probe interval	**fs checkservers -interval**
Display machine's cell membership	**cat /usr/vice/etc/ThisCell**
Change machine's cell membership	Edit **/usr/vice/etc/ThisCell**
Flush cached file/directory	**fs flush**
Flush everything cached from a volume	**fs flushvolume**
Update volume-to-mount-point mappings	**fs checkvolumes**
Display Cache Manager's server preference ranks	**fs getserverprefs**
Set Cache Manager's server preference ranks	**fs setserverprefs**
Display client machine addresses to register	**fs getclientaddrs**
Set client machine addresses to register	**fs setclientaddrs**
Control the display of warning and status messages	**fs messages**

Display and change machine’s system type	**fs sysname**
Enable asynchronous writes	**fs storebehind**

Overview of Cache Manager Customization

An AFS client machine’s kernel includes a set of modifications, commonly referred to as the *Cache Manager*, that enable access to AFS files and directories and communications with AFS server processes. It is common to speak of the Cache Manager as a process or program, and in regular usage it appears to function like one. When configuring it, though, it is helpful to keep in mind that this usage is not strictly accurate.

The Cache Manager mainly fetches files on behalf of application programs running on the machine. When an application requests an AFS file, the Cache Manager contacts the Volume Location (VL) Server to obtain a list of the file server machines that house the volume containing the file. The Cache Manager then translates the application program’s system call requests into remote procedure calls (RPCs) to the File Server running on the appropriate machine. When the File Server delivers the file, the Cache Manager stores it in a local *cache* before delivering it to the application program.

The File Server delivers a data structure called a *callback* along with the file. (To be precise, it delivers a callback for each file fetched from a read/write volume, and a single callback for all data fetched from a read-only volume.) A valid callback indicates that the Cache Manager’s cached copy of a file matches the central copy maintained by the File Server. If an application on another AFS client machine changes the central copy, the File Server breaks the callback, and the Cache Manager must retrieve the new version when an application program on its machine next requests data from the file. As long as the callback is unbroken, however, the Cache Manager can continue to provide the cached version of the file to applications on its machine, which eliminates unnecessary network traffic.

The indicated sections of this chapter explain how to configure and customize the following Cache Manager features. All but the first (choosing disk or memory cache) are optional, because AFS sets suitable defaults for them.

- *disk or memory cache*. The AFS Cache Manager can use machine memory for caching instead of space on the local disk. Deciding which to use is the most basic configuration decision you must make. See "Determining the Cache Type, Size, and Location" on page 356.
- *cache size*. Cache size probably has the most direct influence on client machine performance. It determines how often the Cache Manager must contact the File Server across the network or discard cached data to make room for newly requested files, both of which affect how quickly the Cache Manager delivers files to users. See "Determining the Cache Type, Size, and Location" on page 356.
- *cache location*. For a disk cache, you can alter the conventional cache directory location (**/usr/vice/cache**) to take advantage of greater space availability on other disks on the machine. A larger cache can result in faster file delivery. See "Determining the Cache Type, Size, and Location" on page 356.
- *chunk size and number*. The **afsd** program, which initializes the Cache Manager, allows you to control the size and number of chunks into which a cache is divided, plus related parameters. Setting these parameters is optional, because there are reasonable defaults, but it provides precise control. The AFS distribution includes configuration scripts that set Cache Manager parameters to values that are

reasonable for different configurations and usage patterns. See "Setting Other Cache Parameters with the afsd program" on page 360.

- *knowledge of database server machines*. Enable access to a cell's AFS filespace and other services by listing the cell's database server machines in the **/usr/vice/etc/CellServDB** file on the local disk. See "Maintaining Knowledge of Database Server Machines" on page 364.
- *setuid privilege*. You can control whether the Cache Manager allows programs from a cell to execute with setuid permission. See "Determining if a Client Can Run Setuid Programs" on page 369.
- *cell membership*. Each client belongs to a one cell defined by the local **/usr/vice/etc/ThisCell** file. Cell membership determines the default cell in which the machine's users are authenticated and in which AFS commands run. See "Setting a Client Machine's Cell Membership" on page 372.
- *cached file version*. AFS's system of callbacks normally guarantees that the Cache Manager has the most current versions of files and directories possible. Nevertheless, you can force the Cache Manager to fetch the most current version of a file from the File Server if you suspect that the cache contains an outdated version. See "Forcing the Update of Cached Data" on page 373.
- *File Server and Volume Location Server preferences*. The Cache Manager sets numerical preference ranks for the interfaces on file server machines and Volume Server (VL) machines. The ranks determine which interface the Cache Manager first attempts to use when fetching data from a volume or from the Volume Location Database (VLDB). The Cache Manager sets default ranks as it initializes, basing them on its network proximity to each interface, but you can modify the preference ranks if you wish. See "Maintaining Server Preference Ranks" on page 375.
- *interfaces registered with the File Server*. If the Cache Manager is multihomed (has multiple interface addresses), you can control which of them it registers for File Servers to use when they initiate RPCs to the client machine. See "Managing Multihomed Client Machines" on page 379.
- *display of information messages*. By default, the Cache Manager sends basic error and informational messages to the client machine's console and to command shells. You can disable the messaging. See "Controlling the Display of Warning and Informational Messages" on page 382.
- *system type*. The Cache Manager records the local machine's AFS system type in kernel memory, and substitutes the value for the @sys variable in pathnames. See "Displaying and Setting the System Type Name" on page 383.
- *delayed writes*. By default, the Cache Manager writes all data to the File Server immediately and synchronously when an application program closes a file. You can enable asynchronous writes, either for an individual file, or all files that the Cache Manager handles, and set how much data remains to be written when the Cache Manager returns control to the closing application. See "Enabling Asynchronous Writes" on page 385.

You must make all configuration changes on the client machine itself (at the console or over a direct connection such as a **telnet** connection). You cannot configure the Cache Manager remotely. You must be logged in as the local superuser **root** to issue some commands, whereas others require no privilege. All files mentioned in this chapter must actually reside on the local disk of each AFS client machine (they cannot, for example, be symbolic links to files in AFS).

AFS's **package** program can simplify other aspects of client machine configuration, including those normally set in the machine's AFS initialization file. See "Configuring Client Machines with the package Program" on page 389.

Configuration and Cache-Related Files on the Local Disk

This section briefly describes the client configuration files that must reside in the local **/usr/vice/etc** directory on every client machine. If the machine uses a disk cache, there must be a partition devoted to cache files; by convention, it is mounted at the **/usr/vice/cache** directory.

Note for Windows users: Some files described in this document possibly do not exist on machines that run a Windows operating system. Also, Windows uses a backslash (\) rather than a forward slash (**/**) to separate the elements in a pathname.

Configuration Files in the /usr/vice/etc Directory

The **/usr/vice/etc** directory on a client machine's local disk must contain certain configuration files for the Cache Manager to function properly. They control the most basic aspects of Cache Manager configuration.

If it is important that the client machines in your cell perform uniformly, it is most efficient to update these files from a central source. The following descriptions include pointers to sections that discuss how best to maintain the files.

afsd

The binary file for the program that initializes the Cache Manager. It must run each time the machine reboots in order for the machine to remain an AFS client machine. The program also initializes several daemons that improve Cache Manager functioning, such as the process that handles callbacks.

cacheinfo

A one-line file that sets the cache's most basic configuration parameters: the local directory at which the Cache Manager mounts the AFS filespace, the local disk directory to use as the cache, and how many kilobytes to allocate to the cache.

The *IBM AFS Quick Beginnings* explains how to create this file as you install a client machine. To change the cache size on a machine that uses a memory cache, edit the file and reboot the machine. On a machine that uses a disk cache, you can change the cache size without rebooting by issuing the **fs setcachesize** command. For instructions, see "Determining the Cache Type, Size, and Location" on page 356.

CellServDB

This ASCII file names the database server machines in the local cell and in any foreign cell to which you want to enable access from this machine. (Database server machines are the machines in a cell that run the Authentication, Backup, Protection, and VL Server processes; see "Database Server Machines" on page 69.)

The Cache Manager must be able to reach a cell's database server machines to fetch files from its filespace. Incorrect or missing information in the **CellServDB** file can slow or completely block access. It is important to update the file whenever a cell's database server machines change.

As the **afsd** program initializes the Cache Manager, it loads the contents of the file into kernel memory. The Cache Manager does not read the file between reboots, so to incorporate changes to the file into kernel memory, you must reboot the machine. Alternatively, you can issue the **fs newcell** command to insert the changes directly into kernel memory without changing the file. It can also be convenient to upgrade the file from a central source. For instructions, see "Maintaining Knowledge of Database Server Machines" on page 364.

(The **CellServDB** file on client machines is not the same as the one kept in the **/usr/afs/etc** directory on server machines, which lists only the local cell's database server machines. For instructions on maintaining the server **CellServDB** file, see "Maintaining the Server CellServDB File" on page 88).

NetInfo

This optional ASCII file lists one or more of the network interface addresses on the client machine. If it exists when the Cache Manager initializes, the Cache Manager uses it as the basis for the list of interfaces that it registers with File Servers. See "Managing Multihomed Client Machines" on page 379.

NetRestrict

This optional ASCII file lists one or more network interface addresses. If it exists when the Cache Manager initializes, the Cache Manager removes the specified addresses from the list of interfaces that it registers with File Servers. See "Managing Multihomed Client Machines" on page 379.

ThisCell

This ASCII file contains a single line that specifies the complete domain-style name of the cell to which the machine belongs. Examples are `abc.com` and `stateu.edu`. This value defines the default cell in which the machine's users become authenticated, and in which the command interpreters (for example, the **bos** command) contact server processes.

The *IBM AFS Quick Beginnings* explains how to create this file as you install the AFS client functionality. To learn about changing a client machine's cell membership, see "Setting a Client Machine's Cell Membership" on page 372.

In addition to these files, the **/usr/vice/etc** directory also sometimes contains the following types of files and subdirectories:

- The AFS initialization script, called **afs.rc** on many system types. In the conventional configuration specified by the *IBM AFS Quick Beginnings*, it is a symbolic link to the actual script kept in the same directory as other initialization files used by the operating system.
- A subdirectory that houses AFS kernel library files used by a dynamic kernel loading program.
- A subdirectory called **C**, which houses the Cache Manager catalog file called **afszcm.cat**. The fstrace program uses the catalog file to translate operation codes into character strings, which makes the message in the trace log more readable. See "About the fstrace Command Suite" on page 306.

Cache-Related Files

A client machine that uses a disk cache must have a local disk directory devoted to the cache. The conventional mount point is **/usr/vice/cache**, but you can use another partition that has more available space.

Do not delete or directly modify any of the files in the cache directory. Doing so can cause a kernel panic, from which the only way to recover is to reboot the machine. By default, only the local superuser **root** can read the files directly, by virtue of owning them.

A client machine that uses a memory cache keeps all of the information stored in these files in machine memory instead.

CacheItems

A binary-format file in which the Cache Manager tracks the contents of cache chunks (the **V** files in the directory, described just following), including the file ID number (fID) and the data version number.

VolumeItems

A binary-format file in which the Cache Manager records the mapping between mount points and the volumes from which it has fetched data. The Cache Manager uses the information when responding to the **pwd** command, among others.

Vn

A cache chunk file, which expands to a maximum size (by default, 64 KB) to house data fetched from AFS files. The number of **V**n files in the cache depends on the cache size among other factors. The n is the index assigned to each file; they are numbered sequentially, but the Cache Manager does not necessarily use them in order or contiguously. If an AFS file is larger than the maximum size for **V**n files, the Cache Manager divides it across multiple **V**n files.

Determining the Cache Type, Size, and Location

This section explains how to configure a memory or disk cache, how to display and set the size of either type of cache, and how to set the location of the cache directory for a disk cache.

The Cache Manager uses a disk cache by default, and it is the preferred type of caching. To configure a memory cache, include the **-memcache** flag on the **afsd** command, which is normally invoked in the machine's AFS initialization file. If configured to use a memory cache, the Cache Manager does no disk caching, even if the machine has a disk.

Choosing the Cache Size

Cache size influences the performance of a client machine more directly than perhaps any other cache parameter. The larger the cache, the faster the Cache Manager is likely to deliver files to users. A small cache can impair performance because it increases the frequency at which the Cache Manager must discard cached data to make room for newly requested data. When an application asks for data that has been discarded, the Cache Manager must request it from the File Server, and fetching data across the network is almost always slower than fetching it from the local disk. The Cache Manager never discards data from a file that has been modified locally but not yet stored back to the File Server. If the cache is very small, the Cache Manager possible cannot find any data to discard. For more information about the algorithm it uses when discarding cached data, see "How the Cache Manager Chooses Data to Discard" on page 360).

The amount of disk or memory you devote to caching depends on several factors. The amount of space available in memory or on the partition housing the disk cache directory imposes an absolute limit. In addition, you cannot allocate more than 95% of the space available on the cache directory's partition to a disk cache. The **afsd** program exits without starting the Cache Manager and prints an appropriate message to the standard output stream if you violate this restriction. 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
```

where number is how many kilobytes were allocated just before the failure.

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 usually work, and (for a memory cache) the number of processes that usually 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.

AFS imposes an absolute limit on cache size in some versions. See the *IBM AFS Release Notes* for the version you are using.

Displaying and Setting the Cache Size and Location

The Cache Manager determines how big to make the cache by reading the **/usr/vice/etc/cacheinfo** file as it initializes. As directed in the *IBM AFS Quick Beginnings*, you must create the file before running the **afsd** program. The file also defines the directory on which to mount AFS (by convention, **/afs**), and the local disk directory to use for a cache directory.

To change any of the values in the file, log in as the local superuser **root**. You must reboot the machine to have the new value take effect. For instructions, see "To edit the cacheinfo file" on page 358.

To change the cache size at reboot without editing the **cacheinfo** file, include the **-blocks** argument to the **afsd** command; see the command's reference page in the IBM AFS Administration Reference.

For a disk cache, you can also use the **fs setcachesize** command to reset the cache size without rebooting. The value you set persists until the next reboot, at which time the cache size returns to the value specified in the **cacheinfo** file or by the **-blocks** argument to the **afsd** command. For instructions, see "To change the disk cache size without rebooting" on page 359.

To display the current cache size and the amount of space the Cache Manager is using at the moment, use the **fs getcacheparms** command as detailed in "To display the current cache size" on page 358.

To display the cache size set at reboot

1. Use a text editor or the **cat** command to display the contents of the **/usr/vice/etc/cacheinfo** file.

   ```
   % cat /usr/vice/etc/cacheinfo
   ```

To display the current cache size

1. Issue the **fs getcacheparms** command on the client machine.

   ```
   % fs getcacheparms
   ```

   where **getca** is the shortest acceptable abbreviation of **getcacheparms**.

   The output shows the number of kilobyte blocks the Cache Manager is using as a cache at the moment the command is issued, and the current size of the cache. For example:

   ```
   AFS using 13709 of the cache's available 15000 1K byte blocks.
   ```

To edit the cacheinfo file

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Use a text editor to edit the **/usr/vice/etc/cacheinfo** file, which has three fields, separated by colons:

   - The first field names the local directory on which to mount the AFS filespace. The conventional location is **/afs**.
   - 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.
   - The third field defines cache size as a number of kilobyte (1024-byte) blocks.

   The following example mounts the AFS filespace at the **/afs** directory, names **/usr/vice/cache** as the cache directory, and sets cache size to 50,000 KB:

   ```
             /afs:/usr/vice/cache:50000
   ```

To change the disk cache size without rebooting

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs setcachesize** command to set a new disk cache size.

   Note: This command does not work for a memory cache.

   ```
   # fs setcachesize <size in 1K byte blocks (0 => reset)>
   ```

   where

   setca

   Is the shortest acceptable abbreviation of **setcachesize**.

   size in 1K byte blocks (0 => reset)

   Sets the number of kilobyte blocks to be used for the cache. Specify a positive integer (**1024** equals 1 MB), or **0** (zero) to reset the cache size to the value specified in the **cacheinfo** file.

To reset the disk cache size to the default without rebooting

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs setcachesize** command to reset the size of the local disk cache (the command does not work for a memory cache). Choose one of the two following options:

   - To reset the cache size to the value specified in the local **cacheinfo** file, specify the value **0** (zero)

     ```
     # fs setcachesize 0
     ```

   - To reset the cache size to the value set at the last reboot of the machine, include the **-reset** flag. Unless the **-blocks** argument was used on the **afsd** command, this is also the value in the **cacheinfo** file.

     ```
     # fs setcachesize -reset
     ```

   where

setca

Is the shortest acceptable abbreviation of **setcachesize**.

0

Resets the disk cache size to the value in the third field of the **/usr/vice/etc/cacheinfo** file.

-reset

Resets the cache size to the value set at the last reboot.

How the Cache Manager Chooses Data to Discard

When the cache is full and application programs request more data from AFS, the Cache Manager must flush out cache chunks to make room for the data. The Cache Manager considers two factors:

1. How recently an application last accessed the data.
2. Whether the chunk is *dirty*. A dirty chunk contains changes to a file that have not yet been saved back to the permanent copy stored on a file server machine.

The Cache Manager first checks the least-recently used chunk. If it is not dirty, the Cache Manager discards the data in that chunk. If the chunk is dirty, the Cache Manager moves on to check the next least recently used chunk. It continues in this manner until it has created a sufficient number of empty chunks.

Chunks that contain data fetched from a read-only volume are by definition never dirty, so the Cache Manager can always discard them. Normally, the Cache Manager can also find chunks of data fetched from read/write volumes that are not dirty, but a small cache makes it difficult to find enough eligible data. If the Cache Manager cannot find any data to discard, it must return I/O errors to application programs that request more data from AFS. Application programs usually have a means for notifying the user of such errors, but not for revealing their cause.

Setting Other Cache Parameters with the afsd program

There are only three cache configuration parameters you must set: the mount directory for AFS, the location of the disk cache directory, and the cache size. They correspond to the three fields in the **/usr/vice/etc/cacheinfo** file, as discussed in "Determining the Cache Type, Size, and Location" on page 356. However, if you want to experiment with fine-tuning cache performance, you can use the arguments on the **afsd** command to control several other parameters. This section discusses a few of these parameters that have the most direct effect on cache performance. To learn more about the **afsd** command's arguments, see its reference page in the *IBM AFS Administration Reference*.

In addition, the AFS initialization script included in the AFS distribution for each system type includes several variables that set several **afsd** arguments in a way that is suitable for client machines of different sizes and usage patterns. For instructions on using the script most effectively, see the section on configuring the Cache Manager in the *IBM AFS Quick Beginnings*.

Setting Cache Configuration Parameters

The cache configuration parameters with the most direct effect on cache performance include the following:

- *total cache size*. This is the amount of disk space or machine memory available for caching, as discussed in detail in "Determining the Cache Type, Size, and Location" on page 356.
- *number of cache chunks*. For a disk cache, each chunk is a **V**n file in the local cache directory (see "Cache-Related Files" on page 355). For a memory cache, each chunk is a set of contiguous blocks allocated in machine memory.

  This parameter does not have as much of an effect on cache performance as total size. However, adjusting it can influence how often the Cache Manager must discard cached data to make room for new data. Suppose, for example, that you set the disk cache size to 50 MB and the number of chunks (**V**n files) to 1,000. If each of the ten users on the machine caches 100 AFS files that average 20 KB in size, then all 1,000 chunks are full (a chunk can contain data from only one AFS file) but the cache holds only about 20 MB of data. When a user requests more data from the File Server, the Cache Manager must discard cached data to reclaim some chunks, even though the cache is filled to less than 50% of its capacity. In such a situation, increasing the number of chunks enables the Cache Manager to discard data less often.

- *chunk size*. This parameter determines the maximum amount of data that can fit in a chunk. If a cached element is smaller than the chunk size, the remaining space in the chunk is not used (a chunk can hold no more than one element). If an element cannot fit in a single chunk, it is split across as many chunks as needed. This parameter also determines how much data the Cache Manager requests at a time from the File Server (how much data per *fetch RPC*, because AFS uses partial file transfer).

  The main reason to change chunk size is because of its relation to the amount of data fetched per RPC. If your network links are very fast, it can improve performance to increase chunk size; if the network is especially slow, it can make sense to decrease chunk size.

- *number of dcache entries in memory*. The Cache Manager maintains one dcache entry for each cache chunk, recording a small amount of information, such as the file ID (fID) and version number of the AFS file corresponding to the chunk.

  For a disk cache, dcache entries reside in the **/usr/vice/cache/CacheItems** file; a small number are duplicated in machine memory to speed access.

  For a memory cache, the number of dcache entries equals the number of cache chunks. For a discussion of the implications of this correspondence, see "Controlling Memory Cache Configuration" on page 363.

For a description of how the Cache Manager determines defaults for number of chunks, chunk size, and number of dcache entries in a disk cache, see "Configuring a Disk Cache" on page 362; for a memory cache, see "Controlling Memory Cache Configuration" on page 363. The instructions also explain how to use the **afsd** command's arguments to override the defaults.

Configuring a Disk Cache

The default number of cache chunks (**V**n files) in a disk cache is calculated by the **afsd** command to be the greatest of the following:

- 100
- 1.5 times the result of dividing cache size by chunk size (cachesize/chunksize * 1.5)
- The result of dividing cachesize by 10 MB (cachesize/10240)

You can override this value by specifying a positive integer with the **-files** argument. Consider increasing this value if more than 75% of the **V**n files are already used soon after the Cache Manager finishes initializing. Consider decreasing it if only a small percentage of the chunks are used at that point. In any case, never specify a value less than 100, because a smaller value can cause performance problems.

The following example sets the number of **V**n files to 2,000:

```
/usr/vice/etc/afsd -files 2000
```

Note: It is conventional to place the **afsd** command in a machine's AFS initialization file, rather than entering it in a command shell. Furthermore, the values specified in this section are examples only, and are not necessarily suitable for a specific machine.

The default chunk size for a disk cache is 64 KB. In general, the only reason to change it is to adjust to exceptionally slow or fast networks; see "Setting Cache Configuration Parameters" on page 361. You can use the **-chunksize** argument to override the default. Chunk size must be a power of 2, so provide an integer between 0 (zero) and 30 to be used as an exponent of 2. For example, a value of 10 sets chunk size to 1 KB (210 = 1024); a value of 16 equals the default for disk caches (216 = 64 KB). Specifying a value of 0 (zero) or greater than 30 returns chunk size to the default. Values less than 10 (1 KB) are not recommended. The following example sets chunk size to 16 KB (214):

```
/usr/vice/etc/afsd -chunksize 14
```

For a disk cache, the default number of dcache entries duplicated in memory is one-half the number of chunks specified with the **-files** argument, to a maximum of 2,000 entries. You can use the **-dcache** argument to change the default, even exceeding 2,000 if you wish. Duplicating more than half the dcache entries in memory is not usually necessary, but sometimes improves performance slightly, because access to memory is faster than access to disk. The following example sets the number to 750:

```
/usr/vice/etc/afsd -dcache 750
```

When configuring a disk cache, you can combine the **afsd** command’s arguments in any way. The main reason for this flexibility is that the setting you specify for disk cache size (in the **cacheinfo** file or with the **-blocks** argument) is an absolute maximum limit. You cannot override it by specifying higher values for the **-files** or **-chunksize** arguments, alone or in combination. A related reason is that the Cache Manager does not have to reserve a set amount of memory on disk. **V**n files (the chunks in a disk cache) are initially zero-length, but can expand up to the specified chunk size and shrink again, as needed. If you set the number of **V**n files to such a large value that expanding all of them to the full allowable size exceeds the total cache size, they simply never grow to full size.

Controlling Memory Cache Configuration

Configuring a memory cache differs from configuring a disk cache in that not all combinations of the **afsd** command’s arguments are allowed. This limitation results from the greater interaction between the configuration parameters in a memory cache than a disk cache. If all combinations are allowed, it is possible to set the parameters in an inconsistent way. A list of the acceptable and unacceptable combinations follows a discussion of default values.

The default chunk size for a memory cache is 8 KB. In general, the only reason to change it is to adjust to exceptionally slow or fast networks; see "Setting Cache Configuration Parameters" on page 361.

There is no predefined default for number of chunks in a memory cache. The Cache Manager instead calculates the correct number by dividing the total cache size by the chunk size. Recall that for a memory cache, all dcache entries must be in memory. This implies that the number of chunks equals the number of dcache entries in memory, and that there is no default for number of dcache entries (like the number of chunks, it is calculated by dividing the total size by the chunk size).

The following are acceptable combinations of the **afsd** command’s arguments when configuring a memory cache:

- **-blocks** alone, which overrides the cache size specified in the **/usr/vice/etc/cacheinfo** file. The Cache Manager divides the value of this argument by the default chunk size of eight KB to calculate the number of chunks and dcache entries. The following example sets cache size to five MB (5,120 KB) and the number of chunks to 640 (5,120 divided by 8):

  ```
  /usr/vice/etc/afsd -memcache -blocks 5120
  ```

- **-chunksize** alone, to override the default of eight KB. The chunk size must be a power of two, so provide an integer between 0 (zero) and 30 to be used as an exponent of two. For example, a value of ten sets chunk size to 1 KB (210 = 1024); a value of 13 equals the default for memory caches (213 = 8 KB). Specifying a value of 0 (zero) or greater than 30 returns the chunk size to the default. Values less than ten (equivalent to 1 KB) are not recommended. The following example sets the chunk size to four KB (212). Assuming a total cache size of four MB (4,096 KB), the resulting number of chunks is 1024.

  ```
  /usr/vice/etc/afsd -memcache -chunksize 12
  ```

- **-blocks** and **-chunksize** together override the defaults for cache size and chunk size. The Cache Manager divides the first by the second to calculate the number of chunks and dcache entries. For example, the following example sets the cache size to six MB (6,144 KB) and chunksize to four KB (212), resulting in 1,536 chunks:

```
/usr/vice/etc/afsd -memcache -blocks 6144 -chunksize 12
```

The following arguments or combinations explicitly set the number of chunks and dcache entries. It is best not to use them, because they set the cache size indirectly, forcing you to perform a hand calculation to determine the size of the cache. Instead, set the **-blocks** and **-chunksize** arguments alone or in combination; in those cases, the Cache Manager determines the number of chunks and dcache entries itself. Because the following combinations are not recommended, no examples are included.

- The **-dcache** argument alone explicitly sets the number of chunks and dcache entries. The Cache Manager multiples this value times the default chunk size of 8 KB to derive the total cache size (overriding the value in the **cacheinfo** file).
- The combination of **-dcache** and **-chunksize** sets the chunk number and size. The Cache Manager sets the specified values and multiplies them together to obtain total cache size (overriding the value in the **cacheinfo** file).

Do not use the following arguments for a memory cache:

- **-files** alone. This argument controls the number of **V**n files for a disk cache, but is ignored for a memory cache.
- **-blocks** and **-dcache**. An error message results, because it is possible to provide values such that dividing the first (total size) by the second (number of chunks) results in a chunk size that is not a power of two.

Maintaining Knowledge of Database Server Machines

For the users of an AFS client machine to access a cell's AFS filespace and other services, the Cache Manager and other client-side agents must have an accurate list of the cell's database server machines. The affected functions include the following:

- Accessing files. The Cache Manager contacts the Volume Location (VL) Server to learn which file server machine houses the volume containing a requested file or directory. If the Cache Manager cannot contact a cell's VL Servers, it cannot fetch files.
- Authenticating. The **klog** program and AFS-modified login utilities contact the Authentication Server to obtain tokens, which the AFS server processes accept as proof that the user is authenticated.
- Creating protection groups. The **pts** command interpreter contacts the Protection Server when users create protection groups or request information from the Protection Database.
- Editing access control lists (ACLs). The **fs** command interpreter contacts the File Server that maintains the read/write volume containing a file or directory; the location information comes from the VL Server.

To enable a machine's users to access a cell, you must list the names and IP addresses of its database server machines in the **/usr/vice/etc/CellServDB** file on the machine's local disk. In addition to the machine's home cell, you can list any foreign cells that you want to enable users to access. (To enable access to a cell's filespace, you must also mount its **root.cell** volume in the local AFS filespace; the conventional location is just under the AFS root directory, **/afs**. For instructions, see the *IBM AFS Quick Beginnings*.)

How Clients Use the List of Database Server Machines

As the **afsd** program runs and initializes the Cache Manager, it reads the contents of the **CellServDB** file into kernel memory. The Cache Manager does not consult the file again until the machine next reboots. In contrast, the command interpreters for the AFS command suites (such as **fs** and **pts**) read the **CellServDB** file each time they need to contact a database server process.

When a cell's list of database server machines changes, you must change both the **CellServDB** file and the list in kernel memory to preserve consistent client performance; some commands probably fail if the two lists of machines disagree. One possible method for updating both the **CellServDB** file and kernel memory is to edit the file and reboot the machine. To avoid needing to reboot, you can instead perform both of the following steps:

1. Issue the **fs newcell** command to alter the list in kernel memory directly, making the changes available to the Cache Manager.
2. Edit the **CellServDB** file to make the changes available to command interpreters. For a description of the file's format, see "The Format of the CellServDB file" on page 365.

The consequences of missing or incorrect information in the **CellServDB** file or kernel memory are as follows:

- If there is no entry for a cell, the machine's users cannot access the cell.
- If a cell's entry does not include a database server machine, then the Cache Manager and command interpreters never attempt to contact the machine. The omission does not prevent access to the cell--as long as the information about the other database server machines is correct and the server processes, machines, and network are functioning correctly--but it can put an undue burden on the machines that are listed. If all of the listed machines become inaccessible to clients, then the cell becomes inaccessible even if the omitted database server machine is functioning correctly.
- If a machine's name or address is incorrect, or the machine is not actually running the database server processes, then requests from clients time out. Users can experience lengthy delays because they have to wait the full timeout period before the Cache Manager or command interpreter contacts another database server machine.

The Format of the CellServDB file

When editing the **/usr/vice/etc/CellServDB** file, you must use the correct format for cell and machine entries. Each cell has a separate entry. The first line has the following format:

```
>cell_name      #organization
```

where cell_name is the cell's complete Internet domain name (for example, **abc.com**) and organization is an optional field that follows any number of spaces and the number sign (#) and can name the organization to which the cell corresponds (for example, the ABC 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.abc.com**). In this case, the number sign does not indicate a comment: machine_name is a required field.

The order in which the cells appear is not important, but it is convenient to put the client machine's home cell first. Do not include any blank lines in the file, not even after the last entry.

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

```
>abc.com        #ABC Corporation (home cell)
192.12.105.3       #db1.abc.com
192.12.105.4       #db2.abc.com
192.12.105.55      #db3.abc.com
>stateu.edu     #State University cell
138.255.68.93      #serverA.stateu.edu
138.255.68.72      #serverB.stateu.edu
138.255.33.154     #serverC.stateu.edu
```

Maintaining the Client CellServDB File

Because a correct entry in the **CellServDB** file is vital for consistent client performance, you must also update the file on each client machine whenever a cell's list of database server machines changes (for instance, when you follow the instructions in the *IBM AFS Quick Beginnings* to add or remove a database server machine). To facilitate the client updates, you can use the **package** program, which copies files from a central source in AFS to the local disk of client machines. It is conventional to invoke the **package** program in a client machine's AFS initialization file so that it runs as the machine reboots, but you can also issue the **package** command at any time. For instructions, see "Running the package program" on page 407.

If you use the **package** program, the conventional location for your cell's central source **CellServDB** file is **/afs/**cell_name**/common/etc/CellServDB**, where cell_name is your cell name.

Creating a symbolic or hard link from **/usr/vice/etc/CellServDB** to a central source file in AFS is not a viable option. The **afsd** program reads the file into kernel memory before the Cache Manager is completely initialized and able to access AFS.

Because every client machine has its own copy of the **CellServDB** file, you can in theory make the set of accessible cells differ on various machines. In most cases, however, it is best to maintain consistency between the files on all client machines in the cell: differences between machines are particularly confusing if users commonly use a variety of machines rather than just one.

The AFS Product Support group maintains a central **CellServDB** file that includes all cells that have agreed to make their database server machines access to other AFS cells. It is advisable to check this file periodically for updated information. See "Making Your Cell Visible to Others" on page 22.

An entry in the local **CellServDB** is one of the two requirements for accessing a cell. The other is that the cell's **root.cell** volume is mounted in the local filespace, by convention as a subdirectory of the **/afs** directory. For instructions, see "To create a cellular mount point" on page 154.

> **Note:** The **/usr/vice/etc/CellServDB** file on a client machine is not the same as the **/usr/afs/etc/CellServDB** file on the local disk of a file server machine. The server version lists only the database server machines in the server machine's home cell, because server processes never need to contact foreign cells. It is important to update both types of **CellServDB** file on all machines in the cell whenever there is a change to your cell's database server machines. For more information about maintaining the server version of the **CellServDB** file, see "Maintaining the Server CellServDB File" on page 88.

To display the /usr/vice/etc/CellServDB file

1. Use a text editor or the **cat** command to display the contents of the **/usr/vice/etc/CellServDB** file. By default, the mode bits on the file permit anyone to read it.

   ```
   % cat /usr/vice/etc/CellServDB
   ```

To display the list of database server machines in kernel memory

1. Issue the **fs listcells** command.

   ```
   % fs listcells [&]
   ```

   where **listc** is the shortest acceptable abbreviation of **listcells**.

   To have your shell prompt return immediately, include the ampersand (**&**), which makes the command run in the background. It can take a while to generate the complete output because the kernel stores database server machines' IP addresses only, and the **fs** command interpreter has the cell's name resolution service (such as the Domain Name Service or a local host table) translate them into hostnames. You can halt the command at any time by issuing an interrupt signal such as **Ctrl-c**.

   The output includes a single line for each cell, in the following format:

   ```
   Cell cell_name on hosts list_of_hostnames.
   ```

   The name service sometimes returns hostnames in uppercase letters, and if it cannot resolve a name at all, it returns its IP address. The following example illustrates all three possibilities:

   ```
   % fs listcells
       .
       .
   Cell abc.com on hosts db1.abc.com db2.abc.com db3.abc.com
   Cell stateu.edu on hosts SERVERA.STATEU.EDU SERVERB.STATEU.EDU
   ```

```
                                 SERVERC.STATEU.EDU
Cell ghi.org on hosts 191.255.64.111 191.255.64.112
   .
   .
```

To change the list of a cell's database server machines in kernel memory

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. If you a use a central copy of the **CellServDB** file as a source for client machines, verify that its directory's ACL grants you the **l** (**lookup**), **r** (**read**), and **w** (**write**) permissions. The conventional directory is **/afs/**cell_name**/common/etc**. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   # fs listacl [<dir/file path>]
   ```

3. Issue the **fs newcell** command to add or change a cell's entry in kernel memory. Repeat the command for each cell.

   Note: You cannot use this command to remove a cell's entry completely from kernel memory. In the rare cases when you urgently need to prevent access to a specific cell, you must edit the **CellServDB** file and reboot the machine.

   ```
   # fs newcell <cell name> <primary servers>+ \
                 [-linkedcell <linked cell name>]
   ```

   where

   n

   Is the shortest acceptable abbreviation of **newcell**.

   cell name

   Specifies the complete Internet domain name of the cell for which to record a new list of database server machines.

   primary servers

   Specifies the fully-qualified hostname or IP address in dotted-decimal format for each database server machine in the cell. The list you provide completely replaces the existing list.

   -linkedcell

   Specifies the complete Internet domain name of the AFS cell to link to a DCE cell for the purposes of DFS fileset location. You can use this argument if the machine's AFS users access

DFS via the AFS/DFS Migration Toolkit Protocol Translator. For instructions, see the *IBM AFS/DFS Migration Toolkit Administration Guide and Reference*.

4. Add or edit the cell's entry in the local **/usr/vice/etc/CellServDB** file, using one of the following three methods. In each case, be sure to obey the formatting requirements described in "The Format of the CellServDB file" on page 365.
   - If you maintain a central source version of the **CellServDB** file and use the **package** program, first use a text editor to alter the central copy of the file. Then issue the **package** command to transfer the contents of the file to the local machine. For complete instructions, see "Running the package program" on page 407.

     ```
     # /etc/package -v -c <name of package file>
     ```

   - If you maintain a central source **CellServDB** file but do not use the **package** program, first use a text editor to alter the central copy of the file. Then use a copying command such as the **cp** command to copy it to the local **/usr/vice/etc/CellServDB** file.
   - If you do not use a central source **CellServDB** file, edit the local machine's **/usr/vice/etc/CellServDB** file directly.

Determining if a Client Can Run Setuid Programs

A *setuid program* is one whose binary file has the UNIX setuid mode bit turned on. While a setuid program runs, the user who initialized it assumes the local identity (UNIX UID) of the binary file's owner, and so is granted the permissions in the local file system that pertain to the owner. Most commonly, the issuer's assumed identity (often referred to as *effective UID*) is the local superuser **root**.

AFS does not recognize effective UID: if a setuid program accesses AFS files and directories, it uses the current AFS identity of the user who initialized the program, not of the program's owner. Nevertheless, it can be useful to store setuid programs in AFS for use on more than one client machine. AFS enables a client machine's administrator to determine whether the local Cache Manager allows setuid programs to run or not.

By default, the Cache Manager allows programs from its home cell to run with setuid permission, but denies setuid permission to programs from foreign cells. A program belongs to the same cell as the file server machine that houses the volume in which the file resides, as specified in the file server machine's **/usr/afs/etc/ThisCell** file. The Cache Manager determines its own home cell by reading the **/usr/vice/etc/ThisCell** file at initialization.

To change a cell's setuid status with respect to the local machine, become the local superuser **root** and issue the **fs setcell** command. To determine a cell's current setuid status, use the **fs getcellstatus** command.

When you issue the **fs setcell** command, you directly alter a cell's setuid status as recorded in kernel memory, so rebooting the machine is not necessary. However, nondefault settings do not persist across reboots of the machine unless you add the appropriate **fs setcell** command to the machine's AFS initialization file.

Only members of the **system:administrators** group can turn on the setuid mode bit on an AFS file or directory. When the setuid mode bit is turned on, the UNIX **ls -l** command displays the third user mode bit as an **s** instead of an **x**, but for an AFS file or directory, the **s** appears only if setuid permission is enabled for the cell in which the file resides.

To determine a cell's setuid status

1. Issue the **fs getcellstatus** command to check the setuid status of each desired cell.

   ```
   % fs getcellstatus <cell name>
   ```

   where

   getce

   Is the shortest acceptable abbreviation of **getcellstatus**.

   cell name

   Names each cell for which to report setuid status. Provide the complete Internet domain name or a shortened form that distinguishes it from the other cells listed in the local **/usr/vice/etc/CellServDB** file.

The output reports the setuid status of each cell:

- the string `no setuid allowed` indicates that the Cache Manager does not allow programs from the cell to run with `setuid permission`
- setuid allowed indicates that the Cache Manager allows programs from the cell to run with setuid permission

To change a cell's setuid status

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs setcell** command to change the setuid status of the cell.

   ```
   # fs setcell <cell name>+ [-suid] [-nosuid]
   ```

   where

   setce

   Is the shortest acceptable abbreviation of **setcell**.

cell name

Names each cell for which to change setuid status as specified by the **-suid** or **-nosuid** flag. Provide each cell's complete Internet domain name or a shortened form that distinguishes it from the other cells listed in the local **/usr/vice/etc/CellServDB** file.

-suid

Enables programs from each specified cell to execute with setuid permission. Provide this flag or the **-nosuid** flag, or omit both to disable setuid permission for each cell.

-nosuid

Prevents programs from each specified cell from executing with setuid permission. Provide this flag or the **-suid** flag, or omit both to disable setuid permission for each cell.

Setting the File Server Probe Interval

The Cache Manager periodically sends a probe to server machines to verify that they are still accessible. Specifically, it probes the database server machines in its cell and those file servers that house data it has cached.

If a server process does not respond to a probe, the client machine assumes that it is inaccessible. By default, the interval between probes is three minutes, so it can take up to three minutes for a client to recognize that a server process is once again accessible after it was inaccessible.

To adjust the probe interval, include the **-interval** argument to the **fs checkservers** command while logged in as the local superuser **root**. The new interval setting persists until you again issue the command or reboot the machine, at which time the setting returns to the default. To preserve a nondefault setting across reboots, include the appropriate **fs checkservers** command in the machine's AFS initialization file.

To set a client's file server probe interval

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs checkservers** command with the **-interval** argument.

```
# fs checkservers -interval <seconds between probes>
```

where

checks

Is the shortest acceptable abbreviation of **checkservers**.

-interval

Specifies the number of seconds between probes. Provide an integer value greater than zero.

Setting a Client Machine's Cell Membership

Each client machine belongs to a particular cell, as named in the **/usr/vice/etc/ThisCell** on its local disk. The machine's cell membership determines three defaults important to users of the machine:

- The cell for which users of the machine obtain tokens (authenticate) when they use the **login** program or issue the **klog** command. There are two effects:
  - The **klog** program and AFS-modified login utilities contact an Authentication Server in the cell named in the **ThisCell** file.
  - The **klog** program and AFS-modified login utilities combine the contents of the **ThisCell** file with the password that the user provides, generating an encryption key from the combination. The user's entry in the Authentication Database includes an encryption key also generated from the combination of password and cell name. If the cell name in the **ThisCell** file is incorrect, users cannot authenticate even if they provide the correct password.
- The cell the Cache Manager considers its local, or home, cell. The Cache Manager allows programs from its local cell to run with setuid permission, but not programs from foreign cells, as discussed further in "Determining if a Client Can Run Setuid Programs" on page 369.
- The default database server machines that are contacted by the AFS command interpreters running on this machine.

To display a client machine's cell membership

1. Use a text editor or the **cat** command to display the contents of the **/usr/vice/etc/ThisCell** file.

```
% cat /usr/vice/etc/ThisCell
```

To set a client machine's cell membership

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Using a text editor, replace the cell name in the **/usr/vice/etc/ThisCell** file.

3. **(Optional.)** Reboot the machine to enable the Cache Manager to use the new cell name immediately; the appropriate command depends on the machine's system type. The **klog** program, AFS-modified login utilities, and the AFS command interpreters use the new cell name the next time they are invoked; no reboot is necessary.

```
# sync
# shutdown
```

Forcing the Update of Cached Data

AFS's callback mechanism normally guarantees that the Cache Manager provides the most current version of a file or directory to the application programs running on its machine. However, you can force the Cache Manager to discard (flush) cached data so that the next time an application program requests it, the Cache Manager fetches the latest version available at the File Server.

You can control how many file system elements to flush at a time:

- To flush only specific files or directories, use the **fs flush** command. This command forces the Cache Manager to discard the data and status information it has cached from the specified files or directories. It does not discard information from an application program's buffer or information that has been altered locally (changes made in the cache but not yet saved permanently to the File Server). However, the next time an application requests the element's data or status information, the Cache Manager has to contact the File Server to get it.
- To flush everything cached from a certain volume, use the **fs flushvolume** command. This command works like the **fs flush** command, but differs in two ways:
  - The Cache Manager discards data for all elements in the cache that come from the same volume as the specified files or directories.
  - The Cache Manager discards only data, not status information. This difference has little practical effect, but can lead to different output from the **ls** command when the two different commands are used to flush the same element.

In addition to callbacks, the Cache Manager has a mechanism for tracking other kinds of possible changes, such as changes in a volume's location. If a volume moves and the Cache Manager has not accessed any data in it for a long time, the Cache Manager's volume location record can be wrong. To resynchronize it, use the **fs checkvolumes** command. When you issue the command, the Cache Manager creates a new table of mappings between volume names, ID numbers, and locations. This forces the Cache Manager to reference newly relocated and renamed volumes before it can provide data from them.

It is also possible for information about mount points to become corrupted in the cache. Symptoms of a corrupted mount point included garbled output from the **fs lsmount** command, and failed attempts to change directory to or list the contents of a mount point. Use the **fs flushmount** command to discard a corrupted mount point. The Cache Manager must refetch the mount point the next time it crosses it in a pathname. (The Cache Manager periodically refreshes cached mount points, but the only other way to discard them immediately is to reinitialize the Cache Manager by rebooting the machine.

To flush certain files or directories

1. Issue the **fs flush** command.

   ```
   % fs flush [<dir/file path>+]
   ```

   where

   flush

   Must be typed in full.

   dir/file path

   Names each file or directory structure to flush from the cache. Omit this argument to flush the current working directory. Flushing a directory structure does not flush any files or subdirectories cached from it.

To flush all data from a volume

1. Issue the **fs flushvolume** command.

   ```
   % fs flushvolume [<dir/file path>+]
   ```

   where

   flushv

   Is the shortest acceptable abbreviation of **flushvolume**.

   dir/file path

   Names a file or directory from each volume to flush from the cache. The Cache Manager flushes everything in the cache that it has fetched from the same volume. Omit this argument to flush all cached data fetched from the volume that contains the current working directory.

To force the Cache Manager to notice other volume changes

1. Issue the **fs checkvolumes** command.

   ```
   % fs checkvolumes
   ```

   where **checkv** is the shortest acceptable abbreviation of **checkvolumes**.

The following command confirms that the command completed successfully:

```
All volumeID/name mappings checked.
```

To flush one or more mount points

1. Issue the **fs flushmount** command.

   ```
   % fs flush [<dir/file path>+]
   ```

   where

   flushm

   Is the shortest acceptable abbreviation of **flushmount**.

   dir/file path

   Names each mount point to flush from the cache. Omit this argument to flush the current working directory. Files or subdirectories cached from the associated volume are unaffected.

Maintaining Server Preference Ranks

As mentioned in the introduction to this chapter, AFS uses client-side data caching and callbacks to reduce the amount of network traffic in your cell. The Cache Manager also tries to make its use of the network as efficient as possible by assigning *preference ranks* to server machines based on their network proximity to the local machine. The ranks bias the Cache Manager to fetch information from the server machines that are on its own subnetwork or network rather than on other networks, if possible. Reducing the network distance that data travels between client and server machine tends to reduce network traffic and speed the Cache Manager's delivery of data to applications.

The Cache Manager stores two separate sets of preference ranks in kernel memory. The first set of ranks applies to machines that run the Volume Location (VL) Server process, hereafter referred to as *VL Server machines*. The second set of ranks applies to machines that run the File Server process, hereafter referred to as *file server machines*. This section explains how the Cache Manager sets default ranks, how to use the **fs setserverprefs** command to change the defaults or set new ranks, and how to use the **fs getserverprefs** command to display the current set of ranks.

How the Cache Manager Sets Default Ranks

As the **afsd** program initializes the Cache Manager, it assigns a preference rank of 10,000 to each of the VL Server machines listed in the local **/usr/vice/etc/CellServDB** file. It then randomizes the ranks by adding an integer randomly chosen from the range 0 (zero) to 126. It avoids assigning the same rank to machines in one cell, but it is possible for machines from different cells to have the same rank. This does not present a problem in use, because the Cache Manager compares the ranks of only one cell's database server machines at a time. Although AFS supports the use of multihomed database server machines, the Cache Manager only uses the single address listed for each database server machine in the local **/usr/vice/etc/CellServDB** file. Only Ubik can take advantage of a multihomed database server machine's multiple interfaces.

The Cache Manager assigns preference ranks to a file server machine when it obtains the server's VLDB record from the VL Server, the first time that it accesses a volume that resides on the machine. If the machine is multihomed, the Cache Manager assigns a distinct rank to each of its interfaces (up to the number of interfaces that the VLDB can store for each machine, which is specified in the *IBM AFS Release Notes*). The Cache Manager compares the interface's IP address to the local machine's address and applies the following algorithm:

- If the local machine is a file server machine, the base rank for each of its interfaces is 5,000.
- If the file server machine interface is on the same subnetwork as the local machine, its base rank is 20,000.
- If the file server machine interface is on the same network as the local machine, or is at the distant end of a point-to-point link with the local machine, its base rank is 30,000.
- If the file server machine interface is on a different network than the local machine, or the Cache Manager cannot obtain network information about it, its base rank is 40,000.

If the client machine has only one interface, the Cache Manager compares it to the server interface's IP address and sets a rank according to the algorithm. If the client machine is multihomed, the Cache Manager compares each of the local interface addresses to the server interface, and assigns to the server interface the lowest rank that results from comparing it to all of the client interfaces.

After assigning a base rank to a file server machine interface, the Cache Manager adds to it a number randomly chosen from the range 0 (zero) to 15. As an example, a file server machine interface in the same subnetwork as the local machine receives a base rank of 20,000, but the Cache Manager records the actual rank as an integer between 20,000 and 20,015. This process reduces the number of interfaces that have exactly the same rank. As with VL Server machine ranks, it is possible for file server machine interfaces from foreign cells to have the same rank as interfaces in the local cell, but this does not present a problem. Only the relative ranks of the interfaces that house a specific volume are relevant, and AFS supports storage of a volume in only one cell at a time.

How the Cache Manager Uses Preference Ranks

Each preference rank pairs an interface's IP address with an integer that can range from 1 to 65,534. A lower rank (lower number) indicates a stronger preference. Once set, a rank persists until the machine reboots, or until you use the **fs setserverprefs** command to change it.

The Cache Manager uses VL Server machine ranks when it needs to fetch volume location information from a cell. It compares the ranks for the cell's VL Server machines and attempts to contact the VL Server process on the machine with the best (lowest integer) rank. If it cannot reach that VL Server, it tries to contact the VL Server with the next best rank, and so on. If all of a cell's VL Server machines are inaccessible, the Cache Manager cannot fetch data from the cell.

Similarly, when the Cache Manager needs to fetch data from a volume, it compares the ranks for the interfaces of machines that house the volume, and attempts to contact the interface that has the best rank. If it cannot reach the **fileserver** process via that interface, it tries to contact the interface with the next best integer rank, and so on. If it cannot reach any of the interfaces for machines that house the volume, it cannot fetch data from the volume.

Displaying and Setting Preference Ranks

To display the file server machine ranks that the Cache Manager is using, use the **fs getserverprefs** command. Include the **-vlservers** flag to display VL Server machine ranks instead. By default, the output appears on the standard output stream (stdout), but you can write it to a file instead by including the **-file** argument.

The Cache Manager stores IP addresses rather than hostnames in its kernel list of ranks, but by default the output identifies interfaces by hostname after calling a translation routine that refers to either the cell's name service (such as the Domain Name Server) or the local host table. If an IP address appears in this case, it is because the translation attempt failed. To bypass the translation step and display IP addresses rather than hostnames, include the **-numeric** flag. This can significantly speed up the output.

You can use the **fs setserverprefs** command to reset an existing preference rank, or to set the initial rank of a file server machine interface or VL Server machine for which the Cache Manager has no rank. The ranks you set persist until the machine reboots or until you issue the **fs setserverprefs** command again. To make a rank persist across a reboot, place the appropriate **fs setserverprefs** command in the machine's AFS initialization file.

As with default ranks, the Cache Manager adds a randomly chosen integer to each rank range that you assign. For file server machine interfaces, the randomizing number is from the range 0 (zero) to 15; for VL Server machines, it is from the range 0 (zero) to 126. For example, if you assign a rank of 15,000 to a file server machine interface, the Cache Manager stores an integer between 15,000 to 15,015.

To assign VL Server machine ranks, list them after the **-vlserver** argument to the **fs setserverprefs** command.

To assign file server machine ranks, use or more of the three possible methods:

1. List them after the **-servers** argument on the command line.
2. Record them in a file and name it with the **-file** argument. You can easily generate a file with the proper format by including the **-file** argument to the **fs getserverprefs** command.
3. Provide them via the standard input stream, by including the **-stdin** flag. This enables you to feed in values directly from a command or script that generates preferences using an algorithm appropriate for your cell. It must generate them in the proper format, with one or more spaces between each pair and between the two parts of the pair. The AFS distribution does not include such a script, so you must write one if you want to use this method.

You can combine any of the **-servers**, **-file**, and **-stdin** options on the same command line if you wish. If more than one of them specifies a rank for the same interface, the one assigned with the **-servers** argument takes precedence. You can also provide the **-vlservers** argument on the same command line to set VL Server machine ranks at the same time as file server machine ranks.

The **fs** command interpreter does not verify hostnames or IP addresses, and so willingly stores ranks for hostnames and addresses that don't actually exist. The Cache Manager never uses such ranks unless the same VLDB record for a server machine records the same incorrect information.

To display server preference ranks

1. Issue the **fs getserverprefs** command to display the Cache Manager's preference ranks for file server machines or VL Server machines.

   ```
   % fs getserverprefs [-file <output to named file>] [-numeric] [-vlservers]
   ```

   where

   gp

   Is an acceptable alias for **getserverprefs** (**gets** is the shortest acceptable abbreviation).

   -file

   Specifies the pathname of the file to which to write the list of ranks. Omit this argument to display the list on the standard output stream (stdout).

   -numeric

   Displays the IP address, rather than the hostname, of each ranked machine interface. Omit this flag to have the addresses translated into hostnames, which takes longer.

   -vlservers

   Displays ranks for VL Server machines rather than file server machines.

   The following example displays file server machine ranks. The **-numeric** flag is not used, so the appearance of an IP address indicates that is not currently possible to translate it to a hostname.

   ```
   % fs gp
   fs5.abc.com         20000
   fs1.abc.com         30014
   server1.stateu.edu  40011
   fs3.abc.com         20001
   fs4.abc.com         30001
   192.12.106.120      40002
   192.12.106.119      40001
      .    .    .    .    .      . .
   ```

To set server preference ranks

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

   ```
   % su root
   ```

```
   Password: <root_password>
```

2. Issue the **fs setserverprefs** command to set the Cache Manager's preference ranks for one or more file server machines or VL Server machines.

```
   # fs setserverprefs [-servers <fileserver names and ranks>+]  \
                       [-vlservers <VL server names and ranks>+]  \
                       [-file <input from named file>] [-stdin]
```

where

sp

Is an acceptable alias for **setserverprefs** (**sets** is the shortest acceptable abbreviation).

-servers

Specifies one or more pairs of file server machine interface and rank. Identify each interface by its fully-qualified hostname or IP address in dotted decimal format. Acceptable ranks are the integers from **1** to **65534**. Separate the parts of a pair, and the pairs from one another, with one or more spaces.

-vlservers

Specifies one or more pairs of VL Server machine and rank. Identify each machine by its fully-qualified hostname or IP address in dotted decimal format. Acceptable ranks are the integers from **1** to **65534**.

-file

Specifies the pathname of a file that contains one more pairs of file server machine interface and rank. Place each pair on its own line in the file. Use the same format for interfaces and ranks as with the **-servers** argument.

-stdin

Indicates that pairs of file server machine interface and rank are being provided via the standard input stream (stdin). The program or script that generates the pairs must format them in the same manner as for the **-servers** argument.

Managing Multihomed Client Machines

The File Server can choose the interface to which to send a message when it initiates communication with the Cache Manager on a multihomed client machine (one with more than one network interface and IP address). If that interface is inaccessible, it automatically switches to an alternate. This improves AFS performance, because it means that the outage of an interface does not interrupt communication between File Server and Cache Manager.

The File Server can choose the client interface when it sends two types of messages:

- A message to break the callback that the Cache Manager holds on a cached file
- A *ping* message to check that the Cache Manager is still accessible and responding; the File Server sends such a message every few minutes

(The File Server does not choose which client interface to respond to when filling a Cache Manager's request for AFS data. In that case, it always responds to the client interface via which the Cache Manager sent the request.)

The Cache Manager compiles the list of eligible interfaces on its client machine automatically as it initializes, and records them in kernel memory. When the Cache Manager first establishes a connection with the File Server, it sends along the list of interface addresses. The File Server records the addresses, and uses the one at the top of the list when it needs to break a callback or send a ping to the Cache Manager. If that interface is inaccessible, the File Server simultaneously sends a message to all of the other interfaces in the list. Whichever interface replies first is the one to which the File Server sends future messages.

You can control which addresses the Cache Manager registers with File Servers by listing them in two files in the **/usr/vice/etc** directory on the client machine's local disk: **NetInfo** and **NetRestrict**. If the **NetInfo** file exists when the Cache Manager initializes, the Cache Manager uses its contents as the basis for the list of interfaces. Otherwise, the Cache Manager uses the list of interfaces configured with the operating system. It then removes from the list any addresses that appear in the **/usr/vice/etc/NetRestrict** file, if it exists. The Cache Manager records the resulting list in kernel memory.

You can also use the **fs setclientaddrs** command to change the list of addresses stored in the Cache Manager's kernel memory, without rebooting the client machine. The list of addresses you provide on the command line completely replaces the current list in kernel memory. The changes you make persist only until the client machine reboots, however. To preserve the revised list across reboots, list the interfaces in the **NetInfo** file (and if appropriate, the **NetRestrict** file) in the local **/usr/vice/etc** directory. (You can also place the appropriate **fs setclientaddrs** command in the machine's AFS initialization script, but that is less efficient: by the time the Cache Manager reads the command in the script, it has already compiled a list of interfaces.)

To display the list of addresses that the Cache Manager is currently registering with File Servers, use the **fs getclientaddrs** command.

Keep the following in mind when you change the **NetInfo** or **NetRestrict** file, or issue the **fs getclientaddrs** or **fs setclientaddrs** commands:

- When you issue the **fs setclientaddrs** command, the revised list of addresses does not propagate automatically to File Servers with which the Cache Manager has already established a connection. They continue to use the list that the Cache Manager registered with them when it first established a connection. To force previously contacted File Servers to use the revised list, you must either reboot each file server machine, or reboot the client machine after changing its **NetInfo** file, **NetRestrict** file, or both.
- The **fs** command interpreter verifies that each of the addresses you specify on the **fs setclientaddrs** command line is actually configured with the client machine's operating system. If it is not, the command fails with an error message that marks the address as a `Nonexistent interface`.
- As previously noted, the File Server does not use the registered list of addresses when it responds to the Cache Manager's request for data (as opposed to initiating communication itself). It always

attempts to send its reply to the interface from which the Cache Manager sent the request. If the reply attempt fails, the File Server selects an alternate route for resending the reply according to its server machine's network routing configuration, not the list of addresses registered by the Cache Manager.

- The Cache Manager does not use the list of interfaces when choosing the interface via which to establish a connection to a File Server.
- The list of addresses that the **fs getclientaddrs** command displays is not necessarily the one that a specific File Server is using, if an administrator has issued the **fs setclientaddrs** command since the Cache Manager first contacted that File Server. It determines only which addresses the Cache Manager registers when connecting to File Servers in future.

To create or edit the client NetInfo file

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Using a text editor, open the **/usr/vice/etc/NetInfo** file. Place one IP address in dotted decimal format (for example, `192.12.107.33`) on each line. On the first line, put the address that you want each File Server to use initially. The order of the remaining machines does not matter, because if an RPC to the first interface fails, the File Server simultaneously sends RPCs to all of the other interfaces in the list. Whichever interface replies first is the one to which the File Server then sends pings and RPCs to break callbacks.
3. If you want the Cache Manager to start using the revised list immediately, either reboot the machine, or use the **fs setclientaddrs** command to create the same list of addresses in kernel memory directly.

To create or edit the client NetRestrict file

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Using a text editor, open the **/usr/vice/etc/NetRestrict** file. Place one IP address in dotted decimal format on each line. The order of the addresses is not significant. Use the value **255** as a wildcard that represents all possible addresses in that field. For example, the entry `192.12.105.255` indicates that the Cache Manager does not register any of the addresses in the 192.12.105 subnet.
3. If you want the Cache Manager to start using the revised list immediately, either reboot the machine, or use the **fs setclientaddrs** command to set a list of addresses that does not included the prohibited ones.

To display the list of addresses from kernel memory

1. Issue the **fs getclientaddrs** command.

   ```
   % fs getclientaddrs
   ```

   where **gc** is an acceptable alias for **getclientaddrs** (**getcl** is the shortest acceptable abbreviation).

The output lists each IP address on its own line, in dotted decimal format.

To set the list of addresses in kernel memory

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs setclientaddrs** command to replace the list of addresses currently in kernel memory with a new list.

   ```
   # fs setclientaddrs [-address <client network interfaces>+]
   ```

   where

   sc

   Is an acceptable alias for **setclientaddrs** (**setcl** is the shortest acceptable abbreviation).

   -address

   Specifies one or more IP addresses in dotted decimal format (hostnames are not acceptable). Separate each address with one or more spaces.

Controlling the Display of Warning and Informational Messages

By default, the Cache Manager generates two types of warning and informational messages:

- It sends *user messages*, which provide user-level status and warning information, to user screens.
- It sends *console messages*, which provide system-level status and warning information, to the client machine's designated console.

You can use the **fs messages** command to control whether the Cache Manager displays either type of message, both types, or neither. It is best not to disable messages completely, because they provide useful information.

If you want to monitor Cache Manager status and performance more actively, you can use the **afsmonitor** program to collect an extensive set of statistics (it also gathers File Server statistics). If you experience performance problems, you can use **fstrace** suite of commands to gather a low-level trace of Cache Manager operations, which the AFS Support and Development groups can analyze to help solve your problem. To learn about both utilities, see "Monitoring and Auditing AFS Performance" on page 295.

To control the display of warning and status messages

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs messages** command, using the **-show** argument to specify the type of messages to be displayed.

   ```
   # fs messages -show <user|console|all|none>
   ```

   where

   me

   Is the shortest acceptable abbreviation of **messages**.

   -show

   Specifies the types of messages to display. Choose one of the following values:

   user

   Sends user messages to user screens.

   console

   Sends console messages to the console.

   all

   Sends user messages to user screens and console messages to the console (the default if the **-show** argument is omitted).

   none

   Disables messages completely.

Displaying and Setting the System Type Name

The Cache Manager stores the system type name of the local client machine in kernel memory. It reads in the default value from a hardcoded definition in the AFS client software.

The Cache Manager uses the system name as a substitute for the @sys variable in AFS pathnames. The variable is useful when creating a symbolic link from the local disk to an AFS directory that houses binaries for the client machine's system type. Because the @sys variable automatically steers the Cache Manager to the appropriate directory, you can create the same symbolic link on client machines of different system types. (You can even automate the creation operation by using the package utility described in "Configuring Client Machines with the package Program" on page 389.) The link also remains valid when you upgrade the machine to a new system type.

Configuration is simplest if you use the system type names that AFS assigns. For a list, see the *IBM AFS Release Notes*.

To display the system name stored in kernel memory, use the **sys** or **fs sysname** command. To change the name, add the latter command's **-newsys** argument.

To display the system type name

1. Issue the **fs sysname** or **sys** command.

   ```
   % fs sysname
   % sys
   ```

The output of the **fs sysname** command has the following format:

```
Current sysname is 'system_name'
```

The **sys** command displays the system_name string with no other text.

To change the system type name

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs sysname** command, using the **-newsys** argument to specify the new name.

   ```
   # fs sysname <new sysname>
   ```

   where

   sys

   Is the shortest acceptable abbreviation of **sysname**.

   new sysname

   Specifies the new system type name.

Enabling Asynchronous Writes

By default, the Cache Manager writes all data to the File Server immediately and synchronously when an application program closes a file. That is, the **close** system call does not return until the Cache Manager has actually written all of the cached data from the file back to the File Server. You can enable the Cache Manager to write files asynchronously by specifying the number of kilobytes of a file that can remain to be written to the File Server when the Cache Manager returns control to the application.

Enabling asynchronous writes can be helpful to users who commonly work with very large files, because it usually means that the application appears to perform faster. However, it introduces some complications. It is best not to enable asynchronous writes unless the machine's users are sophisticated enough to understand the potential problems and how to avoid them. The complications include the following:

- In most cases, the Cache Manager returns control to applications earlier than it does by default, but it is not guaranteed to do so. Users cannot always expect faster performance.
- If an asynchronous write fails, there is no way to notify the application, because the **close** system call has already returned with a code indicating success.
- Asynchronous writing increases the possibility that the user fails to notice when a write operation makes a volume exceed its quota. As always, the portion of the file that exceeds the quota is lost, as indicated by a message like the following:

  ```
  No space left on device
  ```

  To avoid losing data because of insufficient quota, before closing a file users must verify that the volume housing the file has enough free space to accommodate it.

When you enable asynchronous writes by issuing the **fs storebehind** command, you set the number of kilobytes of a file that can still remain to be written to the File Server when the Cache Manager returns control to the application program. You can apply the setting either to all files manipulated by applications running on the machine, or only to certain files:

- The setting that applies to all files is called the *default store asynchrony* for the machine, and persists until the machine reboots. If, for example, you set the default store asynchrony to 10 KB, it means that when an application closes a file, the Cache Manager can return control to the application as soon as no more than 10 KB of a file that the application has closed remain to be written to the File Server.
- The setting for an individual file overrides the default store asynchrony and persists as long as there is an entry for the file in the internal table that the Cache Manager uses to track information about files. In general, such an entry persists at least until an application closes the file or exits completely, but the Cache Manager is free to recycle the entry if the file is inactive and it needs to free up slots in the

table. To be sure the entry exists in the table, issue the **fs storebehind** command shortly before closing the file.

To set the default store asynchrony

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs storebehind** command with the **-allfiles** argument.

   ```
   # fs storebehind -allfiles  <new default (KB)> [-verbose]
   ```

   where

   st

   Is the shortest acceptable abbreviation of **storebehind**.

   -allfiles

   Sets the number of kilobytes of data that can remain to be written to the File Server when the Cache Manager returns control to the application that closed a file.

   -verbose

   Produces a message that confirms the new setting.

To set the store asynchrony for one or more files

1. Verify that you have the **w** (**write**) permission on the access control list (ACL) of each file for which you are setting the store asynchrony, by issuing the **fs listacl** command, which is described fully in "Displaying ACLs" on page 519.

   ```
   % fs listacl dir/file path
   ```

   Alternatively, become the local superuser **root** on the client machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs storebehind** command with the **-kbytes** and **-files** arguments.

   ```
   # fs storebehind -kbytes <asynchrony for specified names> \
                    -files <specific pathnames>+  \
   ```

```
                          [-verbose]
```

where

st

Is the shortest acceptable abbreviation of **storebehind**.

-kbytes

Sets the number of kilobytes of data that can remain to be written to the File Server when the Cache Manager returns control to the application that closed a file named by the **-files** argument.

-files

Specifies each file for which to set a store asynchrony that overrides the default. Partial pathnames are interpreted relative to the current working directory.

-verbose

Produces a message that confirms that new setting.

To display the default store asynchrony

1. Issue the **fs storebehind** command with no arguments, or with the **-verbose** flag only.

```
% fs storebehind  [-verbose]
```

where

st

Is the shortest acceptable abbreviation of **storebehind**.

-verbose

Produces output that reports the default store asynchrony.

To display the store asynchrony for one or more files

1. Issue the **fs storebehind** command with the **-files** argument only.

```
% fs storebehind -files <specific pathnames>+
```

where

st

Is the shortest acceptable abbreviation of **storebehind**.

-files

Specifies each file for which to display the store asynchrony. Partial pathnames are interpreted relative to the current working directory.

The output lists each file separately. If a value has previously been set for the specified files, the output reports the following:

```
Will store up to y kbytes of file asynchronously.
Default store asynchrony is x kbytes.
```

If the default store asynchrony applies to a file (because you have not set a **-kbytes** value for it), the output reports the following:

```
Will store file according to default.
Default store asynchrony is x kbytes.
```

Chapter 11. Configuring Client Machines with the package Program

The **package** program automates many aspects of the client configuration process. With the **package** program, you can easily configure the local disk of numerous clients by defining global configuration files.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands or instructions in a prototype file:

Configure a client machine's local disk	**package**
Define directory	**D** [update_code] directory owner group mode_bits
Define file	**F** [update_code] file source_file [owner group mode_bits]
Define symbolic link	**L** [update_code] link actual_file [owner group mode_bits]
Define block special device	**B** device_name major_device_number minor_device_number owner group mode_bits
Define character special device	**C** device_name major_device_number minor_device_number owner group mode_bits
Define socket	**S** socket_name [owner group mode_bits]

Using the package Program

The **package** program uses system-independent *prototype files* to define a standard disk configuration; a prototype file indicates which files reside on the local client disk, which files are links into AFS, etc. The prototype files are then compiled into *configuration files* for each different system type.

Not all client machines have the same configuration. If desired, you can create different prototype files for different client functions (print server, regular client, etc.).

The **package** program compares the contents of a local client disk with the configuration file. If there are any differences, the **package** program makes the necessary updates to the local disk by copying the files from AFS onto the disk. The **package** program can also be configured to delete files that are not part of the system configuration or automatically reboot the client when certain files (such as the **dkload** file) have been updated.

The **package** program does require that you take some time to prepare the prototype files, but it provides the following benefits:

- You no longer need to configure each machine individually; the prototype configuration file applies to all machines.
- You can change the configuration of machines simply by changing the prototype file and rebooting the clients.
- Disk organization is uniform across a set of machines.

- The configuration files serve as a record of files on the disk and symbolic links into AFS.

Using Package on File Server Machines

While the **package** program was designed for use on client machines, it can also be used to configure a file server machine's disk. However, if any of the files referred to in a configuration file reside in volumes on the file server, the **package** program cannot access the volumes during reboot (and until the File Server process and Volume Server process start up again).

Since the **package** program aborts when it cannot access a file, you need to eliminate references to files in AFS that reside in volumes on the file server machine. Because of these constraints, the remainder of this chapter assumes the **package** program is being used for client configurations.

Package Overview

There are three main steps to follow before running the **package** program:

1. Preparing function-specific *prototype files* (and any included *library files*).
2. Modifying the **package Makefile** and compiling prototype files into system-specific *configuration files*.
3. Modifying client machines to run the appropriate **package** configuration file automatically.

The following sections summarize these steps.

Preparing Prototype Files

Begin by listing the different functions or roles client machines perform and the local disk configurations that support those functions. Example roles include a standard client that provides AFS access, a print server that drives a printer, and a backup machine on which you issue commands from the **backup** suite. Create a different *prototype file* for each role.

A prototype file defines the disk configuration that supports a specific role. Usually, prototype files are function-specific, but system independent; system-specific values can be defined using variables and library files. Then, when you modify a variable or library file, the change gets propagated to all appropriate clients when the **package** program is invoked.

Methods for building flexible prototype files that are easy to maintain are presented in "Example Prototype and Library Files" on page 392.

Compiling Prototype Files

Prototype files are usually system-independent, but can include `ifdef` statements to satisfy the needs of different system types. The prototype files are compiled to generate operating-system specific versions. During compilation, the **package** program selects the definitions suitable for each system type and replaces any variables with actual values. These compiled, machine-specific files are called *configuration files*.

Prototype files are compiled using a standard-type **Makefile** file, as described in "The Package Makefile File" on page 399.

Preparing Clients

Once system-specific configuration files exist, the **package** program is ready to run on the clients. You must first make the **package** binary available and specify the correct configuration file.

Modify the clients as described below:

1. Create a **.package** file in the root (**/**) directory of each client's local disk that defines the default configuration file.
2. Make the **package** binary (**/etc/package**) available on the local disk.
3. Modify the machine's initialization file (**/etc/rc** or equivalent) to include a call to the **package** program.

These steps are discussed more completely in "Modifying Client Machines" on page 406.

The package Directory Structure

This section assumes that the **package**-related files have been installed in three subdirectories of the **/afs/**cellname/**wsadmin** directory: **src**, **lib** and **etc**, as recommended in the *IBM AFS Quick Beginnings*.

These directories contain several sample prototype, library, and configuration files, which can help to clarify how the **package** program works. However, they are not necessarily suitable for use in your cell; you must modify them for your needs.

The src directory

The **src** directory contains some sample prototype files (used to build the configuration files), the **Makefile** file used to build them, and the resulting compiled configuration files.

Prototype files have names of the form function**.proto**. For example, a **minimal.proto** file defines the minimum set of library files need to run AFS and a**staff.dkload.proto** file defines a client configuration that uses the a dynamic kernel loading program. Prototype files can also contain definitions for system administrative files, such as a **hosts.equiv** file.

The **Makefile** file is used to compile the system-independent prototype files into system-specific configuration files. To learn how to modify this file for use in your cell, see "The Package Makefile File" on page 399.

Configuration files are the compiled version of the prototype files and are named function**.**sysname. Configuration files also appear in the **etc** subdirectory, which the **package** program accesses when configuring disks.

The lib directory

The **lib** directory contains many of the example library files referred to in prototype files. For example, the **base.generic** file is a system-independent file which includes a definition of the cell name, system options, and variables; these are used to set the owner, group, and mode_bits fields in the file and the symbolic link definitions.

The etc directory

The **etc** directory contains the system-specific configuration files built from the prototype files in the **src** subdirectory. The **package** program uses the configuration files in the **etc** directory to configure disks.

Some of the example files include **minimal** and **staff** prototype files compiled for different system types.

Example Prototype and Library Files

A prototype file is a template that defines the configuration of a client's local disk. Prototype files are usually function-specific (for example, a backup machine, print server, etc.) but system-independent. Prototype files support the use of `ifdef` statements and variables, so you can include system-specific definitions. The actual system-specific configuration file is generated when the prototype file is compiled.

The components defined in a prototype file can include the directories, files, symbolic links, block special devices, character special devices and sockets that need to reside on a client's local disk in order for it to perform a specific role, such as a print server or backup machine. Thus, we recommend that you construct a unique prototype file for each different client function.

To make the **package** program more effective and easy to maintain, create prototype files that are modular and generic, instead of specific, by using library files and variables:

- By creating general-purpose library files, you can include the same library file in many prototype files. Thus, you can make global configuration changes by modifying a single library file; you do not need to modify each prototype file.
- Variables enable you to change definitions simply by changing the variable's value.

An Example Prototype File

The following is part of an example prototype file that contains the minimum definitions necessary to run AFS. A similar file called **minimal.proto** can reside in your **src** subdirectory. As recommended, this prototype file references library files and does not include actual definitions.

```
          .
          .
# Package prototype for a minimal configuration.
# Base components
%include ${wsadmin}/lib/base.generic
# Machine-specific components
%ifdef rs_aix42
```

```
%include ${wsadmin}/lib/rs_aix42.readonly
%include ${wsadmin}/lib/rs_aix42.AFS
%endif rs_aix42
%ifdef alpha_dux40
%include ${wsadmin}/lib/alpha_dux40.readonly
%include ${wsadmin}/lib/alpha_dux40.AFS
%endif alpha_dux40
%ifdef sun4x_56
%include ${wsadmin}/lib/sun4x_56.readonly
%include ${wsadmin}/lib/sun4x_56.AFS
%endif sun4x_56
           .
           .
```

In the previous example, the first uncommented line includes the **/lib/base.generic** library file. This library file can contain definitions appropriate for many prototype files; the **base.generic** library file can also be included in other prototype files, like a **staff.proto** or **backup.proto** file. An example library file appears in the following section.

Note that system-specific definitions are permitted through the use of `ifdef` statements and variables (for example, `${wsadmin}` is used to specify pathnames). Thus, the same prototype file can be used to configure a machine running AIX 4.2 or Solaris 2.6, even though they require different files, directories, symbolic links and devices.

In the next uncommented lines of this example, the administrator has constructed different library files for different system types. Each of these is compiled into unique configuration files. For instance, the following lines in this prototype file tell the **package** program to use the library files **lib/rs_aix42.readonly** and **lib/rs_aix42.AFS** for the configuration file when the value rs_aix42 has been declared. (The system-type definition is declared in the **Makefile**; see "The Package Makefile File" on page 399.)

```
%ifdef rs_aix42
%include ${wsadmin}/lib/rs_aix42.readonly
%include ${wsadmin}/lib/rs_aix42.AFS
%endif rs_aix42
```

Similarly, the following lines tell the **package** program to use the library files **lib/sun4x_56.readonly** and **lib/sun4x_56.AFS** when the value sun4x_56 has been declared.

```
%ifdef sun4x_56
%include ${wsadmin}/lib/sun4x_56.readonly
%include ${wsadmin}/lib/sun4x_56.AFS
%endif sun4x_56
```

Example Library File

The following is part of an example library file for basic configuration definitions. A similar file, called **base.generic**, can reside in your **lib** subdirectory. Note that configurations are defined using standard `ifdef` statements.

```
           .
```

```
         .
#
# Base package definitions.
#
%ifndef      cell
%define      cell    abc.com
%endif       cell
%ifndef      sys
%include /etc/package.sys
%endif       sys
%define      ${name}         ${name}
%define      ${cpu}          ${cpu}
%define      ${sys}          ${sys}
%define      ${dept}         ${dept}
%define      ${hostname}     ${hostname}
%ifdef       rs_aix42
%    define  AIX
%    define  rootlinks
%ifndef      noafsd
%    define  afsd
%endif       noafsd
%endif       rs_aix42
         .
         .
#
# Some definitions to handle common combinations of owner, group,
# and protection fields.
#
%define      rzmode          root wheel 600
%define      usermode        root wheel 666
%define      systemmode      root wheel 644
%define      diskmode        root wheel 644
%define      ptymode         root wheel 666
%define      ttymode         root wheel 666
         .
         .
%define aix_rootbin     root bin
%define aix_rootprintq  root printq
%define aix_rootstaff   root staff
%define aix_rootsys     root system
%define aix_binbin      bin bin
%define aix_binmail     bin mail
%define aix_binsys      bin system
%define aix_addsys      adduser system
%define aix_romode      444
%define aix_loginmode   544
%define aix_usermode    666
%define aix_systemmode  644
%define aix_textmode    644
%define aix_rwmode1     660
%define aix_allrugw     664
```

The following example library file uses **package**-specific syntax to define files, directories, sockets, etc. Each line, called a *configuration file instruction*, defines a specific component of disk configuration. The proper syntax for these instructions is briefly described in "Package Configuration File Instruction Syntax" on page 395; see the reference page for the **package** configuration file in the *IBM AFS Administration Reference* for detailed descriptions.

In this example, the library file contains instructions specific to the configuration of an **rs_aix42** machine. You can have similar library files in your **lib** subdirectory.

```
         .
         .
#
# Generic configuration for an AFS rs_aix42 machine.
#
D    /                                     ${treemode}
D    /afs
FAQ  /unix           ${machine}/unix.std      ${binmode}
LA   /unix.std        /unix
D    /bin                                  ${treemode}
F    /bin/as          ${machine}             ${binmode}
F    /bin/ld          ${machine}             ${binmode}
F    /bin/nm          ${machine}             ${binmode}
FO   /bin/login       ${afstest}             ${suidmode}
         .
         .
FAQ  /usr/vice/etc/ThisCell  ${common}/etc/ThisCell ${textmode}
FQ   /usr/vice/etc/afsd      ${afstest}/root.client ${binmode}
FA   /usr/vice/etc/bos       ${afstest}/bin/bos     ${binmode}
FA   /usr/vice/etc/fs        ${afstest}/bin/fs      ${binmode}
```

Package Configuration File Instruction Syntax

Within a library file, configuration file instructions are used to define the specific disk configuration. Each instruction can be used to define a file, directory, socket, or device on the client machine. The syntax for each valid instruction type is described briefly here; detailed descriptions of the fields appear in the *IBM AFS Command Reference Manual*.

- **D** defines a directory
- **F** defines a file
- **L** defines a link
- **B** defines a block special device
- **C** defines a character special device
- **S** defines a socket

Note: Each configuration instruction must appear on a single, unbroken line. Instructions sometimes appear here on multiple lines only for legibility.

The configuration file must be completely correct. If there are any syntax errors or incorrect values, the **package** command interpreter exits without executing any instruction.

Local Files versus Symbolic Links

You can take advantage of the AFS by keeping the number of files on the local client disk to a minimum; instead, create symbolic links that point into AFS. This can improve machine performance by allowing more space for caching and swapping.

Some files, however, must reside on the local disk, as described below. Create these files in the prototype or library files using the **F** (file) instruction, not the **L** (symbolic link) instruction.

The following types of files must reside on the local disk of all AFS clients:

- Boot sequence files executed before the **afsd** program runs.

  Until **afsd** runs and initializes the Cache Manager, AFS is inaccessible from the client. Any files that are executed before the **afsd** program runs must reside on the local client disk.

  For example, on a machine that uses a disk cache, the **/usr/vice/cache** directory must exist when you bring up the Cache Manager, so that there is a location to create cache files. The binary files **/etc/mount** and **/etc/umount** must be available on the local disk as the machine boots in order to mount the **/usr/vice/cache** directory.

  In addition, certain UNIX files, such as initialization files (**/etc/rc** or equivalent) and file system mapping files (**/etc/fstab** or equivalent), must reside on the local disk.

- Diagnostic and recovery files

  Certain commands can be used to diagnose and recover from problems caused by a file server outage. It is best to keep copies of the binaries for these commands on the local disk. For example, store the **bos** and **fs** binaries in the **/usr/vice/etc** directory on the local disk, as well as in the **/usr/afsws** directory (which in the conventional configuration is a symbolic link into AFS). Then, set PATH variables so that the **/usr/afsws** directory appears before the **/usr/vice/etc** directory. Thus, even if users cannot access AFS (for example, due to a file server outage) they can still access copies of the **bos** and **fs** binaries in the **/usr/vice/etc** directory on the local disk.

- Files in the **/usr/vice** directory

  The contents of the **/usr/vice** directory, including the cache files in the **cache** subdirectory and the configuration files in the **etc** subdirectory, must reside on the local disk. For a description of the files in the directory, see "Configuration and Cache-Related Files on the Local Disk" on page 353.

Defining a Directory

The **D** instruction defines a directory to be created on the local disk. If a symbolic link, file, or other element on the local disk has the same name, it is replaced with a directory. If the directory already exists, its owner, group, and mode bits are changed if necessary to conform with the instruction.

Use the following instruction to define a directory:

```
D[update_code]   directory   owner   group   mode_bits
```

The following example defines the **/usr** directory:

```
   D /usr root wheel 755
```

Defining a File

The **F** instruction defines a file to be created on the local disk. The source file can reside in either AFS or the local disk.

If a file of this name already exists, then it is updated with (overwritten by) the source file, unless the **I** update code is specified. If a symbolic link or directory of this name exists, the **package** program replaces it with the source file.

> **Note:** Some files must reside on the local disk; they cannot be symbolic links. See "Local Files versus Symbolic Links" on page 396.

Use the following instruction to define a file:

```
F[update_code]   file   source_file  [owner   group   mode_bits]
```

An example which creates/updates the file **/bin/grep** on the local disk, using **/afs/abc.com/rs_aix42/bin/grep** as the source:

```
   F /bin/grep /afs/abc.com/rs_aix42 root wheel 755
```

In the following example, two update codes are used, and the *owner*, *group* and *mode_bits* slots are left empty, so that the disk file adopts the source file's values for those slots.

```
   FAQ /usr/vice/etc/ThisCell /afs/abc.com/common/etc/ThisCell
```

Defining a Symbolic Link

The **L** instruction defines a symbolic link to be created on the local disk. The symbolic link can point to the AFS file system or the local disk. If the identical symbolic link already exists, the **package** program does nothing. However, if an element of the same name exists on the disk as a file or directory, the **package** program replaces the element with a symbolic link.

> **Note:** Some files must reside on the local disk; they cannot be symbolic links. See "Local Files versus Symbolic Links" on page 396.

Use the following instruction to define a symbolic link:

```
L[update_code]  link actual_file  [owner   group   mode_bits]
```

> **Note:** Do not create a symbolic link to a file whose name begins with the number sign (**#**) or percent sign (**%**). The Cache Manager interprets such a link as a mount point to a regular or Read/Write volume, respectively.

The following example creates a symbolic link from the **/etc/ftpd** directory on the local disk to the **/afs/abc.com/hp_ux110/etc/ftpd** file in AFS. Since the *owner*, *group* and *mode_bits* fields are empty, the symbolic link adopts values for those fields from the actual file:

```
   L /etc/ftpd /afs/abc.com/hp_ux110
```

This example uses the **A** update code:

```
   LA /etc/printcap /afs/abc.com/common/etc/printcap.remote
               root wheel 644
```

Defining a Block Special Device

The **B** instruction defines a block special device, which is a device that handles data in units of multibyte blocks, such as a disk. If a device of the same name already exists, the **package** program replaces it with the specified block device.

Use the following instruction to define a block special device (it appears on two lines here only for legibility):

```
B device_name   major_device_number   minor_device_number  \
      owner   group   mode_bits
```

The following example defines a disk called **/dev/hd0a** to have major and minor device numbers **1** and **0**:

```
   B /dev/hd0a 1 0 root wheel 644
```

Defining a Character Special Device

The **C** instruction defines a character special device, which is device that handles data in units of a single character at a time, such as a terminal or tty. If a device of the same name already exists, the **package** program replaces it with the specified character device.

Use the following instruction to define a character special device (it appears here on two lines only for legibility):

```
C device_name   major_device_number   minor_device_number  \
      owner   group   mode_bits
```

The following example defines the tty called **/dev/ttyp5** with major and minor device numbers **6** and **5**:

```
C /dev/ttyp5 6 5 root wheel 666
```

Defining a Socket

The **S** instruction defines a socket, which is communications device for UDP and TCP/IP connections. If a socket of the same name already exists, the **package** program replaces it.

Use the following instruction to define a socket:

```
S   socket_name   [owner   group   mode_bits]
```

The following example defines a socket called **/dev/printer**:

```
S /dev/printer root wheel 777
```

Constructing Prototype and Library Files

This section describes the general steps required to create **package** prototype and library files. Refer to the previous sections for guidelines, and the files in your **wsadmin** directory for examples. The construction of prototype and library files is different for each cell.

To construct a prototype file and its component library files

1. Determine where the three **package**-related subdirectories (**src**, **lib** and **etc**) reside in your cell's file tree; the following instructions assume they were loaded into the **/afs/**cellname**/wsadmin** directory, as described in the IBM AFS Quick Beginnings.
2. Decide how many different functions you want client machines in your cell to perform. We recommend that you construct a separate prototype file for each function. Common functions include:
   - Standard workstation: provides users with access to files in AFS
   - Printer server: drives a printer; can be combined with "staff" functionality
   - Backup machine: performs backups of AFS volumes to tape by running the AFS Backup System software
3. Determine the minimum functionality needed for all clients (such as AFS setup) and place these generic definitions in one or more library files.
4. For each type of client (printer server, backup machine, and so on), place all system-independent definitions in one file, and all operating-system dependent definitions in another file.

The Package Makefile File

Once you have created the appropriate prototype and library files, you must compile the prototype for each system type. The result is a system-specific configuration file.

The **Makefile** file defines the prototype and library files used and the order of compilation. We recommend that you create your **Makefile** file by modifying the example provided with the AFS distribution, as described in this section. In the conventional configuration, it is located at **/afs/**cellname**/wsadmin/src/Makefile**.

Overview

The following list summarizes the sections in the **package Makefile** file, identifying each by the header name that begins the section. More detailed descriptions follow.

CONFIG=

Lists all of the configuration files to be created and defines which prototype files are compiled for which system types. See "The CONFIG Section" on page 400.

BASE_LIBS=

Lists the pathnames of all operating-system- and function independent library files included in any prototype files. See "The BASE_LIBS Section" on page 401.

MACHINE_LIBS=

Lists the pathnames of all operating-system-specific library files included in any prototype files. See "The MACHINE_LIBS Section" on page 402.

LIBS=

A one-line instruction that defines LIBS as the combination of BASE_LIBS and MACHINE_LIBS. See "The LIBS Section" on page 402.

.SUFFIXES

Defines all of the suffixes that can appear on a prototype or configuration file. See "The .SUFFIXES Section" on page 402.

Finally, the **Makefile** file contains a set of instructions that the **package** program follows to generate configuration files. It is not generally necessary to alter this section. See "The Makefile Instructions Section" on page 402.

The CONFIG Section

As mentioned, a configuration file is a prototype file that has been compiled for a specific operating system type. The CONFIG section of the **Makefile** file defines the prototype files to compile for each system type. The resulting compiled file is a system-specific configuration file.

Study the following example taken from the sample **Makefile** file. Configuration files are defined by specifying the prototype-system combination as prototype_file**.**sysname. Note that it is not necessary to generate a configuration file for each prototype-system type combination.

```
#Makefile...
#    (C) Copyright IBM Corporation 1999
#    Licensed Materials - Property of IBM
#    All Rights Reserved.
#
CONFIG = \
         staff.rs_aix42 \
         staff.alpha_dux40 \
         staff.xdm.alpha_dux40 \
         staff.sun4x_56 \
         staff.hp_ux110 \
         minimal.rs_aix42 \
         minimal.alpha_dux40 \
         minimal.hp_ux110 \
         minimal.sun4x_56
```

An entry in the CONFIG section has the following format:

- The first part of the entry defines the prototype file and is the same as the prototype file name (without the **.proto** extension). The second part of the entry indicates the system type for which the prototype file is to be compiled. A complete list of these suffixes is in the .SUFFIXES section of the **Makefile** file, as described in "The .SUFFIXES Section" on page 402. This prototype_file**.**sysname definition becomes the name of the compiled configuration file.

  For example, **staff.rs_aix42** indicates that the **staff.proto** file is compiled for machines running AIX 4.2. The resulting compiled configuration file is called **staff.rs_aix42**.

- Each configuration file must appear on a separate line.
- A backslash must follow the CONFIG= header and every name but the last one. A backslash must also appear on blank lines.

The BASE_LIBS Section

This section defines the complete pathname of all system- and function-independent library files included in any prototype file. (System-specific library files are defined in the MACHINE_LIBS section). The pathnames can include the ${wsadmin} variable, whose value is supplied on the **make** command line.

You must include all of the library files referred to in your prototype files; files included but not used are ignored.

Study the following example. Note that the all entries (except the last one) must be followed by a backslash.

```
BASE_LIBS = \
        ${wsadmin}/src/admin \
        ${wsadmin}/lib/devel \
        ${wsadmin}/lib/base.generic
```

The MACHINE_LIBS Section

This section lists the complete pathname of all operating-system-specific library files included in prototype files. (System- and function-independent library files are defined in the BASE_LIBS section.)

Study the following example. Note that in this example, library files were grouped by operating-system type. Again, all lines (except the last one) must be followed by a backslash, the ${wsadmin} variable is allowed, and files included but not used are ignored.

```
MACHINE_LIBS = \
        ${wsadmin}/lib/rs_aix42.generic \
        ${wsadmin}/lib/rs_aix42.generic.dev \
        ${wsadmin}/lib/rs_aix42.readonly \
        ${wsadmin}/lib/rs_aix42.readwrite \
        ${wsadmin}/lib/rt_aix42.generic.printer \
 \
 .
 .
        ${wsadmin}/lib/alpha_dux40.AFS \
        ${wsadmin}/lib/hp_ux110.AFS \
        ${wsadmin}/lib/sun4x_56.AFS \
        ${wsadmin}/lib/rs_aix42.AFS
```

The LIBS Section

This section contains only one instruction, which indicates that LIBS is defined as the combination of MACHINE_LIBS and BASE_LIBS. Insert a blank line after the line to separate this section from the next.

```
LIBS = ${MACHINE_LIBS} ${BASE_LIBS}
```

The .SUFFIXES Section

This section lists the valid machine-type suffixes. This list includes system types currently supported for AFS. Unused suffixes are ignored.

```
.SUFFIXES: .rs_aix42 \
           .alpha_dux40 \
           .proto \
           .sun4x_56 \
           .i386_linux22 \
           .hp_ux110
```

The Makefile Instructions Section

The remainder of the **Makefile** file controls how the **package** program generates configuration files.

Study the following instructions; it is assumed that you are familiar with programming and **Makefile** concepts.

```
#The following appear on a single line each in the actual file
.proto.rs_aix42: ;  mpp -Dwsadmin=${wsadmin} -Dsys=rs_aix42
                        -Dname=$* $*.proto > $@
.proto.alpha_dux40: ; mpp -Dwsadmin=${wsadmin} -Dsys=alpha_dux40
                        -Dname=$* $*.proto > $@
.proto.sun4x_56:  ; mpp -Dwsadmin=${wsadmin} -Dsys=sun4x_56
                        -Dname=$* $*.proto > $@
.proto.hp_ux110:  ; mpp -Dwsadmin=${wsadmin} -Dsys=hp_ux110
                        -Dname=$* $*.proto > $@
all: ${CONFIG}
${CONFIG}: ${LIBS}
system: install
install: ${CONFIG}
        cp ${CONFIG} ${wsadmin}/etc
clean:
        rm -f ${CONFIG} *.BAK *.CKP
```

Modifying the Makefile

Modify the **package Makefile** files when you

- Add a new prototype file (function**.proto**).
- Add a new system type.
- Add new library files.

The following sections provide brief examples of how to modify the **Makefile** file for these reasons.

Adding a New Prototype File

When you create a new prototype file, add the file name and each system type for which it is to be built into the CONFIG section of the **Makefile** file.

For example, to add a function**.proto** file for **alpha_dux40** and **hp_ux110**, add the following entries to the CONFIG section:

```
CONFIG = \
...
        function.alpha_dux40 \
        function.hp_ux110 \
...
```

If you have added new library files for this prototype function, add those to the MACHINE_LIBS section.

Adding a New System Type

For each prototype file that you want to build for the new system type, add an entry to the CONFIG section. Also add any new libraries to the MACHINE_LIBS section, and the new system type to the .SUFFIXES section.

The following example shows the modifications appropriate when building the **staff** and **minimal** prototype files for this new system type.

```
CONFIG = \
...
        staff.sysname \
        minimal.sysname \
...
```

If you have created corresponding library files for this new machine type, add them to the MACHINE_LIBS section.

```
MACHINE_LIBS = \
...
        ${wsadmin}/lib/sysname.generic \
        ${wsadmin}/lib/sysname.generic.dev \
        ${wsadmin}/lib/sysname.readonly \
        ${wsadmin}/lib/sysname.readwrite \
...
```

Add the new system type to the SUFFIXES section.

```
.SUFFIXES: ...\
         .sysname \
...
```

Add a line to build the configuration files for this system in the section with the rest of the commands to build configuration files:

```
.proto.sysname: ; mpp -Dwsadmin=${wsadmin} \
-Dsys=sysname  -Dname=$* $*.proto > $
```

Adding New Library Files

If you added a new library file for each system type, sysname.*library_file*, add these files to the MACHINE_LIBS section of the **Makefile**.

```
MACHINE_LIBS = \
...
        ${wsadmin}/lib/rs_aix42.library_file \
...
        ${wsadmin}/lib/alpha_dux40.library_file \
...
```

```
        ${wsadmin}/lib/sun4x_56.library_file \
...
```

If you added a new library file that is common to all system types, library_file, add this only to the BASE_LIBS section:

```
BASE_LIBS = \
...
        ${wsadmin}/lib/library_file \
...
```

Compiling Prototype Files

The **package** program generates configuration files and installs them in the **etc** and **src** subdirectories of the directory designated as **wsadmin=** on the **make** command line. Recompile whenever you modify a prototype or library file.

To compile prototype files

Note: These instructions assume that you store your **package**-related files in the **/afs**/cellname/**wsadmin** directory. If you use a different directory, substitute its name for **/afs**/cellname/**wsadmin**.

1. Verify that you have all privileges in the **/afs/**cellname**/wsadmin** directory and in its **src**, **lib** and **etc** subdirectories. If necessary, issue the **fs listacl** command.

   ```
   % fs listacl [dir/file path]
   ```

2. Change to the **/afs/**cellname**/wsadmin/src** subdirectory.

   ```
   % cd /afs/cellname/wsadmin/src
   ```

3. Create a backup copy of the **Makefile** file included in the AFS distribution.

   ```
   % cp  Makefile Makefile.example
   ```

4. Modify the CONFIG, BASE_LIBS and MACHINE_LIBS sections of the **Makefile** file, as described in "The CONFIG Section" on page 400, "The BASE_LIBS Section" on page 401, and "The MACHINE_LIBS Section" on page 402.
5. Compile the prototype files using the **make** command.

   Use the **wsadmin=** argument to specify the **package** directory. This becomes the value of the ${wsadmin} variable in the prototype and the library files.

   The **package** program generates configuration files and installs them in the **etc** and **src** subdirectories of the directory designated as **wsadmin=**.

```
% make system wsadmin=/afs/cellname/wsadmin
```

Modifying Client Machines

To prepare a client to run the **package** program automatically, perform the following steps. The instructions are generic because they do not refer to system-specific configuration files. If desired, you can invoke the **package** program with specific arguments, as described in the *IBM AFS Administration Reference*.

1. Specify the configuration file to use.

   The **.package** file in the client machine's root (**/**) directory is redirected as an argument to the **package** command; the **.package** file specifies which configuration file the **package** program uses.

2. Make the **package** binary available to the client, either by copying it to the local disk, or by creating a symbolic link to AFS.

   - A symbolic link saves local disk space. However, when the file server machine that houses it is down, the **package** binary is inaccessible.
   - Keeping the **package** binary on the local disk enables you to run the **package** program even if file server is down. However, a file server machine outage usually makes it difficult to run the **package** program because most configuration file instructions refer to files in AFS. A local copy of the **package** binary can be useful if the files referred to in instructions are in replicated volumes.

3. Modify the client machine's initialization file to invoke the **package** program at reboot. The client machine reboots a second time if the **package** program updates any files marked with the **Q** update code.

To prepare a client machine to run the package program

Repeat these instructions on every client that runs the **package** program.

These instructions assume that the **package** configuration files (created when the prototype files were compiled) reside in the **/afs/**cellname**/wsadmin/etc** directory.

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Create the **.package** file in the root (**/**) directory and specify the name of the prototype file to use. Do not include the system-type suffix (such as **.rs_aix42**); the **package** program automatically determines the correct machine type.

   ```
   # echo "/afs/cellname/wsadmin/etc/config_file" >> /.package
   ```

For example, to configure a machine for a member of staff machine (assuming the proper prototype file had been defined and compiled for the system type), the appropriate command is:

```
# echo "/afs/cellname/wsadmin/etc/staff" >> /.package
```

3. Make the **package** binary available on the local disk as **/etc/package**. Issue one of the following commands, depending on whether you want to create a file or create a symbolic link.

   To store the **package** binary locally, enter the following command:

   ```
   # cp /afs/cellname/sysname/usr/afsws/etc/package   /etc/package
   ```

   To create a symbolic link, enter the following command:

   ```
   # ln -s /afs/cellname/sysname/usr/afsws/etc/package   /etc/package
   ```

4. Add the following lines to the appropriate initialization file, after the **afsd** command is invoked. If this is a file server machine, the **bosserver** command must follow the **package** command.

   Using the **-v** and **-c** options is recommended. The **-v** flag produces a detailed trace, and the **-c** option appends the system type to the base name of the configuration file. See the *IBM AFS Administration Reference* for a description of other options.

   Note: Replace the **shutdown** command with a similar command if it is not appropriate for rebooting your machine.

   ```
   if [ -f /etc/package ]; then
           if [ -f /.package ]: then
                   /etc/package -v -c `cat /.package` >/dev/console
           else
                   /etc/package -v >/dev/console
   fi
   case $? in
   0)
           echo "Package completed successfully" >/dev/console 2>&1
           date >/dev/console 2>&1
           ;;
   4)
           echo "Rebooting to restart system" >/dev/console 2>&1
           echo >/fastboot
           shutdown
           ;;
   *)
           echo "Update failed, continuing anyway" >/dev/console 2>&1
           ;;
   esac
   fi
   ```

Running the package program

After you have created and compiled prototype files and modified client machines, you are ready to run the **package** program. It is probably most convenient to run the **package** program automatically at reboot by invoking it in the machine's AFS initialization file, but you can also issue the command at the command shell prompt.

The configuration file must be completely correct. If there are any syntax errors or incorrect values, the program exits without executing any instruction. To check the configuration file, issue the **package** command with the **-noaction** and **-debug** flags at the command shell prompt. They display a list of potential problems without actually executing instructions.

The **package** program follows these general rules. Complete explanations are in "Package Configuration File Instruction Syntax" on page 395.

- The **package** program does not delete any files from the disk unless the **R** update code was specified in the prototype file. If the **R** update code is associated with the parent directory, the **package** program removes anything from the local disk directory that is not specified in the configuration file.
- Local files are updated only if they are out of date. For each **F** instruction in the configuration file, the **package** program compares the time of the local file with the indicated source file. If the source file is newer than the local, the file is updated.
- When the initialization file is modified as recommended in "Modifying Client Machines" on page 406, the **package** program reboots the workstation automatically if any files marked with the **Q** update code are updated, and if the **package** program has been invoked from the initialization file. When a file marked with the **Q** update code is changed, the **package** program exits with status code 4, causing a reboot (as directed in the initialization file). Files that require a reboot before changes are recognized (such as the operating system kernel and **/usr/vice/etc/CellServDB** files) must be marked with a **Q** update code in the configuration file.
- The **package** program copies the configuration file it has just used to **/etc/package.**sysname, where sysname reflects this machine's system type. The **package** command interpreter consults this file if you do not provide a configuration file name. To be sure that it configures the local disk as you wish, review its contents.

To invoke the package program by rebooting

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. **(Recommended)** Verify the following:
   - The **/.package** file identifies the desired configuration file
   - The **package** binary is available as **/etc/package**
   - The initialization file is properly modified to invoke the **package** program automatically

3. Reboot the machine, using the appropriate command.

```
# shutdown
```

To invoke the package program directly (without rebooting)

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Verify the following:
   - The **/.package** file identifies the desired configuration file
   - The **package** binary is available as **/etc/package**
   - The initialization file is properly modified to invoke the **package** program automatically

3. Issue the **package** command.

   ```
   # package  [initcmd]  [-config <base name of configuration file>]  \
    [-fullconfig <full name of configuration file, or stdin for standard input>]  \
    [-overwrite]  [-noaction]  [-verbose]  [-silent] [-rebootfiles]
   ```

   where

   -config

   Specifies the full pathname of the configuration file to use, ending in the file’s base name, which omits the suffix that indicates the machine type. The **package** program knows how to determine a machine’s type, and automatically selects the appropriate version of the base file name. An example of the proper value for this argument is **staff** rather than **staff.rs_aix42**. You can also have the **package** program refer to **/.package** to learn the configuration file name by providing the following value:

   ‘cat /.package‘

   Use either this argument or the **-fullconfig** argument.

   -fullconfig

   Specifies the full name of the configuration file to use, complete with the machine-type extension. Examples are **staff.rs_aix42** and **minimal.hp_ux110** files.

   Another possibility is the string **stdin**, which indicates that the issuer is providing configuration information via the standard input stream, either as a piped file or by typing the configuration file at the keyboard. Press **<Ctrl-d>** to conclude the input.

   Use either this argument or the **-config** argument.

-overwrite

Overwrite elements on the local disk with the source version indicated in the configuration file, even if the first (owner) **w** (**write**) mode bit is turned off on the local disk copy of the file. Files protected by the **I** update code are not overwritten; see the definition for the **F** instruction.

-noaction

Displays on the standard output stream a trace of potential problems in running the command, rather than actually running it. If the **-verbose** flag is added, the trace also notes the actions the **package** program attempts.

-silent

Explicitly invokes the default level of tracing, which includes only a list of problems encountered while executing the command.

-verbose

Produces a detailed trace of the program's actions on the standard output stream. The trace records on the transfer and ownership/mode bit setting of each element in the configuration file.

-rebootfiles

Prevents the overwrite of any element marked with the **Q** update-mode code in the configuration file. This effectively prevents the machine from rebooting automatically again when the **package** program is invoked from an initialization file.

4. If you think files marked with the **Q** update code were updated, reboot the machine. This reboot does not occur automatically.

IV. Managing Users and Groups

Chapter 12. Creating and Deleting User Accounts with the uss Command Suite

The **uss** command suite helps you create and delete AFS user accounts quickly and easily. You can create a single account with the **uss add** command, delete a single account with the **uss delete** command, or create and delete multiple accounts with the **uss bulk** command.

A single **uss add** or **uss bulk** command can create a complete AFS user account because the **uss** command interpreter refers to a template file in which you predefine the configuration of many account components. The **uss delete** command deletes most of the components of a user account, but does not use a template file.

The **uss** suite also easily incorporates shell scripts or other programs that you write to perform parts of account creation and deletion unique to your site. To invoke a script or program automatically as a **uss** command runs, use the appropriate instructions in the template file or bulk input file. Various sections of this chapter discuss possible uses for scripts.

Using the **uss** commands to create and delete accounts is the recommended method because it automates and correctly orders most of the necessary steps. The alternative is to issue a series of separate commands to the various AFS servers, which requires more careful record keeping. For instructions, see "Administering User Accounts" on page 459.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Add a single user account	**uss add**
Delete a single user account	**uss delete**
Add and delete multiple accounts	**uss bulk**

Overview of the uss Command Suite

The commands in the **uss** suite help you to automate the creation and deletion of AFS user accounts:

- The **uss add** command creates all of the components of an account, one account at a time. It consults a template file that defines account configuration.
- The **uss delete** command deletes the major components of an account, one account at a time. It does not use a template file, so you possibly need to perform additional tasks manually.
- The **uss bulk** command can create and delete multiple accounts. It refers to a bulk input file that can contain any number of account-creation and deletion instructions, along with other instructions for further automating the process.

The Components of an AFS User Account

An AFS user account can have many components. The only two required components are entries in the Protection Database and Authentication Database, but the other components add functionality and usability. The following information also appears in a corresponding section of "Administering User Accounts" on page 459, but is repeated here for your convenience.

- A *Protection Database entry* defines the username (the name provided when authenticating with AFS), and maps it to an AFS user ID (AFS UID), a number that the AFS servers use internally when referencing users. The Protection Database also tracks the groups to which the user belongs. For details, see "Administering the Protection Database" on page 487.
- An *Authentication Database entry* records the user's AFS password in a scrambled form suitable for use as an encryption key.
- A home *volume* stores all the files in the user's home directory together on a single partition of a file server machine. The volume has an associated quota that limits its size. For a complete discussion of volumes, see "Managing Volumes" on page 131.
- A *mount point* makes the contents of the user's volume visible and accessible in the AFS filespace, and acts as the user's home directory. For more details about mount points, see "About Mounting Volumes" on page 134.
- Full access permissions on the home directory's *access control list (ACL)* and ownership of the directory (as displayed by the UNIX **ls -ld** command) enable the user to manage his or her files. For details on AFS file protection, see "Managing Access Control Lists" on page 513.
- A *local password file entry* (in the **/etc/passwd** file or equivalent) of each AFS client machine enables the user to log in and access AFS files through the Cache Manager. A subsequent section in this chapter further discusses local password file entries.
- Other optional *configuration files* make the account more convenient to use. Such files help the user log in and log out more easily, receive electronic mail, print, and so on.

Privilege Requirements for the uss Commands

To issue **uss** commands successfully, you usually need all of the standard AFS administrative privileges: membership in the **system:administrators** group, inclusion in the **/usr/afs/etc/UserList** file on every relevant server machine, and the `ADMIN` flag on your Authentication Database entry. For details on administrative privilege, see "Managing Administrative Privilege" on page 531.

Avoiding and Recovering from Errors and Interrupted Operations

As for any complex operation, there are a number of possible reasons that an account-creation or deletion operation can halt before it completes. You can easily avoid several of the common reasons by making the following checks before issuing a **uss** command:

- Verify that you have all of the administrative privileges you need to complete an operation, as described in "Privilege Requirements for the uss Commands" on page 414. The instructions for using the **uss add**, **uss delete**, and **uss bulk** commands include this check as a step.

- Proofread the template and bulk input files for correct syntax and acceptable values. For discussion, see "Constructing a uss Template File" on page 421 and "Constructing a Bulk Input File" on page 451.
- Do not issue **uss** commands when you are aware of network, server machine, or server process outages. Because **uss** operations affect so many components of AFS, it is unlikely that the command can succeed when there are outages.

Another way to avoid errors that halt an operation is to preview the **uss** command by combining the **-dryrun** flag with the other arguments to be used on the actual command. The **uss** command interpreter generates a screen trace of the actions to be performed by the actual command, without performing them.

Using the **-dryrun** flag reveals many basic errors that can halt an operation, particularly the ones due to incorrect syntax in the command line, template file, or bulk input file. It does not catch all possible errors, however, because the command interpreter is not actually attempting to perform the actions it is tracing. For example, a Volume Server outage does not necessarily halt the volume creation step when the **-dryrun** flag is included, because the command interpreter is not actually contacting the server; such an outage halts the actual creation operation.

When the **uss** command interpreter encounters error conditions minor enough that they do not require halting the operation, it usually generates a message that begins with the string `uss: Warning:` and describes the action it is taking to avoid halting. For example, if a user's Protection Database entry already exists, the following message appears on the standard output stream:

```
uss: Warning: User 'user' already in the protection database
The uid for user 'user' is AFS UID
```

If an error is more serious, the word `Warning` does not appear in the message, which instead describes why the command interpreter cannot perform the requested action. Not all of these errors cause the **uss** operation to halt, but they still require you to take corrective action. For example, attempting to create a mount point fails if you lack the necessary permissions on the parent directory's ACL, or if the mount point pathname in the **V** instruction's mount_point field is malformed. However, this error does not cause the creation operation to halt until later instructions in the template attempt to install subdirectories or files under the nonexistent mount point.

If the command shell prompts returns directly after an error message, then the error generally was serious enough to halt the operation. When an error halts account creation or deletion, the best way to recover is to find and fix the cause, and then reissue the same **uss** command.

The following list describes what happens when components of a user's account already exist when you reissue an account-creation command (the **uss add** command, or the **uss bulk** command when the bulk input file contains **add** instructions):

- If the Protection Database entry already exists, a message confirms its existence and specifies the associated AFS UID.
- If the Authentication Database entry already exists, a message confirms its existence.
- If the volume and associated Volume Location Database (VLDB) entry already exist, a message confirms their existence. However, the **uss** command interpreter does alter the volume's quota, mount point, or ACL if any of the relevant fields in the template **V** instruction have changed since the

command last ran. If the value in the mount_point field has changed, the command interpreter creates the new mount point but does not remove any existing mount points.

- If any of the fields in the template **A** instruction have changed, the **uss** command interpreter makes the changes without comment.
- If a directory, file, or link defined by a template file **D**, **E**, **F**, **L**, or **S** instruction already exists, the **uss** command interpreter replaces the existing element with one that conforms to the template definition. To control whether the **uss** command interpreter prompts for confirmation that you wish to overwrite a given element, use the **-overwrite** flag to the **uss add** or **uss bulk** command:
  - If you include the **-overwrite** flag, the command interpreter automatically overwrites all elements without asking for confirmation.
  - If you omit the flag, the command interpreter prompts once for each account to ask if you want to overwrite all elements associated with it.
- The command interpreter always reexecutes **X** instructions in the template file. If a command's result already holds, reissuing it has the same effect as reissuing it outside the context of the **uss** commands.

The following describes what happens when a **uss delete** command references account components that have already been deleted.

- If the volume and VLDB entry no longer exist, a message confirms their absence.
- If the Authentication Database entry no longer exists, a message confirms its absence.

Creating Local Password File Entries with uss

To obtain authenticated access to a cell's AFS filespace, a user must not only have a valid AFS token, but also an entry in the local password file (**/etc/passwd** or equivalent) of the AFS client machine. This section discusses why it is important for the user's AFS UID to match to the UNIX UID listed in the local password file, the appropriate value to put in the file's password field, and outlines a method for creating a single source password file.

For instructions on using the template file's **E** instruction to generate local password file entries automatically as part of account creation, see "Creating a Common Source Password File" on page 417.

The following information also appears in a corresponding section of "Administering User Accounts" on page 459, but is repeated here for your convenience.

Assigning AFS and UNIX UIDs that Match

A user account is easiest to administer and use if the AFS user ID number (AFS UID) and UNIX UID match. All instructions in the AFS documentation assume that they do.

The most basic reason to make AFS and UNIX UIDs the same is so that the owner name reported by the UNIX **ls -l** and **ls -ld** commands makes sense for AFS files and directories. Following standard UNIX practice, the File Server records a number rather than a username in an AFS file or directory's owner field: the owner's AFS UID. When you issue the **ls -l** command, it translates the UID to a username according to the mapping in the local password file, not the AFS Protection Database. If the AFS and UNIX UIDs do not match, the **ls -l** command reports an unexpected (and incorrect) owner. The output can even vary on different client machines if their local password files map the same UNIX UID to different names.

Follow the recommendations in the indicated sections to make AFS and UNIX UIDs match when you are creating accounts for various types of users:

- If creating an AFS account for a user who already has a UNIX UID, see "Converting Existing UNIX Accounts with uss" on page 419.
- If some users in your cell have existing UNIX accounts but the user for whom you are creating an AFS account does not, then it is best to allow the Protection Server to allocate an AFS UID automatically. To avoid overlap of AFS UIDs with existing UNIX UIDs, set the Protection Database's `max user id` counter higher than the largest UNIX UID, using the instructions in "Displaying and Setting the AFS UID and GID Counters" on page 509.
- If none of your users have existing UNIX accounts, allow the Protection Server to allocate AFS UIDs automatically, starting either at its default or at the value you have set for the `max user id` counter.

Specifying Passwords in the Local Password File

Authenticating with AFS is easiest for your users if you install and configure an AFS-modified login utility, which logs a user into the local file system and obtains an AFS token in one step. In this case, the local password file no longer controls a user's ability to login in most circumstances, because the AFS-modified login utility does not consult the local password file if the user provides the correct AFS password. You can nonetheless use a password file entry's password field (usually, the second field) in the following ways to control login and authentication:

- To prevent both local login and AFS authentication, place an asterisk (*) in the field. This is useful mainly in emergencies, when you want to prevent a certain user from logging into the machine.
- To prevent login to the local file system if the user does not provide the correct AFS password, place a character string of any length other than the standard thirteen characters in the field. This is appropriate if you want to allow only people with local AFS accounts to log into to your machines. A single **X** or other character is the most easily recognizable way to do this.
- To enable a user to log into the local file system even after providing an incorrect AFS password, record a standard UNIX encrypted password in the field by issuing the standard UNIX password-setting command (**passwd** or equivalent).

If you do not use an AFS-modified login utility, you must place a standard UNIX password in the local password file of every client machine the user will use. The user logs into the local file system only, and then must issue the **klog** command to authenticate with AFS. It is simplest if the passwords in the local password file and the Authentication Database are the same, but this is not required.

Creating a Common Source Password File

This section explains how to create a common source version of the local password file when using **uss** commands to create user accounts. The sequence of steps is as follows:

1. Include an **E** instruction in the template file to create a one-line file that has the format of a local password file entry.
2. Incorporate the one-line file into the common source version of the local password file. It makes sense to store this file in AFS. See the following two example scripts for automating this step.
3. Distribute the common password file to each client machine, perhaps by using the AFS **package** utility as described in "Configuring Client Machines with the package Program" on page 389.

As an example, the template file used by the ABC Corporation includes the following **E** instruction to create a file called **passwd_**username in the directory **/afs/.abc.com/common/etc/newaccts** (the entire contents of the template file appear in "Example uss Templates" on page 426 and a full description of the **E** instruction appears in "Creating One-Line Files with the E Instruction" on page 436):

```
E /afs/.abc.com/common/etc/newaccts/passwd_$USER 0644 root \
     "$USER:X:$UID:11:$NAME:$MTPT:/bin/csh"
```

For the user Joe L. Smith with username **smith**, this instruction creates a file called **passwd_smith** which contains the following line:

```
smith:X:1205:11:Joe L. Smith:/afs/abc.com/usr/usr1/smith:/bin/csh
```

A shell script is probably the easiest way to incorporate a set of files created in this manner into a common source password file, and two sample shell scripts appear here. To automate the process even further, you can create a **cron** process in a file server machine's **/usr/afs/local/BosConfig** directory to execute the shell script, perhaps each day at a given time; for details, see "To create and start a new process" on page 117.

Note: The following example scripts are suggestions only. If you choose to use them, or to model similar scripts on them, you must test that your script has the desired result, preferably in a test environment.

Example C Shell Script

The first example is a simple C shell script suitable for the ABC Corporation cell. It incorporates the individual files found in the **/afs/.abc.com/common/uss/newaccts** directory into a new version of the global password file found in the **/afs/.abc.com/common/etc** directory, sorting the files into alphabetical order. It takes care to save the current version with a **.old** extension, then removes the individual files when done.

```
set  dir = /afs/.abc.com/common
cat  $dir/uss/newaccts/passwd_* $dir/etc/passwd  >!  $dir/etc/passwd.new
mv  $dir/etc/passwd  $dir/etc/passwd.old
sort  $dir/etc/passwd.new  >  $dir/etc/passwd
rm  $dir/etc/passwd.new  $dir/uss/newaccts/passwd_*
```

Example Bourne Shell Script

The second, more elaborate, example is a Bourne shell script that first verifies that there are new **passwd_**username files to be incorporated into the global password file. While running, it checks that each new entry does not already exist. Like the shorter C shell example, it incorporates the individual files found in the **/afs/.abc.com/common/uss/newaccts** directory into a new version of the global **passwd** file found in the **/afs/.abc.com/common/etc** directory.

```
#!/bin/sh
DESTDIR=/afs/.abc.com/common/uss/newaccts
cd  $DESTDIR
DEST=/afs/.abc.com/common/etc
cp /afs/.abc.com/common/etc/passwd   /afs/.abc.com/common/uss/newaccts/passwd
echo "copied in passwd file."
PASSWD=/afs/.abc.com/common/uss/newaccts/passwd
ENTRIES=`ls passwd_*`
case $ENTRIES in
"")
     echo No new entry found to be added to passwd file
     ;;
*)
     echo  "Adding new users to passwd file."
     for  i  in  $ENTRIES
     do
        cat  $i  |  awk  -F:  '{print $1  >  "foo"}'
        USER=`cat foo`
        case  `egrep  -e  \^$USER\: $PASSWD` in
        "")
                echo  adding  $USER
                cat  $i  >>  $PASSWD
                ;;
        *)
                echo  $USER already in passwd file
                ;;
        esac
        mv  $i  ../old.passdir/done_${i}
     done
     cd  /afs/.abc.com/common/uss/newaccts
     echo  "sorting password file"
     sort  ${PASSWD}  >  ${PASSWD}.sorted
     echo  "installing files"
     install  ${PASSWD}.sorted ${DEST}/passwd
     echo  "Password file is built, sorted and installed."
     ;;
esac
```

Converting Existing UNIX Accounts with uss

This section discusses the three main issues you need to consider if there are existing UNIX accounts to be converted to AFS accounts.

Making UNIX and AFS UIDs Match

As previously mentioned, AFS users must have an entry in the local password file on every client machine from which they access the AFS filespace as an authenticated user. Both administration and use are much simpler if the UNIX UID and AFS UID match. When converting existing UNIX accounts, you have two alternatives:

- Make the AFS UIDs match the existing UNIX UIDs. In this case, you need to assign the AFS UID yourself as you create an AFS account:

  - If using the **uss add** command, include the **-uid** argument.
  - If using the **uss bulk** command, specify the desired UID in the uid field of the **add** instruction in the bulk input file.

  Because you are retaining the user's UNIX UID, you do not need to alter the UID in the local password file entry. However, if you are using an AFS-modified login utility, you possibly need to change the password field in the entry. For a discussion of how the value in the password field affects login with an AFS-modified login utility, see "Creating Local Password File Entries with uss" on page 416.

  If now or in the future you need to create AFS accounts for users who do not have an existing UNIX UID, then you must guarantee that new AFS UIDs do not conflict with any existing UNIX UIDs. The simplest way is to set the `max user id` counter in the Protection Database to a value higher than the largest existing UNIX UID. See "Displaying and Setting the AFS UID and GID Counters" on page 509.

- Change the existing UNIX UIDs to match the new AFS UIDs that the Protection Server assigns automatically.

  Allow the Protection Server to allocate the AFS UIDs automatically as you create AFS accounts. For instructions on creating a new entry for the local password file during account creation, see "Creating Local Password File Entries with uss" on page 416.

  There is one drawback to changing the UNIX UID: any files and directories that the user owned in the local file system before becoming an AFS user still have the former UID in their owner field. If you want the **ls -l** and **ls -ld** commands to display the correct owner, you must use the **chown** command to change the value to the user's new UID, whether you are leaving the file in the local file system or moving it to AFS. See "Moving Local Files into AFS" on page 421.

Setting the Password Field Appropriately

Existing UNIX accounts already have an entry in the local password file, probably with a (scrambled) password in the password field. You possibly need to change the value in the field, depending on the type of login utility you use:

- If the login utility is not modified for use with AFS, the actual password must appear (in scrambled form) in the password field of the local password file entry.
- If the login utility is modified for use with AFS, choose one of the acceptable values, each of which affects the login utility's behavior differently. See "Creating Local Password File Entries with uss" on page 416.

If you choose to place an actual password in a local password file entry, then you can define a dummy password when you use a template file **E** instruction to create the entry, as described in "Creating One-Line Files with the E Instruction" on page 436. Have the user issue the UNIX password-setting command (**passwd** or equivalent) to replace the dummy with an actual secret password.

Moving Local Files into AFS

New AFS users with existing UNIX accounts probably already own files and directories stored in a machine's local file system, and it usually makes sense to transfer them into the new home volume. The easiest method is to move them onto the local disk of an AFS client machine, and then use the UNIX **mv** command to transfer them into the user's new AFS home directory.

As you move files and directories into AFS, keep in mind that the meaning of their mode bits changes. AFS ignores the second and third sets of mode bits (group and other), and does not use the first set (the owner bits) directly, but only in conjunction with entries on the ACL (for details, see "How AFS Interprets the UNIX Mode Bits" on page 529). Be sure that the ACL protects the file or directory at least as securely as the mode bits.

If you have chosen to change a user's UNIX UID to match a new AFS UID, you must change the ownership of UNIX files and directories as well. Only members of the **system:administrators** group can issue the **chown** command on files and directories once they reside in AFS.

Constructing a uss Template File

Creating user accounts with **uss** commands is generally more convenient than using individual commands. You control the account creation process just as closely, but the **uss** template file enables you to predefine many aspects of account configuration. Because you construct the template before issuing **uss** commands, you have time to consider configuration details carefully and correct syntax errors. The following list summarizes some further advantages of using a template:

- You do not have to remember the correct order in which to create or delete account components, or the order of each command's arguments, which reduces the likelihood of errors.
- You do not have to type the same information multiple times. Instead, you can place constants and variables in the template file that enable you to type as little on the command line as possible. See "Using Constants and Variables in the Template File" on page 423.

- You can create different templates for different types of users. Instead of having to remember which components differ for a given user, specify the appropriate template when issuing the **uss add** or **uss bulk** command.
- You can create any of the three types of AFS account (authentication-only, basic, or full) by including or omitting certain information in the template, as described in "Creating the Three Types of User Accounts" on page 423.

The following list briefly describes the instructions that can appear in a template file and points you to a later section for more details. It lists them in the order that is usually optimal for correct handling of dependencies between the different types of instruction.

G

Defines a directory that is one of a set of parent directories into which the **uss** command interpreter evenly distributes newly created home directories. Place the corresponding template file variable, $AUTO, in the mount_point field of the **V** instruction. See "Evenly Distributing User Home Directories with the G Instruction" on page 428 and "Creating a Volume with the V Instruction" on page 429.

V

Creates a volume, mounts it as the user's home directory at a specified location in the AFS filespace, sets the volume's quota, and defines the owner and ACL for the directory. This instruction must appear in any template that is not empty (zero-length). See "Creating a Volume with the V Instruction" on page 429.

D

Creates a directory, generally a subdirectory of the new home directory, and sets its mode bits, owner, and ACL. See "Creating a Directory with the D Instruction" on page 432.

F

Creates a file by copying a prototype and sets its mode bits and owner. See "Creating a File from a Prototype with the F Instruction" on page 434.

E

Creates a single-line file by copying in the contents of the instruction itself, then sets the file's mode bits and owner. See "Creating One-Line Files with the E Instruction" on page 436.

L

Creates a hard link. See "Creating Links with the L and S Instructions" on page 438.

S

Creates a symbolic link. See "Creating Links with the L and S Instructions" on page 438.

A

Improves account security by imposing restrictions on passwords and authentication attempts. See "Increasing Account Security with the A Instruction" on page 439.

X

Executes a command. See "Executing Commands with the X Instruction" on page 441.

Creating the Three Types of User Accounts

Using the **uss add** and **uss bulk** commands, you can create three types of accounts that differ in their levels of functionality. For a description of the types, see "Configuring AFS User Accounts" on page 35. The following list explains how to construct a template for each type:

- To create an authentication-only account, create an empty (zero-length) template file. Such an account has only two components: entries in the Authentication Database and Protection Database.
- To create a basic account, include a **V** instruction, and **G** instructions if you want to distribute home directories evenly as described in "Evenly Distributing User Home Directories with the G Instruction" on page 428. In addition to Authentication Database and Protection Database entries, this type of account includes a volume mounted at the home directory with owner and ACL set appropriately.
- To create a full account, include **D**, **E**, **F**, **L**, and **S** instructions as appropriate, in addition to the **V** and **G** instructions. This type of account includes configuration files for basic functions such as logging in, printing, and mail delivery. For a discussion of some useful types of configuration files, see "Creating Standard Files in New AFS Accounts" on page 39.

Using Constants and Variables in the Template File

Each instruction in the **uss** template file has several fields that define the characteristics of the element that it creates. The **D** instruction's fields, for instance, define a directory's pathname, owner, mode bits, and ACL.

You can place three types of values in a field: a variable, a constant, or a combination of the two. The appropriate value depends on the desired configuration, and determines which arguments you provide to the **uss add** command or which fields you include in a bulk input file **add** instruction.

If an aspect of account configuration is the same for every user, define a constant value in the appropriate field by inserting a character string. For example, to assign a space quota of 10,000 KB to every user volume, place the string **10000** in the **V** instruction's quota field.

If, on the other hand, an aspect of account configuration varies for each user, put a variable in the appropriate field. When creating each account, provide a value for the variable by providing either the corresponding argument to the **uss add** command or a value in the corresponding field of the **add** instruction in the bulk input file.

The **uss** command suite defines a set of template variables, each of which has a corresponding source for its value, as summarized in "Table 3" on page 423. For a discussion of their intended uses, see the following sections about each template instruction ("Creating a Volume with the V Instruction" on page 429 through "Executing Commands with the X Instruction" on page 441).

Variable	Source for value
$AUTO	Previous **G** instructions in template
$MTPT	**-mount** argument to **uss add** command or mount_point field of bulk input file **add** instruction, when in **V** instruction; **V** instruction's mount_point field when in subsequent instructions
$NAME	**-realname** argument to **uss add** command or mount_point field of bulk input file **add** instruction, if provided; otherwise, **-user** argument to **uss add** command or username field of in bulk input file **add** instruction
$PART	**-partition** argument to **uss add** command or partition field of bulk input file **add** instruction
$PWEXPIRES	**-pwexpires** argument to **uss add** command or password_expires field of bulk input file **add** instruction
$SERVER	**-server** argument to **uss add** command or file_server field of bulk input file **add** instruction
$UID	**-uid** argument to **uss add** command or uid field of bulk input file **add** instruction, if provided; otherwise, allocated automatically by Protection Server
$USER	**-user** argument to **uss add** command or username field of bulk input file **add** instruction
$1 through $9	**-var** argument to **uss add** command or var1 through var9 fields of bulk input file **add** instruction

Table 3. Source for values of uss template variables

A common use of variables is to define the file server machine and partition that house the user's volume, which often vary from user to user. Place the $SERVER variable in the **V** instruction's server field, and the $PART variable in its partition field. If using the **uss add** command, provide the desired value with the **-server** and **-partition** arguments. If using the **uss bulk** command, provide the desired values in the file_server and partition fields of each user's **add** instruction in the bulk input file.

The variables $1 through $9 can be used to customize other aspects of the account. Provide a value for these variables with the **-var** argument to the **uss add** command or in the appropriate field of the bulk input file **add** instruction. The **-var** argument is unusual in that each instance for it has two parts: the number index and the value, separated by a space. For examples of the use of a number variable, see the discussions of the mount_point and quota fields in "Creating a Volume with the V Instruction" on page 429.

If some aspect of account configuration is partly constant and partly variable, you can combine variables and constants in an instruction field. For example, suppose that the ABC Corporation mounts user volumes in the **/afs/abc.com/usr** directory. That part of the pathname is constant, but the name of the mount point and home directory is the user's username, which corresponds to the $USER variable. To configure accounts in this way, combine a constant string and a variable in the **V** instruction's mount_point field as follows:

```
/afs/abc.com/usr/$USER
```

Then provide the value for the $USER variable with the **-user** argument to the **uss add** command, or in the username field of each user's **add** instruction in the bulk input file.

Where to Place Template Files

A template must be available to the **uss** command interpreter as it executes a **uss add** or **uss bulk** command, even if it is the zero-length file appropriate for creating an authentication-only account.

If you do not provide the **-template** argument to the **uss add** or **uss bulk** command, then the command interpreter searches for a template file called **uss.template** in each of the following directories in turn:

1. The current working directory
2. **/afs/cellname/common/uss**, where cellname is the local cell
3. **/etc**

To use a template file with a different name or stored in a different directory, include the **-template** argument to the **uss add** or **uss bulk** command. If you provide a filename only, the command interpreter looks for it in the directories listed just previously. If you provide a pathname and filename, it looks only in the specified directory, interpreting a partial pathname relative to the current working directory.

Some General Rules for Constructing a Template

This section summarizes some general rules to follow when constructing a template file. For each instruction's syntax definition, see the following sections ("Evenly Distributing User Home Directories with the G Instruction" on page 428 through "Executing Commands with the X Instruction" on page 441).

- If a variable takes its value from an element elsewhere within the template, the definition must precede the reference. Putting the instruction lines in the following order usually results in correct resolution of variables:

  G V D F E L S A X

- The fields in each instruction must appear in the order specified by the instruction's syntax definition, which appear in the following sections about each instruction. You cannot omit a field. Separate each field from its neighbors with one or more spaces.
- When specifying a pathname, provide a full one. Partial pathnames are interpreted relative to the current working directory (the one in which the **uss** command is issued), with possibly unintended results.
- Each instruction must appear on a single line in the template file, with a newline character (**<Return>**) only at the end of the instruction. Some example instructions appear in this document on more than one line, but that is only for legibility.
- Provide a value for every variable that appears in the template by including the corresponding argument to the **uss add** command or placing a value in the corresponding field of the bulk input file **add** instruction. A missing value halts the entire creation operation. If a variable does not appear in the template file, the command interpreter ignores the corresponding command-line argument or field in the bulk input file, even if you provide it.

- You can use blank lines in the template file to increase its legibility. If you place comments in the file, begin each comment line with the number sign (**#**).

About Creating Local Disk Directories and Files

It is possible to use the **D**, **E**, and **F** instructions to create directories or files in the local file system of the machine on which you are issuing the **uss** command, but that usage is not recommended. It introduces two potential complications:

- The local file system automatically assigns ownership of a new local disk directory or file to its creator. Because you are the issuer of the **uss** command that is creating the object, it records your current UNIX UID. If that is not appropriate and you want to designate another owner as the object is created, then you must be logged in as the local superuser **root** (the local file system allows only the **root** user to issue the UNIX **chown** command, which the **uss** command interpreter invokes to change the owner from the default value). You must also use the **-admin** argument to the **uss add** or **uss bulk** command to authenticate as a privileged AFS administrator. Only an administrator can create Authentication Database and Protection Database entries, which the **uss** command interpreter always creates as part of a new account.

  The alternative is to become the local superuser **root** after the **uss** operation completes, and issue the necessary **chown** command then. However, that makes the account creation process that much less automated.

- Creating a local disk directory always generates an error message because the **uss** command interpreter cannot successfully set a local directory's ACL. The directory is created nevertheless, and a value still must appear in the **D** instruction's ACL field.

The recommended method for configuring a machine's local disk is to use the AFS **package** utility instead; see "Configuring Client Machines with the package Program" on page 389.

Example uss Templates

This section describes example templates for the basic and full account types (the template for an authentication-only account is empty).

The first example creates a basic account. It contains two **G** instructions and a **V** instruction that defines the volume name, file server machine, partition, quota in kilobytes, mount point, home directory owner, and home directory access control list. In the ABC Corporation cell, a suitable template is:

```
G /afs/.abc.com/usr1
G /afs/.abc.com/usr2
V  user.$USER  $SERVER.abc.com  /vicep$PART  5000  $AUTO/$USER   $UID  \
     $USER all staff rl
```

When issuing the **uss add** command with this type of template, provide the following arguments:

- **-user** to specify the username for the $USER variable
- **-server** to specify the unique part of the file server machine name for the $SERVER variable
- **-partition** to specify the unique part of the partition name for the $PART variable

The Protection Server automatically assigns an AFS UID for the $UID variable, and the **G** instructions provide a value for the $AUTO variable.

The following example template file creates a full account in the ABC Corporation cell. The following sections about each type of instruction describe the effect of the examples. Note that the **V** and **E** instructions appear on two lines each only for the sake of legibility.

```
#
# Specify the available grouping directories
#
G /afs/.abc.com/usr1
G /afs/.abc.com/usr2
#
# Create the user's home volume
#
V user.$USER $SERVER.abc.com /vicep$PART 5000 /afs/.abc.com/$AUTO/$USER \
     $UID $USER all abc:staff rl
#
# Create directories and files for mail
#
D $MTPT/.MESSAGES 0700 $UID $USER all abc:staff none
D $MTPT/.Outgoing 0700 $UID $USER rlidwk postman rlidwk
D $MTPT/Mailbox 0700 $UID $USER all abc:staff none system:anyuser lik
#
# Here are some useful scripts for login etc.
#
F $MTPT/.Xbiff 0755 $UID /afs/abc.com/admin/user/proto
F $MTPT/.Xresources 0644 $UID /afs/abc.com/admin/user/proto
F $MTPT/.Xsession 0755 $UID /afs/abc.com/admin/user/proto
F $MTPT/.cshrc 0755 $UID /afs/abc.com/admin/user/proto
F $MTPT/.login 0755 $UID /afs/abc.com/admin/user/proto
F $MTPT/.logout 0755 $UID /afs/abc.com/admin/user/proto
F $MTPT/.twmrc 0644 $UID /afs/abc.com/admin/user/proto
F $MTPT/preferences 0644 $UID /afs/abc.com/admin/user/proto
#
# Make a passwd entry
#
E /afs/.abc.com/common/etc/newaccts/passwd_$USER 0644 root \
     "$USER:X:$UID:11:$NAME:$MTPT:/bin/csh"
#
# Put in the standard password/authentication checks
#
A $USER 250 noreuse 9 25
#
# Create and mount a public volume for the user
#
X "create_public_vol $USER $1 $2"
```

```
#
# Here we set up the symbolic link to public directory
#
S /afs/abc.com/public/$USER $MTPT/public
```

Evenly Distributing User Home Directories with the G Instruction

In cells with thousands of user accounts, it often makes sense to distribute the mount points for user volumes into multiple parent directories, because placing them all in one directory noticeably slows down directory lookup when a user home directory is accessed. A possible solution is to create parent directories that group user home directories alphabetically, or that reflect divisions like academic or corporate departments. However, in a really large cell, some such groups can still be large enough to slow directory lookup, and users who belong to those groups are unfairly penalized every time they access their home directory. Another drawback to groupings that reflect workplace divisions is that you must move mount points when users change departmental affiliation.

An alternative is an even distribution of user home directories into multiple parent directories that do not represent workplace divisions. The **uss** command suite enables you to define a list of directories by placing a **G** instruction for each one at the top of the template file, and then using the $AUTO variable in the **V** instruction's mount_point field. When the **uss** command interpreter encounters the $AUTO variable, it substitutes the directory named by a **G** instruction that currently has the fewest entries. (Actually, the $AUTO variable can appear in any field that includes a pathname, in any type of instruction. In all cases, the command interpreter substitutes the directory that currently has the fewest entries.)

The **G** instruction's syntax is as follows:

```
G  directory
```

where directory specifies either a complete directory pathname or only the final element (the directory itself). The choice determines the appropriate value to place in the **V** instruction's mount_point field.

Specify the read/write path to each directory, to avoid the failure that results when you attempt to create a new mount point in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

For example, the ABC Corporation example template for a full account in "Example uss Templates" on page 426 defines two directories:

```
G /afs/.abc.com/usr1
G /afs/.abc.com/usr2
```

and puts the value **$AUTO/$USER** in the **V** instruction's mount_point field. An alternative with the same result is to define the directories as follows:

```
G usr1
G usr2
```

and specify a more complete pathname in the **V** instruction's mount_point field: **/afs/.abc.com/$AUTO/$USER**.

Creating a Volume with the V Instruction

Unless the template file is empty (zero-length), one and only one **V** instruction must appear in it. (To create other volumes for a user as part of a **uss** account-creation operation, use the **X** instruction to invoke the **vos create** command or a script that invokes that command along with others, such as the **fs mkmount** command. For an example, see "Executing Commands with the X Instruction" on page 441.)

The **V** instruction defines the following AFS entities:

- A volume and associated VLDB entry
- The volume's site (file server machine and partition)
- The volume's mount point in the AFS filespace, which becomes the user's home directory
- The volume's space quota
- The home directory's owner, usually the new user
- The home directory's ACL, which normally at least grants all permissions to the user

The following discussion of the fields in a **V** instruction refers to the example in the full account template from "Example uss Templates" on page 426 (the instruction appears here on two lines only for legibility):

```
V  user.$USER  $SERVER.abc.com  /vicep$PART  5000  \
    /afs/.abc.com/$AUTO/$USER  $UID  $USER all abc:staff rl
```

The **V** instruction's syntax is as follows:

```
V  volume_name  server  partition  quota  mount_point owner  ACL
```

where

V

Indicates a volume creation instruction.

volume_name

Specifies the volume's name as recorded in the VLDB.

To follow the convention of including the user's name as part of the volume name, include the $USER variable in this field. The variable takes its value from the **-user** argument to the **uss add** command or from the bulk input file **add** instruction's username field.

The ABC Corporation example uses the value **user.$USER** to assign the conventional volume name, **user.**username. When creating an account for user **smith**, for example, you then include **-user smith** as an argument to the **uss add** command, or place the value **smith** in the bulk input file **add** instruction's username field.

server

Names the file server machine on which to create the new volume. It is best to provide a fully qualified host name (for example, **fs1.abc.com**), but an abbreviated form is acceptable if the cell's naming service is available to resolve it at the time the volume is created.

To place different users' volumes on different file server machines, use the $SERVER variable in this field, and provide a value for it either with the **-server** argument to the **uss add** command or in the server field of the bulk input file **add** instruction. One easy way to specify a fully qualified hostname without having to type it completely on the command line is to combine a constant and the $SERVER variable. Specifically, the constant specifies the domain-name suffix common to all the file server machines.

In the ABC Corporation example, all of the file server machines in the cell share the **abc.com** domain name suffix, so the server field combines a variable and constant: **$SERVER.abc.com**. To place the new volume on the machine **fs1.abc.com**, you then include **-server fs1** as an argument to the **uss add** command, or place the value **fs1** in the bulk input file **add** instruction's server field.

partition

Specifies the partition on which to create the user's volume; it must be on the file server machine named in the server field. Identify the partition by its complete name (for example, **/vicepa**) or use one of the abbreviations listed in "Rules for Using Abbreviations and Aliases" on page 558.

To place different users' volumes on different partitions, use the $PART variable in this field, and provide a value for it either with the **-partition** argument to the **uss add** command or in the partition field of the bulk input file **add** instruction. Because all full partition names start with the **/vicep** string, it is convenient to combine that string as a constant with the $PART variable.

The ABC Corporation example template combines the constant string **/vicep** and the $PART variable in this way, as **/vicep$PART**.

quota

Sets the maximum number of kilobyte blocks the volume can occupy on the file server machine's disk. It must be an integer. If you assign the same quota to all user volumes, specify a constant value. To assign different quotas to different volumes, place one of the number variables ($1 through $9) in this field, and provide a value for it either with the **-var** argument to the **uss add** command or in the appropriate field of the bulk input file **add** instruction.

The ABC Corporation example grants a 5000 KB initial quota to every new user.

mount_point

Creates a mount point for the volume, which serves as the volume's root directory and the user's home directory. By convention, user home directory names include the username, which you can read in by including the $USER variable in this field.

Specify the read/write path to the mount point, to avoid the failure that results when you attempt to create the new mount point in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). If you use the $AUTO variable in this field, the directories named by each **G** instruction possibly already indicate the read/write path. For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

If other parts of the mount point name also vary from user to user, you can use the $MTPT variable in this field, and provide a value with the **uss add** command's **-mount** argument or in the mount_point field of a bulk input file **add** instruction. Note, however, that when the $MTPT variable appears in subsequent instructions in the template (usually, in **D**, **E**, or **F** instructions), it instead takes as its value the complete contents of this field.

Combine constants and variables based on how you have decided to group home directories together in one or more parent directories. Note that the parent directories must already exist before you run a **uss add** or **uss bulk** command that references the template. Possibilities for grouping home directories include the following:

- Placing all user home directories in a single parent directory; the name **/afs/**cellname**/usr** is an AFS-appropriate variation on the UNIX **/usr** convention. This choice is most appropriate for a cell with a small number of user accounts. The simplest way to implement this choice is to combine a constant string and the $USER variable, as in **/afs/.abc.com/usr/$USER**.
- Distributing home directories evenly into a set of parent directories that do not correspond to workplace divisions. This choice is appropriate in cells with tens of thousands of accounts, where the number of home directories is large enough to slow directory lookup significantly if they all reside together in one parent directory, but distribution according to workplace divisions is not feasible.

  The $AUTO variable is designed to distribute home directories evenly in this manner. As explained in "Evenly Distributing User Home Directories with the G Instruction" on page 428, the **uss** command interpreter substitutes the directory that is defined by a preceding **G** template instruction and that currently has the fewest entries. The example ABC Corporation template illustrates this choice by using the value **/afs/.abc.com/$AUTO/$USER**.
- Distributing home directories into multiple directories that reflect divisions like academic or corporate departments. Perhaps the simplest way to implement this scheme is to use the $MTPT variable to represent the department, as in **/afs/.ghi.com/usr/$MTPT/$USER**. You then provide **-user smith** and **-mount acctg** arguments to the **uss add** command to create the mount point **/afs/.ghi.com/usr/acctg/smith**.
- Distributing home directories into alphabetic subdirectories of **usr** (**usr/a**, **usr/b** and so on), based on the first letter or letters in the username. The advantage is that knowing the username enables you easily to locate a home directory. A potential drawback is that the distribution is not likely to be even, and if there are a large number of accounts, then slowed directory lookup unfairly affects users whose names begins with popular letters.

  Perhaps the simplest way to implement this scheme is to use the $MTPT variable to represent the letter or letters, as in **/afs/.jkl.com/usr/$MTPT/$USER**. Then provide the **-user smith** and **-mount s/m** arguments to the **uss add** command to create the mount point **/afs/.jkl.com/usr/s/m/smith**.

owner

Specifies the username or UID of the user to be designated the mount point's owner in the output from the UNIX **ls -ld** command. To follow the standard convention for home directory ownership, use the $UID variable in this field, as in the ABC Corporation example template. The Protection Server then automatically assigns an AFS UID unless you provide the **-uid** argument to the **uss add** command or fill in the uid field in the bulk input file **add** instruction. (If you are converting existing UNIX accounts, see the discussion of additional considerations in "Converting Existing UNIX Accounts with uss" on page 419.)

ACL

Sets the ACL on the new home directory. Provide one or more paired values, each pair consisting of an AFS username or group name and the desired permissions, in that order (a group name must already exist in the Protection Database to be used). Separate the two parts of the pair, and each pair, with a space. For a discussion of the available permissions, see "The AFS ACL Permissions" on page 515.

At minimum, grant all permissions to the new user by including the value **$USER all** in this field. The File Server automatically grants all permissions to the **system:administrators** group as well. You cannot grant permissions to the issuer of the **uss** command, because as the last step in account creation the **uss** command interpreter automatically deletes that user from any ACLs set during the creation process.

The ABC Corporation example uses the following value to grant all permissions to the new user and **r** (**read**) and **l** (**lookup**) permissions to the members of the **abc:staff** group:

$USER all abc:staff rl

Creating a Directory with the D Instruction

Each **D** instruction in the template file creates a directory; there is no limit on the number of them in the template. If a **D** instruction creates a subdirectory in a new user's home directory (its intended use), then it must follow the **V** instruction. Creating a directory on the local disk of the machine where the **uss** command runs is not recommended for the reasons outlined in "About Creating Local Disk Directories and Files" on page 426.

The following discussion of the fields in a **D** instruction refers to one of the examples in the full account template in "Example uss Templates" on page 426:

```
D $MTPT/Mailbox 0700 $UID $USER all abc:staff none  system:anyuser lik
```

The **D** instruction's syntax is as follows:

```
D  pathname  mode_bits  owner  ACL
```

where

D

Indicates a directory creation instruction.

pathname

Specifies the directory's full pathname. If it is a subdirectory of the user's home directory, it is simplest to use the $MTPT variable to specify the home directory pathname. When the $MTPT variable appears in a **D** instruction, it takes its value from the preceding **V** instruction's mount_point field (this dependency is why a **D** instruction must follow the **V** instruction).

Specify the read/write pathname to the directory, to avoid the failure that results when you attempt to create a new directory in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). If you use the $MTPT variable in this field, the value in the **V** instruction's mount_point field possibly already indicates the read/write path. For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

The ABC Corporation example uses the value **$MTPT/Mailbox** to place the **Mailbox** subdirectory in the user's home directory.

mode_bits

Defines the directory's UNIX mode bits. Acceptable values are the standard three- or four-digit numbers corresponding to a combination of permissions. Examples: **0755** corresponds to **rwxr-xr-x**, and **0644** to **rw-r--r--**. The first (owner) **x** bit must be turned on to enable access to a directory.

The ABC Corporation example uses the value **0700** to set the mode bits on the **Mailbox** subdirectory to **rwxr-----**.

owner

Specifies the username or UID of the user to be designated the directory's owner in the output from the UNIX **ls -ld** command.

If the directory resides in AFS, place the $UID variable in this field, as in the ABC Corporation example template. The Protection Server then automatically assigns an AFS UID unless you provide the **-uid** argument to the **uss add** command or fill in the uid field in the bulk input file **add** instruction. (If you are converting existing UNIX accounts, see the discussion of additional considerations in "Converting Existing UNIX Accounts with uss" on page 419.)

If the directory resides on the local disk, it is simplest to specify the username or UNIX UID under which you are issuing the **uss** command. For a discussion of the complications that arise from designating another user, see "About Creating Local Disk Directories and Files" on page 426.

ACL

Sets the ACL on the new directory. Provide one or more paired values, each pair consisting of an AFS username or group name and the desired permissions, in that order (a group name must already exist in the Protection Database to be used). Separate the two parts of the pair, and each pair, with a space. For a description of the available permissions, see "The AFS ACL Permissions" on page 515.

At minimum, grant all permissions to the new user by including the value **$USER all**. You cannot grant permissions to the issuer of the **uss** command, because as the last step in account creation the **uss** command interpreter automatically deletes that user from any ACLs set during the creation process. An error message always appears if the directory is on the local disk, as detailed in "About Creating Local Disk Directories and Files" on page 426.

The ABC Corporation example uses the following value to grant all permissions to the new user, no permissions to the members of the **abc:staff** group, and the **l** (**lookup**), **i** (**insert**), and **k** (**lock**) permissions to the members of the **system:anyuser** group:

$USER all abc:staff none system:anyuser lik

It grants such extensive permissions to the **system:anyuser** group to enable any system user (including a mail-delivery daemon) to insert mail into the **Mailbox** directory. The absence of the **r** (**read**) permission prevents members of the **system:anyuser** group from reading the mail files.

Creating a File from a Prototype with the F Instruction

Each **F** instruction in the template file creates a file by copying the contents of an existing prototype file; there is no limit on the number of them in the template, and each can refer to a different prototype. If an **F** instruction creates a file in a new user's home directory or a subdirectory of it (the intended use), then it must follow the **V** or **D** instruction that creates the parent directory. Creating a file on the local disk of the machine where the **uss** command runs is not recommended for the reasons detailed in "About Creating Local Disk Directories and Files" on page 426.

The **E** instruction also creates a file, but the two types of instruction have complementary advantages. Files created with an **E** instruction can be customized for each user, because variables can appear in the field that specifies the contents of the file. In contrast, the contents of a file created using the **F** instruction are the same for every user. An **E** file can be only a single line, however, whereas an **F** file can be any length.

The following discussion of the fields in a **F** instruction refers to one of the examples in the full account template in "Example uss Templates" on page 426:

```
F $MTPT/.login 0755 $UID /afs/abc.com/admin/user/proto
```

The **F** instruction's syntax is as follows:

```
F  pathname  mode_bits  owner  prototype_file
```

where

F

Indicates a file creation instruction.

pathname

Specifies the full pathname of the file to create, including the filename. If it resides in the user's home directory or a subdirectory of it, it is simplest to use the $MTPT variable to specify the home directory pathname. When the $MTPT variable appears in an **F** instruction, it takes its value from the preceding **V** instruction's mount_point field (this dependency is why an **F** instruction must follow the **V** instruction).

Specify the read/write path to the file, to avoid the failure that results when you attempt to create a new file in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). If you use the $MTPT variable in this field, the value in the **V** instruction's mount_point field possibly already indicates the read/write path. For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

The ABC Corporation example uses the value **$MTPT/.login** to place a file called **.login** in the user's home directory.

mode_bits

Defines the file's UNIX mode bits. Acceptable values are the standard three- or four-digit numbers corresponding to a combination of permissions. Examples: **0755** corresponds to **rwxr-xr-x**, and **0644** to **rw-r--r--**.

The ABC Corporation example uses the value **0755** to set the mode bits on the **.login** file to **rwxr-xr-x**.

owner

Specifies the username or UID of the user to be designated the file's owner in the output from the UNIX **ls -l** command.

If the file resides in AFS, place the $UID variable in this field, as in the ABC Corporation example template. The Protection Server then automatically assigns an AFS UID unless you provide the **-uid** argument to the **uss add** command or fill in the uid field in the bulk input file **add** instruction. (If you are converting existing UNIX accounts, see the discussion of additional considerations in "Converting Existing UNIX Accounts with uss" on page 419.)

If the file resides on the local disk, it is simplest to specify the username or UNIX UID under which you are issuing the **uss** command. For a discussion of the complications that arise from designating another user, see "About Creating Local Disk Directories and Files" on page 426.

prototype_file

Names the AFS or local directory that houses the prototype file to copy. The prototype file's name must match the final element in the pathname field.

The ABC Corporation example references a prototype file called **.login** in the directory **/afs/abc.com/admin/user/proto**.

Creating One-Line Files with the E Instruction

Each **E** instruction in the template file creates a file by echoing a specified single line into it; there is no limit on the number of them in the template. If an **E** instruction creates a file in a new user's home directory or a subdirectory of it (the intended use), then it must follow the **V** or **D** instruction that creates the parent directory. Creating a file on the local disk of the machine where the **uss** command runs is not recommended for the reasons detailed in "About Creating Local Disk Directories and Files" on page 426.

The **F** instruction also creates a file, but the two types of instruction have complementary advantages. Files created with an **E** instruction can be customized for each user, because variables can appear in the field that specifies the contents of the file. The command interpreter replaces the variables with appropriate values before creating the file. In contrast, the contents of a file created using the **F** instruction are the same for every user. An **E** file can be only a single line, however, whereas an **F** file can be any length.

The **E** instruction is particularly suited to creating an entry for the new user in the cell's common source password file, which is then copied to client machines to serve as the local password file (**/etc/passwd** or equivalent). The following discussion of the fields refers to an example of this type of use, from the ABC Corporation's full account template shown in "Example uss Templates" on page 426. For further discussion of how to incorporate the files created in this way into a common source password file, see "Creating a Common Source Password File" on page 417.

```
E /afs/.abc.com/common/etc/newaccts/passwd_$USER 0644 root  \
   "$USER:X:$UID:11:$NAME:$MTPT:/bin/csh"
```

The **E** instruction's syntax is as follows:

```
E  pathname  mode_bits  owner  "contents"
```

where

E

Indicates a file creation instruction.

pathname

Specifies the full pathname of the file to create, including the filename. It can include variables. If it resides in the user's home directory or a subdirectory of it, it is simplest to use the $MTPT variable to specify the home directory pathname. When the $MTPT variable appears in an **E** instruction, it takes its value from the preceding **V** instruction's mount_point field (this dependency is why an **E** instruction must follow the **V** instruction.)

Specify the read/write path to the file, to avoid the failure that results when you attempt to create a new file in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). If you use the $MTPT variable in this field, the value in the **V** instruction's mount_point field possibly already indicates the read/write path. For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

The ABC Corporation example writes the file created by the **E** instruction to **/afs/.abc.com/common/etc/newaccts** directory, naming it after the new user:

```
/afs/.abc.com/common/etc/newaccts/passwd_$USER
```

mode_bits

Defines the file's UNIX mode bits. Acceptable values are the standard three- or four-digit numbers corresponding to a combination of permissions. Examples: **0755** corresponds to **rwxr-xr-x**, and **0644** to **rw-r--r--**.

The ABC Corporation example uses the value **0644** to set the mode bits on the **passwd_**user file to **r-xr--r--**.

owner

Specifies the username or UID of the user to be designated the file's owner in the output from the UNIX **ls -l** command.

If the file resides in AFS and is to be owned by the user, place the $UID variable in this field. The Protection Server then automatically assigns an AFS UID unless you provide the **-uid** argument to the **uss add** command or fill in the uid field in the bulk input file **add** instruction. (If you are converting existing UNIX accounts, see the discussion of additional considerations in "Converting Existing UNIX Accounts with uss" on page 419.)

If the file resides on the local disk, specify the username or UNIX UID under which you are issuing the **uss** command. For a discussion of the complications that arise from designating another user, see "About Creating Local Disk Directories and Files" on page 426.

The ABC Corporation example is creating an AFS file intended for incorporation into the common password file, rather than for direct use by the new user. It therefore designates the local superuser **root** as the owner of the new file. Designating an alternate owner on an AFS file does not introduce complications: issuing the **chown** command on AFS files requires membership in the **system:administrators** group, but the issuer of the **uss** command is necessarily authenticated as a member of that group.

contents

Specifies the one-line character string to write into the new file. Surround it with double quotes if it contains one or more spaces. It cannot contain the newline character, but can contain any of the standard variables, which the command interpreter resolves as it creates the file.

The ABC Corporation example has the following value in the contents field, to create a password file entry:

```
$USER:X:$UID:10:$NAME:$MTPT:/bin/csh
```

Creating Links with the L and S Instructions

Each **L** instruction in the template file creates a hard link between two files, as achieved by the standard UNIX **ln** command. The **S** instruction creates a symbolic link between two files, as achieved by the standard UNIX **ln -s** command. An explanation of links is beyond the scope of this document, but the basic effect in both cases is to create a second name for an existing file, so that it can be accessed via either name. Creating a link does not create a second copy of the file.

There is no limit on the number of **L** or **S** instructions in a template file. If the link is in a new user's home directory or a subdirectory of it (the intended use), then it must follow the **V** or **D** instruction that creates the parent directory, and the **F**, **E**, or **X** instruction that creates the file being linked to. Creating a file on the local disk of the machine where the **uss** command runs is not recommended, for the reasons detailed in "About Creating Local Disk Directories and Files" on page 426.

Note that AFS allows hard links only between files that reside in the same directory. This restriction is necessary to eliminate the confusion that results from associating two potentially different ACLs (those of the two directories) with the same file. Symbolic links are legal between two files that reside in different directories and even in different volumes. The ACL on the actual file applies to the link as well.

You do not set the owner or mode bits on a link created with an **L** or **S** instruction, as you do for directories or files. The **uss** command interpreter automatically records the UNIX UID of the **uss** command's issuer as the owner, and sets the mode bits to **lrwxrwxrwx** (777).

The following discussion of the fields in an **L** or **S** instruction refers to an example in the full account template from "Example uss Templates" on page 426, namely

```
S /afs/abc.com/public/$USER $MTPT/public
```

The **L** and **S** instructions' syntax is as follows:

```
L  existing_file  link
S  existing_file  link
```

where

L

Indicates a hard link creation instruction.

S

Indicates a symbolic link creation instruction.

existing_file

Specifies the complete pathname of the existing file. If it resides in the user's home directory or a subdirectory of it, it is simplest to use the $MTPT variable to specify the home directory pathname. When the $MTPT variable appears in an **L** or **S** instruction, it takes its value from the preceding **V** instruction's mount_point field (this dependency is why the instruction must follow the **V** instruction).

Do not create a symbolic link to a file whose name begins with the number sign (**#**) or percent sign (**%**). When the Cache Manager reads a symbolic link whose contents begin with one of those characters, it interprets it as a regular or read/write mount point, respectively.

The ABC Corporation example creates a link to the publicly readable volume created and mounted by a preceding **X** instruction, by specifying the path to its mount point:

```
/afs/abc.com/public/$USER
```

link

Specifies the complete pathname of the second name for the file. If it resides in the user's home directory or a subdirectory of it, it is simplest to use the $MTPT variable to specify the home directory pathname.

Specify the read/write path to the link, to avoid the failure that results when you attempt to create a new link in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). If you use the $MTPT variable in this field, the value in the **V** instruction's mount_point field possibly already indicates the read/write path. For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

The ABC Corporation example creates a link called **public** in the user's home directory:

```
$MTPT/public
```

Increasing Account Security with the A Instruction

The **A** instruction in the template file enhances cell security by imposing the following restrictions on users' password choice and authentication attempts.

- Limiting the user's password lifetime. When the lifetime expires, the user can no longer use the password to authenticate and must change it.
- Prohibiting the reuse of the user's 20 most-recently used passwords.
- Limiting the number of consecutive times that a user can provide an incorrect password during authentication, and for how long the Authentication Server refuses further authentication attempts after the limit is exceeded (referred to as an *account lockout*). For regular user accounts in most cells, the recommended limit is nine and lockout time is 25 minutes.

The following discussion of the fields in an **A** instruction refers to the example in the full account template from "Example uss Templates" on page 426, which sets a password lifetime of 250 days, prohibits reuse of passwords, limits the number of failed authentication attempts to nine, and creates a lockout time of 25 minutes if the authentication limit is exceeded:

```
A $USER 250 noreuse 9 25
```

The **A** instruction's syntax is as follows:

```
A  username  password_lifetime  password_reuse  failures  locktime
```

where

A

Indicates a security enhancing instruction.

username

Names the Authentication Database entry on which to impose security restrictions. Use the $USER variable to read in the username from the **uss add** command's **-user** argument, or from the username field of an **add** instruction in the bulk input file. The ABC Corporation example uses this value.

password_lifetime

Sets the number of days after the user's password is changed that it remains valid. When the password becomes invalid (expires), the user is unable to authenticate, but has 30 more days in which to issue the **kpasswd** command to change the password (after that, only an administrator can change it).

Specify an integer from the range **1** through **254** to specify the number of days until expiration, the value **0** to indicate that the password never expires, or the value $PWEXPIRES to read in the number of days from the **uss add** or **uss bulk** command's **-pwexpires** argument. If the **A** instruction does not appear in the template file, by default the user's password never expires.

The ABC Corporation example sets a password lifetime of 250 days.

password_reuse

Determines whether or not the user can change his or her password (using the **kpasswd** or **kas setpassword** command) to one that is similar to any of his or her last 20 passwords. The acceptable values are **reuse** to allow reuse and **noreuse** to prohibit it. If the **A** instruction does not appear in the template file, the default is to allow password reuse.

The ABC Corporation example prohibits password reuse.

failures

Sets the number of consecutive times the user can provide an incorrect password during authentication (using the **klog** command or a login utility that grants AFS tokens). When the user exceeds the limit, the Authentication Server rejects further authentication attempts for the amount of time specified in the locktime field.

Specify an integer from the range **1** through **254** to specify the number of failures permitted, or the value **0** to indicate that there is no limit to the number of unsuccessful attempts. If the **A** instruction does not appear in the template file, the default is to allow an unlimited number of failures.

The ABC Corporation example sets the limit to nine failed attempts.

locktime

Specifies how long the Authentication Server refuses authentication attempts from a user who has exceeded the failure limit set in the failures field.

Specify a number of hours and minutes (hh:mm) or minutes only (mm), from the range **01** (one minute) through **36:00** (36 hours). The Authentication Server automatically reduces any larger value to **36:00** and also rounds up any nonzero value to the next highest multiple of 8.5 minutes. A value of **0** (zero) sets an infinite lockout time, in which case an administrator must always issue the **kas unlock** command to unlock the account.

The ABC Corporation example sets the lockout time to 25 minutes, which is rounded up to 25 minutes 30 seconds (the next highest multiple of 8.5 minutes).

Executing Commands with the X Instruction

The **X** instruction in the template file executes a command, which can be a standard UNIX command, a shell script or program, or an AFS command. The command string can include standard template variables, and any number of **X** instructions can appear in a template file. If an instruction manipulates an element created by another instruction, it must appear after that instruction.

The following discussion of the field in an **X** instruction refers to the example in the full account template from "Example uss Templates" on page 426:

```
X "create_public_vol $USER $1 $2"
```

The **X** instruction's syntax is as follows:

```
X "command"
```

where command specifies the command to execute. Surround it with double quotes if it contains spaces. The command string can contain any of the standard variables, which the **uss** command interpreter resolves before passing the command on to the appropriate other command interpreter, but it cannot contain newline characters.

The ABC Corporation example invokes a script called **create_public_vol**, which creates another volume associated with the new user and mounts it in a publicly readable part of the ABC Corporation's filespace:

```
"create_public_vol $USER $1 $2"
```

It uses the $USER variable to read in the username and make it part of both the volume name and mount point name. The **uss** command issuer supplies a file server machine name for the $1 variable and a partition name for the $2 variable, to specify the site for the new volume.

Creating Individual Accounts with the uss add Command

After you have created a template file, you can create an individual account by issuing the **uss add** command (for template creation instructions see "Constructing a uss Template File" on page 421). When you issue the command, the **uss** command interpreter contacts various AFS servers to perform the following actions:

- Create a Protection Database entry. By default, the Protection Server assigns an AFS UID which becomes the value of the $UID variable used in the template.
- Create an Authentication Database entry, recording an encrypted version of the initial password.
- Create the account components defined in the indicated template file, contacting the File Server, Volume Server, and Volume Location (VL) Server as necessary.

To review which types of instructions to include in a template to create different file system objects, see "Constructing a uss Template File" on page 421. If the template is empty, the **uss add** command creates an authentication-only account consisting of Protection Database and Authentication Database entries.

When you issue the **uss add** command, provide a value for each variable in the template file by including the corresponding command-line argument. If you fail to supply a value for a variable, the **uss** command interpreter substitutes a null string, which usually causes the account creation to fail. If you include a command line argument for which the corresponding variable does not appear in the template, it is ignored.

"Table 4" on page 442 summarizes the mappings between variables and the arguments to the **uss add** command. It is adapted from "Table 3" on page 423, but includes only those variables that take their value from command line arguments.

Variable	Command-line Argument
$MTPT	**-mount** (for occurrence in **V** instruction)
$NAME	**-realname** if provided; otherwise **-user**
$PART	**-partition**
$PWEXPIRES	**-pwexpires**
$SERVER	**-server**
$UID	**-uid** if provided; otherwise allocated by Protection Server
$USER	**-user**
$1 through $9	**-var**

Table 4. Command-line argument sources for uss template variables

To create an AFS account with the uss add command

1. Authenticate as an AFS identity with all of the following privileges. In the conventional configuration, the **admin** user account has them, or you possibly have a personal administrative account. (To increase cell security, it is best to create special privileged accounts for use only while

performing administrative procedures; for further discussion, see "An Overview of Administrative Privilege" on page 531.) If necessary, issue the **klog** command to authenticate.

```
% klog admin_user
Password: <admin_password>
```

The following list specifies the necessary privileges and indicates how to check that you have them.

- Membership in the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

  ```
  % pts membership system:administrators
  ```

- Inclusion in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

  ```
  % bos listusers <machine name>
  ```

- The `ADMIN` flag on the Authentication Database entry. However, the Authentication Server always prompts you for a password in order to perform its own authentication. The following instructions direct you to specify the administrative identity on the **uss** command line itself.
- The **i** (**insert**) and **l** (**lookup**) permissions on the ACL of the directory in which you are mounting the user's volume. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

  ```
  % fs listacl [<dir/file path>]
  ```

  Members of the **system:administrators** group always implicitly have the **a** (**administer**) and by default also the **l** (**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

2. **(Optional)** Log in as the local superuser **root**. This is necessary only if you are creating new files or directories in the local file system and want to designate an alternate owner as the object is created. For a discussion of the issues involved, see "About Creating Local Disk Directories and Files" on page 426.
3. Verify the location and functionality of the template file you are using. For a description of where the **uss** command interpreter expects to find the template, see "Where to Place Template Files" on page 424. You can always provide an alternate pathname if you wish. Also note the variables used in the template, to be sure that you provide the corresponding arguments on the **uss** command line.
4. **(Optional)** Change to the directory where the template resides. This affects the type of pathname you must type in Step "6" on page 444.

   ```
   % cd template_directory
   ```

5. **(Optional)** Run the **uss add** command with the **-dryrun** flag to preview the creation of the account. Note any error messages and correct the cause before reissuing the command without the **-dryrun** flag. The next step describes the **uss add** command's syntax. For more information on the **-dryrun** flag, see "Avoiding and Recovering from Errors and Interrupted Operations" on page 414.

6. Issue the **uss add** command to create the account. Enter the command on a single line; it appears here on multiple lines only for legibility.

   The **uss add** operation creates an Authentication Database entry. The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

```
% uss add -user <login name>  -admin <administrator to authenticate>   \
          [-realname <full name in quotes>] [-pass <initial passwd>]   \
          [-pwexpires <password expires in [0..254] days (0 => never)>]  \
          [-server <FileServer for home volume>]  \
          [-partition <FileServer's disk partition for home volume>]  \
          [-mount <home directory mount point>]  \
          [-uid <uid to assign the user>]  \
          [-template <pathname of template file>]  \
          [-var <auxiliary argument pairs (Numval)>+] [-dryrun] \
          [-overwrite]
Administrator's (admin_user) password: <admin_password>
```

   where

   ad

   Is the shortest acceptable abbreviation of **add**.

   -user

   Names the user's Authentication Database and Protection Database entries. Because it becomes the username (the name under which a user logs in), it must obey the restrictions that many operating systems impose on usernames (usually, to contain no more than eight lowercase letters). Also avoid the following characters: colon (**:**), semicolon (**;**), comma (**,**), at sign (**@**), space, newline, and the period (**.**), which is conventionally used only in special administrative names.

   This argument provides the value for the $USER variable in the template file. For suggestions on standardizing usernames, see "Choosing Usernames and Naming Other Account Components" on page 37.

   -admin

   Names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

   -realname

   Specifies the user's actual full name. If it contains spaces or punctuation, surround it with double quotes. If you do not provide it, it defaults to the username provided with the **-user** argument.

This argument provides the value for the $NAME variable in the template file. For information about using this argument and variable as part of an automated process for creating entries in a local password file such as **/etc/passwd**, see "Creating a Common Source Password File" on page 417.

-pass

Specifies the user's initial password. Although the AFS commands that handle passwords accept strings of virtually unlimited length, it is best to use a password of eight characters or less, which is the maximum length that many applications and utilities accept.

Possible choices for initial passwords include the username, a string of digits such as those from a Social Security number, or a standard string such as **changeme**, which is the default if you do not provide this argument. There is no corresponding variable in the template file.

Instruct users to change their passwords to a truly secret string as soon as they authenticate with AFS for the first time. The *IBM AFS User Guide* explains how to use the **kpasswd** command to change an AFS password.

-pwexpires

Sets the number of days after a user's password is changed that it remains valid. Provide an integer from the range **1** through **254** to specify the number of days until expiration, or the value **0** to indicate that the password never expires (the default if you do not provide this argument). When the password becomes invalid (expires), the user is unable to authenticate, but has 30 more days in which to issue the **kpasswd** command to change the password; after that, only an administrator can change it.

This argument provides the value for the $PWEXPIRES variable in the template file.

-server

Names the file server machine on which to create the new user's home volume. It is best to provide a fully qualified hostname (for example, **fs1.abc.com**), but an abbreviated form is acceptable provided that the cell's naming service is available to resolve it when you issue the **uss add** command.

This argument provides the value for the $SERVER variable in the template file. To avoid having to type a fully qualified hostname on the command line, combine the $SERVER variable with a constant (for example, the cell's domain name) in the server field of the **V** instruction in the template file. For an example, see "Creating a Volume with the V Instruction" on page 429.

-partition

Specifies the partition on which to create the user's home volume; it must be on the file server machine named by the **-server** argument. Identify the partition by its complete name (for example, **/vicepa**), or use one of the abbreviations listed in "Rules for Using Abbreviations and Aliases" on page 558.

This argument provides the value for the $PART variable in the template file.

-mount

Specifies the pathname for the user's home directory in the cell's read/write filespace. Partial pathnames are interpreted relative to the current working directory.

This argument provides the value for the $MTPT variable in the template file, but only when it appears in the **V** instruction's mount_point field. When the $MTPT variable appears in any subsequent instructions, it takes its value from the **V** instruction's mount_point field, rather than directly from this argument. For more details, and for suggestions about how to use this argument and the $MTPT variable, see "Creating a Volume with the V Instruction" on page 429.

-uid

Specifies a positive integer other than **0** (zero) to assign as the user's AFS UID. It is best to omit this argument and allow the Protection Server to assign an AFS UID that is one greater than the current value of the `max user id` counter. (To display the counter, use the **pts listmax** command as described in "To display the AFS ID counters" on page 510.)

If you have a reason to use this argument (perhaps because the user already has a UNIX UID), first use the **pts examine** command to verify that there is no existing account with the desired AFS UID; if there is, the account creation process terminates with an error.

This argument provides the value for the $UID variable in the template file.

-template

Specifies the pathname of the template file. If you omit this argument, the command interpreter searches for a template file called **uss.template** in each of the following directories in turn:

a. The current working directory

b. **/afs/**cellname**/common/uss**, where cellname names the local cell

c. **/etc**

If you specify a filename other than **uss.template** but without a pathname, the command interpreter searches for it in the indicated directories. If you provide a full or partial pathname, the command interpreter consults the specified file only; it interprets partial pathnames relative to the current working directory.

If the specified template file is empty (zero-length), the command creates Protection and Authentication Database entries only.

To learn how to construct a template file, see "Constructing a uss Template File" on page 421.

-var

Specifies values for each of the number variables $1 through $9 that can appear in the template file. You can use the number variables to assign values to variables in the **uss** template file that are not part of the standard set.

For each instance of this argument, provide two parts in the indicated order, separated by a space:

- The integer from the range **1** through **9** that matches the variable in the template file. Do not precede it with a dollar sign.
- A string of alphanumeric characters to assign as the value of the variable.

To learn about suggested uses for the number variables, see the description of the **V** instruction's quota field in "Creating a Volume with the V Instruction" on page 429.

-dryrun

Reports actions that the command interpreter needs to perform to run the command, without actually performing them.

-overwrite

Overwrites any directories, files, and links that exist in the file system and for which there are definitions in **D**, **E**, **F**, **L**, or **S** instructions in the template file named by the **-template** argument. If you omit this flag, the command interpreter prompts you once for confirmation that you want to overwrite all such elements.

7. If the new user home directory resides in a replicated volume, use the **vos release** command to release the volume, as described in "To replicate a read/write volume (create a read-only volume)" on page 142.

```
% vos release <volume name or ID>
```

Note: This step can be necessary even if the home directory's parent directory is not itself a mount point for a replicated volume (and is easier to overlook in that case). For example, the ABC Corporation template puts the mount points for user volumes in the **/afs/abc.com/usr** directory. Because that is a regular directory rather than a mount point, it resides in the **root.cell** volume mounted at the **/afs/abc.com** directory. That volume is replicated, so after changing it by creating a new mount point the administrator must issue the **vos release** command.

8. Create an entry for the new user in the local password file (**/etc/passwd** or equivalent) on each AFS client machine that he or she can log into. For suggestions on automating this step, see "Creating a Common Source Password File" on page 417.

Even if you do not use the automated method, set the user's UNIX UID to match the AFS UID assigned automatically by the Protection Server or assigned with the **-uid** argument. The new user's AFS UID appears in the trace produced by the **uss add** output, or you can use the **pts examine** command to display it, as described in "To display a Protection Database entry" on page 489.

Deleting Individual Accounts with the uss delete Command

The **uss delete** command deletes an AFS user account according to the arguments you provide on the command line; unlike the **uss add** command, it does not use a template file. When you issue the command, the **uss** command interpreter contacts various AFS servers to perform the following actions:

- Remove the mount point for the user's home volume
- Remove the user's home volume and delete the associated VLDB entry, unless you include the **-savevolume** flag
- Delete the user's Authentication Database entry
- Delete the user's Protection Database entry

Before issuing the **uss delete** command, you can also perform the following optional tasks:

- Copy the user's home volume to tape or another permanent medium and record the username and UID on a reserved list. This information enables you to restore the user's account easily if he or she returns to your cell. For information about using the AFS Backup System to back up volumes, see "Configuring the AFS Backup System" on page 195 and "Backing Up and Restoring AFS Data" on page 241.
- If the user has exclusive use of any other volumes (such as a volume for storing project-related data), make a backup copy of each one and then remove it and its mount point as instructed in "Removing Volumes and their Mount Points" on page 180.
- Use the **pts listowned** command to display any groups that the user owns; instructions appear in "To list the groups that a user or group owns" on page 493. Decide whether to use the **pts delete** command to remove the groups or the **pts chown** command to transfer ownership to another user or group. Instructions appear in "To delete Protection Database entries" on page 503 and "To change a group's owner" on page 504. Alternatively, you can have the user remove or transfer ownership of the groups before leaving. A group that remains in the Protection Database after its owner is removed is considered orphaned, and only members of the **system:administrators** group can administer it.

You can automate some of these tasks by including **exec** instructions in the bulk input file and using the **uss bulk** command to delete the account. See "Creating and Deleting Multiple Accounts with the uss bulk Command" on page 451.

To delete an AFS account

1. Authenticate as an AFS identity with all of the following privileges. In the conventional configuration, the **admin** user account has them, or you possibly have a personal administrative account. (To increase cell security, it is best to create special privileged accounts for use only while performing administrative procedures; for further discussion, see "An Overview of Administrative Privilege" on page 531.) If necessary, issue the **klog** command to authenticate.

   ```
   % klog admin_user
   Password: <admin_password>
   ```

   The following list specifies the necessary privileges and indicates how to check that you have them.

   - Membership in the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

     ```
     % pts membership system:administrators
     ```

   - Inclusion in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

     ```
     % bos listusers <machine name>
     ```

   - The `ADMIN` flag on the Authentication Database entry. However, the Authentication Server always prompts you for a password in order to perform its own authentication. The following instructions direct you to specify the administrative identity on the **uss** command line itself.
   - The **d** (**delete**) permission on the ACL of the directory that houses the user's home directory. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

     ```
     % fs listacl [<dir/file path>]
     ```

     Members of the **system:administrators** group always implicitly have the **a** (**administer**) and by default also the **l** (**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

2. Consider and resolve the issues discussed in the introduction to this section concerning the continued maintenance of a deleted user's account information, owned groups, and volumes.
3. **(Optional)** Run the **uss delete** command with the **-dryrun** flag to preview the deletion of the account. Note any error messages and correct the cause before reissuing the command without the **-dryrun** flag. The next step describes the **uss delete** command's syntax.
4. Issue the **uss delete** command to delete the account. Enter the command on a single line; it appears here on multiple lines only for legibility.

   The delete operation always removes the user's entry from the Authentication Database. The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the ADMIN flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

```
% uss delete -user <login name>  \
             -mountpoint <mountpoint for user's volume>  \
             [-savevolume]  -admin  <administrator to authenticate>  \
             [-dryrun]
Administrator's (admin_user) password: <admin_password>
```

where

d

Is the shortest acceptable abbreviation of **delete**.

-user

Names the entry to delete from the Protection and Authentication Databases.

-mountpoint

Specifies the pathname of the mount point to delete (the user's home directory). Unless the **-savevolume** argument is included, the volume mounted there is also deleted from the file server machine where it resides, as is its record from the VLDB. Partial pathnames are interpreted relative to the current working directory.

Specify the read/write path to the mount point, to avoid the failure that results when you attempt to delete a mount point from a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

-savevolume

Retains the user's volume and VLDB entry.

-admin

Names an administrative account that has the ADMIN flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

-dryrun

Reports actions that the command interpreter needs to perform to run the command, without actually performing them.

5. If the deleted user home directory resided in a replicated volume, use the **vos release** command to release the volume, as described in "To replicate a read/write volume (create a read-only volume)" on page 142.

```
% vos release <volume name or ID>
```

Note: This step can be necessary even if the home directory's parent directory is not itself a mount point for a replicated volume (and is easier to overlook in that case). For example, the ABC Corporation template puts the mount points for user volumes in the **/afs/abc.com/usr** directory. Because that is a regular directory rather than a mount point, it resides in the **root.cell** volume mounted at the **/afs/abc.com** directory. That volume is replicated, so after changing it by deleting a mount point the administrator must issue the **vos release** command.

6. Delete the user's entry from the local password file (**/etc/passwd** or equivalent) of each client machine. If you use the AFS **package** utility, it is sufficient to remove the entry from the common source version of the file. If you intend to reactivate the user's account in the future, it is simpler to comment out the entry or place an asterisk (*) in the password field.

Creating and Deleting Multiple Accounts with the uss bulk Command

The **uss bulk** command allows you to create and delete many accounts at once. Before executing the command, you must

- Construct a template if you plan to create any accounts, just as you must do before running the **uss add** command. The same template applies to all accounts created by a single **uss bulk** command.
- Construct a bulk input file of instructions that create and delete accounts and execute any related commands, as described in "Constructing a Bulk Input File" on page 451.

Constructing a Bulk Input File

You can include five types of instructions in a bulk input file: **add**, **delete**, **exec**, **savevolume**, and **delvolume**. The following sections discuss their uses.

Creating a User Account with the add Instruction

Each **add** instruction creates a single user account, and so is basically the equivalent of issuing one **uss add** command. There is no limit to the number of **add** instructions in the bulk input file.

As indicated by the following syntax statement, the order of the instruction's fields matches the order of arguments to the **uss add** command (though some of the command's arguments do not have a corresponding field). Like the **uss add** command's arguments, many of the fields provide a value for a variable in the **uss** template file. Each instruction must be a single line in the file (have a newline character only at its end); it appears on multiple lines here only for legibility.

```
add username[:full_name][:initial_password][:password_expires]
    [:file_server][:partition][:mount_point][:uid]
    [:var1][:var2][:var3][:var4][:var5][:var6][:var7][:var8][:var9][:]
```

For a complete description of the acceptable values in each field, see the **uss Bulk Input File** reference page in the *IBM AFS Administration Reference*, or the description of the corresponding arguments to the **uss add** command, in "To create an AFS account with the uss add command" on page 442. Following are some basic notes:

- Begin the line with the string **add** only, not **uss add**.
- Only the first argument, username, is required. It corresponds to the **-user** argument to the **uss add** command.
- Do not surround the full_name value with double quotes, even though you must use them around the value for the **-realname** argument to the **uss add** command.
- If you want to omit a value for an argument, indicate an empty field by using two colons with nothing between them. Leaving a field empty is acceptable if the corresponding command line argument is optional or if the corresponding variable does not appear in the template file. For every field that precedes the last one to which you assign an actual value, you must either provide a value or indicate an empty field. It is acceptable, but not necessary, to indicate empty fields after the last one in which you assign a value.
- After the last field, end the line with either a colon and newline character (**<Return>**), or a newline alone.
- The final nine fields are for assigning values to the number variables ($1 through $9), with the fields listed in increasing numerical order. Specify the value only, not the variable number.

Deleting a User Account with the delete Instruction

Each **delete** instruction deletes a single user account, and so is basically the equivalent of issuing one **uss delete** command. There is no limit to the number of **delete** instructions in the bulk input file.

Like all instructions in the bulk input file, each **delete** instruction must be a single line in the file (have a newline character only at its end), even though it can cover multiple lines on a display screen. The curly braces (**{ }**) indicate two mutually exclusive choices.

```
delete username:mount_point_path[:{ savevolume | delvolume }][:]
```

For a complete description of the acceptable values in each field, see the **uss Bulk Input File** reference page in the *IBM AFS Administration Reference* or the description of the corresponding arguments to the **uss delete** command, in "To delete an AFS account" on page 448. Following are some basic notes:

- Begin the line with the string **delete** only, not **uss delete**.
- The first two arguments, username and mount_point_path, are required. They correspond to the **-user** and **-mountpoint** arguments to the **uss delete** command.
- The third field, which is optional, controls whether the user's home volume is removed from the file server where it resides, along with the corresponding VLDB entry. There are three possible values:

  - No value treats the volume and VLDB entry according to the prevailing default, which is established by a preceding **savevolume** or **delvolume** instruction in the template file. See the following discussion of those instructions to learn how the default is set.
  - The string **savevolume** preserves the volume and VLDB entry, overriding the default.
  - The string **delvolume** removes the volume and VLDB entry, overriding the default.
- After the last field, end the line with either a colon and newline character (**<Return>**), or a newline alone.

Running a Command or Script with the exec Instruction

The **exec** instruction runs the indicated AFS command, compiled program, or UNIX shell script or command. The command processor assumes the AFS and local identities of the issuer of the **uss bulk** command, who must have the privileges required to run the command.

The instruction's syntax is as follows:

```
exec command
```

It is not necessary to surround the command string with double quotes (" ") or other delimiters.

Setting the Default Treatment of Volumes with the delvolume and savevolume Instructions

The **savevolume** and **delvolume** instructions set the default treatment of volumes referenced by the **delete** instructions that follow them in the bulk input file. Their syntax is as follows:

```
savevolume
delvolume
```

Both instructions are optional and take no arguments. If neither appears in the bulk input file, then by default all volumes and VLDB entries referenced by **delete** instructions are removed. If the **savevolume** instruction appears in the file, it prevents the removal of the volume and VLDB entry referenced by all subsequent **delete** instructions in the file. The **delvolume** instruction explicitly establishes the default (which is deletion) for subsequent **delete** instructions.

The effect of either instruction lasts until the end of the bulk input file, or until its opposite appears. To override the prevailing default for a particular **delete** instruction, put the **savevolume** or **delvolume** string in the instruction's third field. (You can also use multiple instances of the **savevolume** and **delvolume** instructions to toggle back and forth between default preservation and deletion of volumes.)

Example Bulk Input File Instructions

To create an authentication-only account, use an **add** instruction like the following example, which includes only the first (username) argument. The user's real name is set to match the username (**anderson**) and her initial password is set to the string **changeme**.

```
add anderson
```

The following example also creates an authentication-only account, but sets nondefault values for the real name and initial password.

```
add smith:John Smith:js_pswd
```

The next two example **add** instructions require that the administrator of the ABC Corporation cell (**abc.com**) has written a **uss** template file with the following **V** instruction in it:

```
V user.$USER $SERVER.abc.com /vicep$PART 10000 /afs/.abc.com/usr/$3/$USER \
    $UID $USER all
```

To create accounts for users named John Smith from the Marketing Department and Pat Jones from the Finance Department, the appropriate **add** instructions in the bulk input file are as follows:

```
add smith:John Smith:::fs1:a:::::marketing
add jones:Pat Jones:::fs3:c:::::finance
```

The new account for Smith consists of Protection and Authentication Database entries called **smith**. His initial password is the default string **changeme**, and the Protection Server generates his AFS UID. His home volume, called **user.smith**, has a 10,000 KB quota, resides on partition **/vicepa** of file server machine **fs1.abc.com**, and is mounted at **/afs/.abc.com/usr/marketing/smith**. The final **$UID $USER all** part of the **V** instruction gives him ownership of his home directory and all permissions on its ACL. The account for **jones** is similar, except that it resides on partition **/vicepc** of file server machine **fs3.abc.com** and is mounted at **/afs/.abc.com/usr/finance/jones**.

Notice that the fields corresponding to mount_point, uid, var1, and var2 are empty (between the values `a` and `marketing` on the first example line) because the corresponding variables do not appear in the **V** instruction in the template file. The initial_passwd and password_expires fields are also empty.

If you wish, you can specify values or empty fields for all nine number variables in an **add** instruction. In that case, the bulk input file instructions are as follows:

```
add smith:John Smith:::fs1:a:::::marketing::::::
add jones:Pat Jones:::fs3:c:::::finance::::::
```

The following example is a section of a bulk input file with a number of **delete** instructions and a **savevolume** instruction. Because the first three instructions appear before the **savevolume** instruction and their third field is blank, the corresponding volumes and VLDB entries are removed. The **delete** instruction for user **terry** follows the **savevolume** instruction, so her volume is not removed, but the volume for user **johnson** is, because the **delvolume** string in the third field of the **delete** instruction overrides the current default.

```
delete smith:/afs/abc.com/usr/smith
delete pat:/afs/abc.com/usr/pat
delete rogers:/afs/abc.com/usr/rogers
savevolume
delete terry:/afs/abc.com/usr/terry
delete johnson:/afs/abc.com/usr/johnson:delvolume
```

The following example **exec** instruction is useful as a separator between a set of **add** instructions and a set of **delete** instructions. It generates a message on the standard output stream that informs you of the **uss bulk** command's progress.

```
exec echo "Additions completed; beginning deletions..."
```

To create and delete multiple AFS user accounts

1. Authenticate as an AFS identity with all of the following privileges. In the conventional configuration, the **admin** user account has them, or you possibly have a personal administrative account. (To increase cell security, it is best to create special privileged accounts for use only while performing administrative procedures; for further discussion, see "An Overview of Administrative Privilege" on page 531.) If necessary, issue the **klog** command to authenticate.

   ```
   % klog admin_user
   Password: <admin_password>
   ```

   The following list specifies the necessary privileges and indicates how to check that you have them.

   - Membership in the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

     ```
     % pts membership system:administrators
     ```

   - Inclusion in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

     ```
     % bos listusers <machine name>
     ```

   - The `ADMIN` flag on the Authentication Database entry. However, the Authentication Server always prompts you for a password in order to perform its own authentication. The following instructions direct you to specify the administrative identity on the **uss** command line itself.
   - The **d** (**delete**), **i** (**insert**) and **l** (**lookup**) permissions on the ACL of the parent directory for each volume mount point. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

     ```
     % fs listacl [<dir/file path>]
     ```

     Members of the **system:administrators** group always implicitly have the **a** (**administer**) and by default also the **l** (**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

2. **(Optional.)** Log in as the local superuser **root**. This is necessary only if you are creating new files or directories in the local file system and want to designate an alternate owner as the object is created. For a discussion of the issues involved, see "About Creating Local Disk Directories and Files" on page 426.
3. If the bulk input file includes **add** instructions, verify the location and functionality of the template you are using. For a description of where the **uss** command interpreter expects to find the template, see "Where to Place Template Files" on page 424. You can always provide an alternate pathname if you wish. Also note which variables appear in the template, to be sure that you provide the corresponding arguments in the **add** instruction or on the **uss bulk** command line.
4. Create a bulk input file that complies with the rules listed in "Constructing a Bulk Input File" on page 451. It is simplest to put the file in the same directory as the template file you are using.

5. **(Optional.)** Change to the directory where the bulk input file and template file reside.

   ```
   % cd template_directory
   ```

6. Issue the **uss bulk** command to create or delete accounts, or both. Enter the command on a single line; it appears here on multiple lines only for legibility.

   The bulk operation always manipulates user entries in the Authentication Database. The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % uss bulk <bulk input file>  \
              [-template <pathname of template file>]  \
              -admin <administrator to authenticate>  \
              [-dryrun] [-overwrite]  \
              [-pwexpires <password expires in [0..254] days (0 => never)>]  \
              [-pipe]
   Administrator's (admin_user) password: <admin_password>
   ```

   where

   b

   Is the shortest acceptable abbreviation of **bulk**.

   bulk input file

   Specifies the pathname of the bulk input file. Partial pathnames are interpreted relative to the current working directory. For a discussion of the required file format, see "Constructing a Bulk Input File" on page 451.

   -template

   Specifies the pathname of the template file for any **uss add** commands that appear in the bulk input file. Partial pathnames are interpreted relative to the current working directory. For a discussion of the required file format, see "Constructing a uss Template File" on page 421.

   -admin

   Names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as the **admin** account. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

   -dryrun

   Reports actions that the command interpreter needs to perform to run the command, without actually performing them.

   -overwrite

   Overwrites any directories, files and links that exist in the file system and for which there are also **D**, **E**, **F**, **L**, or **S** instructions in the template file named by the **-template** argument. If this flag is omitted, the command interpreter prompts, once for each **add** instruction in the bulk input file, for confirmation that it is to overwrite such elements. Do not include this flag if there are no **add** instructions in the bulk input file.

-pwexpires

Sets the number of days after a user's password is changed that it remains valid, for each user named by an **add** instruction in the bulk input file. Provide an integer from the range **1** through **254** to specify the number of days until expiration, or the value **0** to indicate that the password never expires (the default).

When the password becomes invalid (expires), the user is unable to authenticate, but has 30 more days in which to issue the **kpasswd** command to change the password (after that, only an administrator can change it).

-pipe

Suppresses the Authentication Server's prompt for the password of the issuer or the user named by the **-admin** argument (the Authentication Server always separately authenticates the user who is creating or deleting an entry in the Authentication Database). Instead, the command interpreter accepts the password as piped input from another program, enabling you to run the **uss bulk** command in unattended batch jobs.

7. If a newly created or deleted user home directory resides in a replicated volume, use the **vos release** command to release the volume, as described in "To replicate a read/write volume (create a read-only volume)" on page 142.

```
% vos release <volume name or ID>
```

Note: This step can be necessary even if the home directory's parent directory is not itself a mount point for a replicated volume (and is easier to overlook in that case). For example, the ABC Corporation template puts the mount points for user volumes in the **/afs/abc.com/usr** directory. Because that is a regular directory rather than a mount point, it resides in the **root.cell** volume mounted at the **/afs/abc.com** directory. That volume is replicated, so after changing it by creating or deleting a mount point, the administrator must issue the **vos release** command.

8. If you are creating accounts, create an entry for the new user in the local password file (**/etc/passwd** or equivalent) on each AFS client machine that he or she can log into. For suggestions on automating this step, see "Creating a Common Source Password File" on page 417.

   Even if you do not use the automated method, set the user's UNIX UID to match the AFS UID assigned automatically by the Protection Server or assigned with the **-uid** argument. The new user's AFS UID appears in the trace produced by the **uss add** output or you can use the **pts examine** command to display it, as described in "To display a Protection Database entry" on page 489.

9. If you are deleting accounts, delete the user's entry from the local password file (**/etc/passwd** or equivalent) of each client machine. If you use the AFS **package** utility, it is sufficient to remove the entry from the common source version of the file. If you intend to reactivate the user's account in the future, it is simpler to comment out the entry or place an asterisk (*) in the password field.

Chapter 13. Administering User Accounts

This chapter explains how to create and maintain user accounts in your cell.

The preferred method for creating user accounts is the **uss** program, which enables you to create multiple accounts with a single command. See "Creating and Deleting User Accounts with the uss Command Suite" on page 413. If you prefer to create each account component individually, follow the instructions in "Creating AFS User Accounts" on page 463.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Create Protection Database entry	**pts createuser**
Create Authentication Database entry	**kas create**
Create volume	**vos create**
Mount volume	**fs mkmount**
Create entry on ACL	**fs setacl**
Examine Protection Database entry	**pts examine**
Change directory ownership	**/etc/chown**
Limit failed authentication attempts	**kas setfields** with **-attempts** and **-locktime**
Unlock Authentication Database entry	**kas unlock**
Set password lifetime	**kas setfields** with **-pwexpires**
Prohibit password reuse	**kas setfields** with **-reuse**
Change AFS password	**kas setpassword**
List groups owned by user	**pts listowned**
Rename Protection Database entry	**pts rename**
Delete Authentication Database entry	**kas delete**
Rename volume	**vos rename**
Remove mount point	**fs rmmount**
Delete Protection Database entry	**pts delete**
List volume location	**vos listvldb**
Remove volume	**vos remove**

The Components of an AFS User Account

The differences between AFS and the UNIX file system imply that a complete AFS user account is not the same as a UNIX user account. The following list describes the components of an AFS account. The same information appears in a corresponding section of "Creating and Deleting User Accounts with the uss Command Suite" on page 413, but is repeated here for your convenience.

- A *Protection Database entry* defines the username (the name provided when authenticating with AFS), and maps it to an AFS user ID (AFS UID), a number that the AFS servers use internally when referencing users. The Protection Database also tracks the groups to which the user belongs. For details, see "Administering the Protection Database" on page 487.
- An *Authentication Database entry* records the user's AFS password in a scrambled form suitable for use as an encryption key.
- A home *volume* stores all the files in the user's home directory together on a single partition of a file server machine. The volume has an associated quota that limits its size. For a complete discussion of volumes, see "Managing Volumes" on page 131.
- A *mount point* makes the contents of the user's volume visible and accessible in the AFS filespace, and acts as the user's home directory. For more details about mount points, see "About Mounting Volumes" on page 134.
- Full access permissions on the home directory's *access control list (ACL)* and ownership of the directory (as displayed by the UNIX **ls -ld** command) enable the user to manage his or her files. For details on AFS file protection, see "Managing Access Control Lists" on page 513.
- A *local password file entry* (in the **/etc/passwd** file or equivalent) of each AFS client machine enables the user to log in and access AFS files through the Cache Manager. A subsequent section in this chapter further discusses local password file entries.
- Other optional *configuration files* make the account more convenient to use. Such files help the user log in and log out more easily, receive electronic mail, print, and so on.

Creating Local Password File Entries

To obtain authenticated access to a cell's AFS filespace, a user must not only have a valid AFS token, but also an entry in the local password file (**/etc/passwd** or equivalent) of the machine whose Cache Manager is representing the user. This section discusses why it is important for the user's AFS UID to match to the UNIX UID listed in the local password file, and describes the appropriate value to put in the file's password field.

One reason to use **uss** commands is that they enable you to generate local password file entries automatically as part of account creation. See "Creating a Common Source Password File" on page 417.

Information similar to the information in this section appears in a corresponding section of "Creating and Deleting User Accounts with the uss Command Suite" on page 413, but is repeated here for your convenience

Assigning AFS and UNIX UIDs that Match

A user account is easiest to administer and use if the AFS user ID number (AFS UID) and UNIX UID match. All instructions in the AFS documentation assume that they do.

The most basic reason to make AFS and UNIX UIDs the same is so that the owner name reported by the UNIX **ls -l** and **ls -ld** commands makes sense for AFS files and directories. Following standard UNIX practice, the File Server records a number rather than a username in an AFS file or directory's owner field: the owner's AFS UID. When you issue the **ls -l** command, it translates the UID to a username according to the mapping in the local password file, not the AFS Protection Database. If the AFS and UNIX UIDs do not match, the **ls -l** command reports an unexpected (and incorrect) owner. The output can even vary on different client machines if their local password files map the same UNIX UID to different names.

Follow the recommendations in the indicated sections to make AFS and UNIX UIDs match when creating accounts for various types of users:

- If creating an AFS account for a user who already has a UNIX UID, see "Making UNIX and AFS UIDs Match" on page 462.
- If some users in your cell have existing UNIX accounts but the user for whom you are creating an AFS account does not, then it is best to allow the Protection Server to allocate an AFS UID automatically. To avoid overlap of AFS UIDs with existing UNIX UIDs, set the Protection Database's `max user id` counter higher than the largest UNIX UID, using the instructions in "Displaying and Setting the AFS UID and GID Counters" on page 509.
- If none of your users have existing UNIX accounts, allow the Protection Server to allocate AFS UIDs automatically, starting either at its default or at the value you have set for the `max user id` counter.

Specifying Passwords in the Local Password File

Authenticating with AFS is easiest for your users if you install and configure an AFS-modified login utility, which logs a user into the local file system and obtains an AFS token in one step. In this case, the local password file no longer controls a user's ability to login in most circumstances, because the AFS-modified login utility does not consult the local password file if the user provides the correct AFS password. You can nonetheless use a password file entry's password field (usually, the second field) in the following ways to control login and authentication:

- To prevent both local login and AFS authentication, place an asterisk (*) in the field. This is useful mainly in emergencies, when you want to prevent a certain user from logging into the machine.
- To prevent login to the local file system if the user does not provide the correct AFS password, place a character string of any length other than the standard thirteen characters in the field. This is appropriate if you want to allow only people with local AFS accounts to log into to your machines. A single **X** or other character is the most easily recognizable way to do this.
- To enable a user to log into the local file system even after providing an incorrect AFS password, record a standard UNIX encrypted password in the field by issuing the standard UNIX password-setting command (**passwd** or equivalent).

If you do not use an AFS-modified login utility, you must place a standard UNIX password in the local password file of every client machine the user will use. The user logs into the local file system only, and then must issue the **klog** command to authenticate with AFS. It is simplest if the passwords in the local password file and the Authentication Database are the same, but this is not required.

Converting Existing UNIX Accounts

This section discusses the three main issues you need to consider if your cell has existing UNIX accounts that you wish to convert to AFS accounts.

Making UNIX and AFS UIDs Match

As previously mentioned, AFS users must have an entry in the local password file on every client machine from which they access the AFS filespace as an authenticated user. Both administration and use are much simpler if the UNIX UID and AFS UID match. When converting existing UNIX accounts, you have two alternatives:

- Make the AFS UIDs match the existing UNIX UIDs. In this case, you need to assign the AFS UID yourself by including the **-id** argument to the **pts createuser** command as you create the AFS account.

  Because you are retaining the user's UNIX UID, you do not need to alter the UID in the local password file entry. However, if you are using an AFS-modified login utility, you possibly need to change the password field in the entry. For a discussion of how the value in the password field affects login with an AFS-modified login utility, see "Specifying Passwords in the Local Password File" on page 461.

  If now or in the future you need to create AFS accounts for users who do not have an existing UNIX UID, then you must guarantee that new AFS UIDs do not conflict with any existing UNIX UIDs. The simplest way is to set the `max user id` counter in the Protection Database to a value higher than the largest existing UNIX UID. See "Displaying and Setting the AFS UID and GID Counters" on page 509.

- Change the existing UNIX UIDs to match the new AFS UIDs that the Protection Server assigns automatically.

  Allow the Protection Server to allocate the AFS UIDs automatically as you create AFS accounts. You must then alter the user's entry in the local password file on every client machine to include the new UID.

  There is one drawback to changing the UNIX UID: any files and directories that the user owned in the local file system before becoming an AFS user still have the former UID in their owner field. If you want the **ls -l** and **ls -ld** commands to display the correct owner, you must use the **chown** command to change the value to the user's new UID, whether you are leaving the file in the local file system or moving it to AFS. See "Moving Local Files into AFS" on page 463.

Setting the Password Field Appropriately

Existing UNIX accounts already have an entry in the local password file, probably with a (scrambled) password in the password field. You possibly need to change the value in the field, depending on the type of login utility you use:

- If the login utility is not modified for use with AFS, the actual password must appear (in scrambled form) in the local password file entry.
- If the login utility is modified for use with AFS, choose one of the values discussed in "Specifying Passwords in the Local Password File" on page 461.

Moving Local Files into AFS

New AFS users with existing UNIX accounts probably already own files and directories stored in a machine's local file system, and it usually makes sense to transfer them into the new home volume. The easiest method is to move them onto the local disk of an AFS client machine, and then use the UNIX **mv** command to transfer them into the user's new AFS home directory.

As you move files and directories into AFS, keep in mind that the meaning of their mode bits changes. AFS ignores the second and third sets of mode bits (group and other), and does not use the first set (the owner bits) directly, but only in conjunction with entries on the ACL (for details, see "How AFS Interprets the UNIX Mode Bits" on page 529). Be sure that the ACL protects the file or directory at least as securely as the mode bits.

If you have chosen to change a user's UNIX UID to match a new AFS UID, you must change the ownership of UNIX files and directories as well. Only members of the **system:administrators** group can issue the **chown** command on files and directories once they reside in AFS.

Creating AFS User Accounts

There are two methods for creating user accounts. The preferred method--using the **uss** commands--enables you to create multiple accounts with a single command. It uses a template to define standard values for the account components that are the same for each user (such as quota), but provide differing values for more variable components (such as username). See "Creating and Deleting User Accounts with the uss Command Suite" on page 413.

The second method involves issuing a separate command to create each component of the account. It is best suited to creation of one account at a time, since some of the commands can create only one instance of the relevant component. To review the function of each component, see "The Components of an AFS User Account" on page 459.

Use the following instructions to create any of the three types of user account, which differ in their levels of functionality. For a description of the types, see "Configuring AFS User Accounts" on page 35.

- To create an authentication-only account, perform Step "1" on page 464 through Step "4" on page 466 and also Step "14" on page 470. This type of account consists only of entries in the Authentication Database and Protection Database.
- To create a basic account, perform Step "1" on page 464 through Step "8" on page 468 and Step "11" on page 469 through Step "14" on page 470. In addition to Authentication Database and Protection Database entries, this type of account includes a volume mounted at the home directory with owner and ACL set appropriately.
- To create a full account, perform all steps in the following instructions. This type of account includes configuration files for basic functions such as logging in, printing, and mail delivery, making it more convenient and useful. For a discussion of some useful types of configuration files, see "Creating Standard Files in New AFS Accounts" on page 39.

To create one user account with individual commands

1. Decide on the value to assign to each of the following account components. If you are creating an authentication-only account, you need to pick only a username, AFS UID, and initial password.

   - The username. By convention, the names of many components of the user account incorporate this name. For a discussion of restrictions and suggested naming schemes, see "Choosing Usernames and Naming Other Account Components" on page 37.
   - The AFS UID, if you want to assign a specific one. It is generally best to have the Protection Server allocate one instead, except when you are creating an AFS account for a user who already has an existing UNIX account. In that case, migrating the user's files into AFS is simplest if you set the AFS UID to match the existing UNIX UID. See "Converting Existing UNIX Accounts" on page 462.
   - The initial password. Advise the user to change this at the first login, using the password changing instructions in the *IBM AFS User Guide*.
   - The name of the user's home volume. The conventional name is **user.**username (for example, **user.smith**).
   - The volume's site (disk partition on a file server machine). Some cells designate certain machines or partitions for user volumes only, or it possibly makes sense to place the volume on the emptiest partition that meets your other criteria. To display the size and available space on a partition, use the **vos partinfo** command, which is fully described in "Creating Read/write Volumes" on page 135.
   - The name of the user's home directory (the mount point for the home volume). The conventional location is a directory (or one of a set of directories) directly under the cell directory, such as **/afs/**cellname**/usr**. For suggestions on how to avoid the slowed directory lookup that can result from having large numbers of user home directories in a single **usr** directory, see "Evenly Distributing User Home Directories with the G Instruction" on page 428.
   - The volume's space quota. Include the **-maxquota** argument to the **vos create** command, or accept the default quota of 5000 KB.

- The ACL on the home directory. By default, the ACL on every new volume grants all seven permissions to the **system:administrators** group. After volume creation, use the **fs setacl** command to remove the entry if desired, and to grant all seven permissions to the user.

2. Authenticate as an AFS identity with all of the following privileges. In the conventional configuration, the **admin** user account has them, or you possibly have a personal administrative account. (To increase cell security, it is best to create special privileged accounts for use only while performing administrative procedures; for further discussion, see "An Overview of Administrative Privilege" on page 531.) If necessary, issue the **klog** command to authenticate.

   ```
   % klog admin_user
   Password: <admin_password>
   ```

   The following list specifies the necessary privileges and indicates how to check that you have them.

   - Membership in the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

     ```
     % pts membership system:administrators
     ```

   - Inclusion in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

     ```
     % bos listusers <machine name>
     ```

   - The ADMIN flag on your Authentication Database entry. However, the Authentication Server performs its own authentication, so in Step "4" on page 466 you specify an administrative identity on the **kas** command line itself.

   - The **i** (**insert**) and **a** (**administer**) permissions on the ACL of the directory where you are mounting the user's volume. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

     ```
     % fs listacl [<dir/file path>]
     ```

     Members of the **system:administrators** group always implicitly have the **a** (**administer**) and by default also the **l** (**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

   - Knowledge of the password for the local superuser **root**.

3. Issue the **pts createuser** command to create an entry in the Protection Database. For a discussion of setting AFS UIDs, see "Assigning AFS and UNIX UIDs that Match" on page 460. If you are converting an existing UNIX account into an AFS account, also see "Converting Existing UNIX Accounts" on page 462.

   ```
   % pts createuser <user name> [<user id>]
   ```

   where

cu

Is an acceptable alias for **createuser** (and **createu** is the shortest acceptable abbreviation).

user name

Specifies the user's username (the character string typed at login). It is best to limit the name to eight or fewer lowercase letters, because many application programs impose that limit. The AFS servers themselves accept names of up to 63 lowercase letters. Also avoid the following characters: colon (**:**), semicolon (**;**), comma (**,**), at sign (**@**), space, newline, and the period (**.**), which is conventionally used only in special administrative names.

user id

Is optional and appropriate only if the user already has a UNIX UID that the AFS UID must match. If you do not provide this argument, the Protection Server assigns one automatically based on the counter described in "Displaying and Setting the AFS UID and GID Counters" on page 509. If the ID you specify is less than **1** (one) or is already in use, an error results.

4. Issue the **kas create** command to create an entry in the Authentication Database. To avoid having the user's temporary initial password echo visibly on the screen, omit the **-initial_password** argument; instead enter the password at the prompts that appear when you omit the argument, as shown in the following syntax specification.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas create <name of user> \
                -admin  <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   initial_password: <initial_password>
   Verifying, please re-enter initial_password: <initial_password>
   ```

   where

cr

Is the shortest acceptable abbreviation for **create**.

name of user

Specifies the same username as in Step "3" on page 465.

-admin

Names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

initial_password

Specifies the initial password as a string of eight characters or less, to comply with the length restriction that some applications impose. Possible choices for an initial password include the username, a string of digits from a personal identification number such as the Social Security number, or a standard string such as **changeme**. Instruct the user to change the string to a truly secret password as soon as possible by using the **kpasswd** command as described in the *IBM AFS User Guide*.

5. Issue the **vos create** command to create the user's volume.

```
% vos create <machine name> <partition name> <volume name>  \
             [-maxquota <initial quota (KB)>]
```

where

cr

Is the shortest acceptable abbreviation of **create**.

machine name

Names the file server machine on which to place the new volume.

partition name

Names the partition on which to place the new volume.

volume name

Names the new volume. The name can include up to 22 characters. By convention, user volume names have the form **user.**username, where username is the name assigned in Step "3" on page 465.

-maxquota

Sets the volume's quota, as a number of kilobyte blocks. If you omit this argument, the default is 5000 KB.

6. Issue the **fs mkmount** command to mount the volume in the filespace and create the user's home directory.

```
% fs mkmount <directory> <volume name>
```

where

mk

Is the shortest acceptable abbreviation for **mkmount**.

directory

Names the mount point to create. A directory of the same name must not already exist. Partial pathnames are interpreted relative to the current working directory. By convention, user home directories are mounted in a directory called something like **/afs/.**cellname**/usr**, and the home directory name matches the username assigned in Step "3" on page 465.

Specify the read/write path to the mount point, to avoid the failure that results when you attempt to create the new mount point in a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

volume name

Is the name of the volume created in Step "5" on page 467.

7. **(Optional)** Issue the **fs setvol** command with the **-offlinemsg** argument to record auxiliary information about the volume in its volume header. For example, you can record who owns the volume or where you have mounted it in the filespace. To display the information, use the **fs examine** command.

```
% fs setvol <dir/file path> -offlinemsg <offline message>
```

where

sv

Is an acceptable alias for **setvol** (and **setv** the shortest acceptable abbreviation).

dir/file path

Names the mount point of the volume with which to associate the message. Partial pathnames are interpreted relative to the current working directory.

Specify the read/write path to the mount point, to avoid the failure that results when you attempt to change a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

-offlinemsg

Specifies up to 128 characters of auxiliary information to record in the volume header.

8. Issue the **fs setacl** command to set the ACL on the new home directory. At the least, create an entry that grants all permissions to the user, as shown.

You can also use the command to edit or remove the entry that the **vos create** command automatically places on the ACL for a new volume's root directory, which grants all permissions to the **system:administrators** group. Keep in mind that even if you remove the entry, the members of the group by default have implicit **a** (**administer**) and by default **l** (**lookup**) permissions on every ACL, and can grant themselves other permissions as required.

For detailed instructions for the **fs setacl** command, see "Setting ACL Entries" on page 521.

```
% fs setacl <directory> -acl <user name> all \
            [system:administrators desired_permissions]
```

9. **(Optional)** Create configuration files and subdirectories in the new home directory. Possibilities include **.login** and **.logout** files, a shell-initialization file such as **.cshrc**, files to help with printing and mail delivery, and so on.

   If you are converting an existing UNIX account into an AFS account, you possibly wish to move some files and directories into the user's new AFS home directory. See "Converting Existing UNIX Accounts" on page 462.

10. **(Optional)** In the new **.login** or shell initialization file, define the user's $PATH environment variable to include the directories where AFS binaries are kept (for example, the **/usr/afsws/bin** and **/usr/afsws/etc** directories).

11. In Step "12" on page 469 and Step "14" on page 470, you must know the user's AFS UID. If you had the Protection Server assign it in Step "3" on page 465, you probably do not know it. If necessary, issue the **pts examine** command to display it.

    ```
    % pts examine <user or group name or id>
    ```

    where

    e

    Is the shortest acceptable abbreviation of **examine**.

    user or group name or id

    Is the username that you assigned in Step "3" on page 465.

    The first line of the output displays the username and AFS UID. For further discussion and an example of the output, see "Displaying Information from the Protection Database" on page 489.

12. Designate the user as the owner of the home directory and any files and subdirectories created or moved in Step "9" on page 469. Specify the owner by the AFS UID you learned in Step "11" on page 469 rather than by username. This is necessary for new accounts because the user does not yet have an entry in your local machine's password file (**/etc/passwd** or equivalent). If you are converting an existing UNIX account, an entry possibly already exists, but the UID is possibly incorrect. In that case, specifying a username means that the corresponding (possibly incorrect) UID is recorded as the owner.

    Some operating systems allow only the local superuser **root** to issue the **chown** command. If necessary, issuing the **su** command before the **chown** command.

    ```
    % chown new_owner_ID  directory
    ```

where

new_owner_ID

Is the user's AFS UID, which you learned in Step "11" on page 469.

directory

Names the home directory you created in Step "6" on page 467, plus each subdirectory or file you created in Step "9" on page 469.

13. If the new user home directory resides in a replicated volume, use the **vos release** command to release the volume, as described in "To replicate a read/write volume (create a read-only volume)" on page 142.

    ```
    % vos release <volume name or ID>
    ```

    Note: This step can be necessary even if the home directory's parent directory is not itself a mount point for a replicated volume (and is easier to overlook in that case). Suppose, for example, that the ABC Corporation puts the mount points for user volumes in the **/afs/abc.com/usr** directory. Because that is a regular directory rather than a mount point, it resides in the **root.cell** volume mounted at the **/afs/abc.com** directory. That volume is replicated, so after changing it by creating a new mount point the administrator must issue the **vos release** command.

14. Create or modify an entry for the new user in the local password file (**/etc/passwd** or equivalent) of each machine the user can log onto. Remember to make the UNIX UID the same as the AFS UID you learned in Step "11" on page 469, and to fill the password field appropriately (for instructions, see "Specifying Passwords in the Local Password File" on page 461).

    If you use the **package** utility to distribute a common version of the password file to all client machines, then you need to make the change only in the common version. See "Configuring Client Machines with the package Program" on page 389.

Improving Password and Authentication Security

AFS provides several optional features than can help to protect your cell's filespace against unauthorized access. The following list summarizes them, and instructions follow.

- Limit the number of consecutive failed login attempts.

One of the most common ways for an unauthorized user to access your filespace is to guess an authorized user's password. This method of attack is most dangerous if the attacker can use many login processes in parallel or use the RPC interfaces directly.

To protect against this type of attack, use the **-attempts** argument to the **kas setfields** command to limit the number of times that a user can consecutively fail to enter the correct password when using either an AFS-modified login utility or the **klog** command. When the limit is exceeded, the Authentication Server locks the user's Authentication Database entry (disallows authentication attempts) for a period of time that you define with the **-locktime** argument to the **kas setfields** command. If desired, system administrators can use the **kas unlock** command to unlock the entry before the complete lockout time passes.

In certain circumstances, the mechanism used to enforce the number of failed authentication attempts can cause a lockout even though the number of failed attempts is less than the limit set by the **-attempts** argument. Client-side authentication programs such as **klog** and an AFS-modified login utility normally choose an Authentication Server at random for each authentication attempt, and in case of a failure are likely to choose a different Authentication Server for the next attempt. The Authentication Servers running on the various database server machines do not communicate with each other about how many times a user has failed to provide the correct password to them. Instead, each Authentication Server maintains its own separate copy of the auxiliary database file **kaserverauxdb** (located in the **/usr/afs/local** directory by default), which records the number of consecutive authentication failures for each user account and the time of the most recent failure. This implementation means that on average each Authentication Server knows about only a fraction of the total number of failed attempts. The only way to avoid allowing more than the number of attempts set by the **-attempts** argument is to have each Authentication Server allow only some fraction of the total. More specifically, if the limit on failed attempts is *f*, and the number of Authentication Servers is *S*, then each Authentication Server can only permit a number of attempts equal to *f* divided by *S* (the Ubik synchronization site for the Authentication Server tracks any remainder, *f mod S*).

Normally, this implementation does not reduce the number of allowed attempts to less than the configured limit (*f*). If one Authentication Server refuses an attempt, the client contacts another instance of the server, continuing until either it successfully authenticates or has contacted all of the servers. However, if one or more of the Authentication Server processes is unavailable, the limit is effectively reduced by a percentage equal to the quantity *U* divided by *S*, where *U* is the number of unavailable servers and *S* is the number normally available.

To avoid the undesirable consequences of setting a limit on failed authentication attempts, note the following recommendations:

- Do not set the **-attempts** argument (the limit on failed authentication attempts) too low. A limit of nine failed attempts is recommended for regular user accounts, to allow three failed attempts per Authentication Server in a cell with three database server machines.
- Set fairly short lockout times when including the **-locktime** argument. Although guessing passwords is a common method of attack, it is not a very sophisticated one. Setting a lockout time can help discourage attackers, but excessively long times are likely to be more of a burden to authorized users than to potential attackers. A lockout time of 25 minutes is recommended for regular user accounts.
- Do not assign an infinite lockout time on an account (by setting the **-locktime** argument to **0** [zero]) unless there is a highly compelling reason. Such accounts almost inevitably become locked at some point, because each Authentication Server never resets the account's failure counter in its copy of

the **kaauxdb** file (in contrast, when the lockout time is not infinite, the counter resets after the specified amount of time has passed since the last failed attempt to that Authentication Server). Furthermore, the only way to unlock an account with an infinite lockout time is for an administrator to issue the **kas unlock** command. It is especially dangerous to set an infinite lockout time on an administrative account; if all administrative accounts become locked, the only way to unlock them is to shut down all instances of the Authentication Server and remove the **kaauxdb** file on each.

In summary, the recommended limit on authentication attempts is nine and lockout time 25 minutes.

- Limit password lifetime.

  The longer a password is in use, the more time an attacker has to try to learn it. To protect against this type of attack, use the **-pwexpires** argument to the **kas setfields** command to limit how many days a user's password is valid. The user becomes unable to authenticate with AFS after the password expires, but has up to 30 days to use the **kpasswd** command to set a new password. After the 30 days pass, only an administrator who has the `ADMIN` flag on the Authentication Database entry can change the password.

  If you set a password lifetime, many AFS-modified login utilities (but not the **klog** command) set the PASSWORD_EXPIRES environment variable to the number of days remaining until the password expires. A setting of zero means that the password expires today. If desired, you can customize your users' login scripts to display the number of days remaining before expiration and even prompt for a password change when a small number of days remain before expiration.

- Prohibit reuse of passwords.

  Forcing users to select new passwords periodically is not effective if they simply set the new password to the current value. To prevent a user from setting a new password to a string similar to any of the last 20 passwords, use the **-reuse** argument to the **kas setfields** command.

  If you prohibit password reuse and the user specifies an excessively similar password, the Authentication Server generates the following message to reject it:

  ```
  Password was not changed because it seems like a reused password
  ```

  A persistent user can try to bypass this restriction by changing the password 20 times in quick succession (or running a script to do so). If you believe this is likely to be a problem, you can include the **-minhours** argument to the **kaserver** initialization command (for details, see the command's reference page in the *IBM AFS Administration Reference*. If the user attempts to change passwords too frequently, the following message appears.

  ```
  Password was not changed because you changed it too recently; see
  your systems administrator
  ```

- Check the quality of new passwords.

  You can impose a minimum quality standard on passwords by writing a script or program called **kpwvalid**. If the **kpwvalid** file exists, the **kpasswd** and **kas setpassword** command interpreters invoke it to check a new password. If the password does not comply with the quality standard, the **kpwvalid** program returns an appropriate code and the command interpreter rejects the password.

The **kpwvalid** file must be executable, must reside in the same AFS directory as the **kpasswd** and **kas** binaries, and its directory's ACL must grant the **w** (**write**) permission only to the **system:administrators** group.

If you choose to write a **kpwvalid** program, consider imposing standards such as the following.

- A minimum length
- Words found in the dictionary are prohibited
- Numbers, punctuation, or both must appear along with letters

The AFS distribution includes an example **kpwvalid** program. See the **kpwvalid** reference page in the *IBM AFS Administration Reference*.

To limit the number of consecutive failed authentication attempts

1. Issue the **kas setfields** command with the **-attempts** and **-locktime** arguments.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas setfields <name of user>  \
                  -admin <admin principal to use for authentication>  \
                  -attempts <maximum successive failed login tries ([0..254])>  \
                  -locktime <failure penalty [hh:mm or minutes]>
   Administrator's (admin_user) password: <admin_password>
   ```

   where

   name of user

   Names the Authentication Database entry to edit.

   -admin

   Names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as the **admin** account. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

   -attempts

   Specifies the maximum consecutive number of times that a user can fail to provide the correct password during authentication (via the **klog** command or an AFS-modified login utility) before the Authentication Server refuses further attempts for the amount of time specified by the **-locktime** argument. The range of valid values is **0** (zero) through **254**. If you omit this argument or specify **0**, the Authentication Server allows an unlimited number of failures.

-locktime

Specifies how long the Authentication Server refuses authentication attempts after the user exceeds the failure limit specified by the **-attempts** argument.

Specify a time in either hours and minutes (hh:mm) or minutes only (mm), from the range **01** (one minute) through **36:00** (36 hours). The **kas** command interpreter automatically reduces any larger value to 36:00 and also rounds up each nonzero value to the next-higher multiple of 8.5 minutes.

It is best not to provide a value of **0** (zero), especially on administrative accounts, because it sets an infinite lockout time. An administrator must always issue the **kas unlock** command to unlock such an account.

To unlock a locked user account

1. Issue the **kas** command to enter interactive mode.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas -admin <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ka>
   ```

   where **-admin** names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

2. Issue the **(kas) examine** command to verify that the user's account is in fact locked, as indicated by the message shown:

   ```
   ka> examine <name of user>
   User is locked until time
   ```

3. Issue the **(kas) unlock** command to unlock the account.

   ```
   ka> unlock <authentication ID>
   ```

   where

u

Is the shortest acceptable abbreviation of **unlock**.

authentication ID

Names the Authentication Database entry to unlock.

To set password lifetime

1. Issue the **kas setfields** command with the **-pwexpires** argument.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas setfields <name of user>  \
                   -pwexpires <number days password is valid  [0..254])>  \
                   -admin <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ```

   where

   name of user

   Specifies the Authentication Database entry on which to impose a password expiration.

   -pwexpires

   Sets the number of days after the user's password was last changed that it remains valid. Provide an integer from the range **1** through **254** to specify the number of days until expiration.

   When the password becomes invalid (expires), the user is unable to authenticate, but has 30 more days in which to issue the **kpasswd** or **kas setpassword** command to change the password (after that, only an administrator can change it). Note that the clock starts at the time the password was last changed, not when the **kas setfields** command is issued. To avoid retroactive expiration, have the user change the password just before issuing the command.

   -admin

   Names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

To prohibit reuse of passwords

1. Issue the **kas setfields** command with the **-reuse** argument.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas setfields <name of user> -reuse < permit password reuse (yes/no)>  \
                   -admin <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ```

   where

   name of user

   Names the Authentication Database entry for which to set the password reuse policy.

   -reuse

   Specifies whether the Authentication Server allows reuse of passwords similar to any of the user's last 20 passwords. Specify the value **no** to prohibit reuse, or the value **yes** to reinstate the default of allowing password reuse.

   -admin

   Names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

Changing AFS Passwords

After setting an initial password during account creation, you normally do not need to change user passwords, since they can use the **kpasswd** command themselves by following the instructions in the *IBM AFS User Guide*. In the rare event that a user forgets the password or otherwise cannot log in, you can use the **kas setpassword** command to set a new password.

If entries in the local password file (**/etc/passwd** or equivalent) have actual scrambled passwords in their password field, remember to change the password there also. For further discussion, see "Specifying Passwords in the Local Password File" on page 461.

To change an AFS password

1. Issue the **kas setpassword** command to change the password. To avoid having the new password echo visibly on the screen, omit the **-new_password** argument; instead enter the password at the

prompts that appear when you omit the argument, as shown.

The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the ADMIN flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

```
% kas setpassword <name of user>  \
                  -admin <admin principal to use for authentication>
Administrator's (admin_user) password: <admin_password>
new_password: <new_password>
Verifying, please re-enter new_password: <new_password>
```

where

sp

Is an acceptable alias for **setpassword** (**setp** is the shortest acceptable abbreviation).

name of user

Names the Authentication Database entry for which to set the password.

-admin

Names an administrative account that has the ADMIN flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

new_password

Specifies the user's new password. It is subject to the restrictions imposed by the **kpwvalid** program, if you use it.

Displaying and Setting the Quota on User Volumes

User volumes are like all other volumes with respect to quota. Each new AFS volume has a default quota of 5000 KB, unless you use the **-maxquota** argument to the **vos create** command to set a different quota. You can also use either of the following commands to change quota at any time:

- **fs setquota**
- **fs setvol**

You can use any of the three following commands to display a volume's quota:

- **fs quota**

- **fs listquota**
- **fs examine**

For instructions, see "Setting and Displaying Volume Quota and Current Size" on page 176.

Changing Usernames

By convention, many components of a user account incorporate the username, including the Protection and Authentication Database entries, the volume name and the home directory name. When changing a username, it is best to maintain consistency by changing the names of all components, so the procedure for changing a username has almost as many steps as the procedure for creating a new user account.

To change a username

1. Authenticate as an AFS identity with all of the following privileges. In the conventional configuration, the **admin** user account has them, or you possibly have a personal administrative account. (To increase cell security, it is best to create special privileged accounts for use only while performing administrative procedures; for further discussion, see "An Overview of Administrative Privilege" on page 531.) If necessary, issue the **klog** command to authenticate.

   ```
   % klog admin_user
   Password: <admin_password>
   ```

   The following list specifies the necessary privileges and indicates how to check that you have them.

   - Membership in the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

     ```
     % pts membership system:administrators
     ```

   - Inclusion in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

     ```
     % bos listusers <machine name>
     ```

   - The `ADMIN` flag on the Authentication Database entry. However, the Authentication Server performs its own authentication, so the following instructions direct you to specify an administrative identity on the **kas** command line itself.
   - The **a** (**administer**), **d** (**delete**), and **i** (**insert**) permissions on the ACL of the directory where you are removing the current mount point and creating a new one. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

     ```
     % fs listacl [<dir/file path>]
     ```

     Members of the **system:administrators** group always implicitly have the **a** (**administer**) and by default also the **l** (**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

2. Issue the **pts listowned** command to display the names of the groups the user owns. After you change the username in the Protection Database in Step "3" on page 479, you must issue the **pts rename** command to change each group's owner prefix to match the new name, because the Protection Server does not automatically make this change. For a complete description of the **pts listowned** command, see "Displaying Information from the Protection Database" on page 489.

   ```
   % pts listowned <user or group name or id>
   ```

3. Issue the **pts rename** command to change the user's name in the Protection Database.

   ```
   % pts rename <old name> <new name>
   ```

4. Issue the **pts rename** command to change the group names you noted in Step "2" on page 479, so that their owner prefix (the part of the group name before the colon) accurately reflects the owner's new name.

   Repeat the command for each group. Step "3" on page 479 details its syntax.

   ```
   % pts rename <old name> <new name>
   ```

5. Issue the **kas** command to enter interactive mode.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas -admin <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ka>
   ```

   where **-admin** names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

6. Issue the **(kas) delete** command to delete the user's existing Authentication Database entry.

   ```
   ka> delete <name of user>
   ```

   where

   del

   Is the shortest acceptable abbreviation for **delete**, or you can use the alias **rm**.

   name of user

   Names the Authentication Database entry to delete.

7. Issue the **(kas) create** command to create an Authentication Database entry for the new username. To avoid having the user's password echo visibly on the screen, do not include the **-initial_password** argument; instead enter the password at the prompts that appear in that case, as shown in the following syntax specification.

   ```
   ka> create  <name of user>
   initial_password: <password>
   Verifying, please re-enter initial_password: <password>
   ```

   where

   cr

   Is the shortest acceptable abbreviation for **create**.

   name of user

   Specifies the new username.

   password

   Specifies the password for the new user account. If the user is willing to tell you his or her current password, you can retain it. Otherwise, provide a string of eight characters or less to comply with the length restriction that some applications impose. Possible choices for an initial password include the username, a string of digits from a personal identification number such as the Social Security number, or a standard string such as **changeme**. Instruct the user to change the string to a truly secret password as soon as possible by using the **kpasswd** command as instructed in the *IBM AFS User Guide*.

8. Issue the **quit** command to leave interactive mode.

   ```
   ka> quit
   ```

9. Issue the **vos rename** command to change the name of the user's volume. For complete syntax, see "To rename a volume" on page 190.

   ```
   % vos rename  <old volume name>  <new volume name>
   ```

10. Issue the **fs rmmount** command to remove the existing mount point. For the directory argument, specify the read/write path to the mount point, to avoid the failure that results when you attempt to delete a mount point from a read-only volume.

    ```
    % fs rmmount <directory>
    ```

11. Issue the **fs mkmount** command to create a mount point for the volume's new name. Specify the read/write path to the mount point for the directory argument, as in the previous step. For complete syntax, see Step "6" on page 467 in "To create one user account with individual commands" on page 464.

    ```
    % fs mkmount <directory> <volume name>
    ```

12. If the changes you made in Step "10" on page 480 and Step "11" on page 480 are to a mount point that resides in a replicated volume, use the **vos release** command to release the volume, as described in "To replicate a read/write volume (create a read-only volume)" on page 142.

```
% vos release <volume name or ID>
```

Note: This step can be necessary even if the home directory's parent directory is not itself a mount point for a replicated volume (and is easier to overlook in that case). For example, the ABC Corporation template puts the mount points for user volumes in the **/afs/abc.com/usr** directory. Because that is a regular directory rather than a mount point, it resides in the **root.cell** volume mounted at the **/afs/abc.com** directory. That volume is replicated, so after changing it the administrator must issue the **vos release** command.

Removing a User Account

Before removing an account, it is best to make a backup copy of the user's home volume on a permanent storage medium such as tape. If you need to remove several accounts, it is probably more efficient to use the **uss delete** command instead; see "Deleting Individual Accounts with the uss delete Command" on page 448.

To remove a user account

1. Authenticate as an AFS identity with all of the following privileges. In the conventional configuration, the **admin** user account has them, or you possibly have a personal administrative account. (To increase cell security, it is best to create special privileged accounts for use only while performing administrative procedures; for further discussion, see "An Overview of Administrative Privilege" on page 531.) If necessary, issue the **klog** command to authenticate.

   ```
   % klog admin_user
   Password: <admin_password>
   ```

   The following list specifies the necessary privileges and indicates how to check that you have them.

   - Membership in the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

     ```
     % pts membership system:administrators
     ```

   - Inclusion in the **/usr/afs/etc/UserList** file. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

     ```
     % bos listusers <machine name>
     ```

   - The `ADMIN` flag on the Authentication Database entry. However, the Authentication Server performs its own authentication, so the following instructions direct you to specify an administrative identity on the **kas** command line itself.

- The **d** (**delete**) permission on the ACL of the directory where you are removing the user volume's mount point. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

  ```
  % fs listacl [<dir/file path>]
  ```

  Members of the **system:administrators** group always implicitly have the **a** (**administer**) and by default also the **l** (**lookup**) permission on every ACL and can use the **fs setacl** command to grant other rights as necessary.

2. **(Optional)** If it is possible you need to restore the user's account someday, note the username and AFS UID, possibly in a file designated for that purpose. You can later restore the account with its original AFS UID.
3. **(Optional)** Copy the contents of the user's volume to tape. You can use the **vos dump** command as described in "Dumping and Restoring Volumes" on page 183 or the AFS Backup System as described in "Backing Up Data" on page 251.
4. **(Optional)** If you intend to remove groups that the user owns from the Protection Database after removing the user's entry, issue the **pts listowned** command to display them. For complete instructions, see "Displaying Information from the Protection Database" on page 489.

   ```
   % pts listowned <user or group name or id>
   ```

5. (**Optional)** Issue the **pts delete** command to remove the groups the user owns. However, if it is likely that other users have placed the groups on the ACLs of directories they own, it is best not to remove them.

   ```
   % pts delete <user or group name or id>+
   ```

   where

   del

   Is the shortest acceptable abbreviation for **delete**.

   user or group name or id

   Specifies the name or AFS UID of each group displayed in the output from Step "4" on page 482.

6. Issue the **kas delete** command to remove the user's Authentication Database entry.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   ```
   % kas delete <name of user>  \
                -admin  <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ```

where

d

Is the shortest acceptable abbreviation for **delete**.

name of user

Names the Authentication Database entry to delete.

-admin

Names an administrative account that has the `ADMIN` flag on its Authentication Database entry, such as **admin**. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

7. Issue the **vos listvldb** command to display the site of the user's home volume in preparation for removing it. By convention, user volumes are named **user**.username.

   ```
   % vos listvldb <volume name or ID>
   ```

   where

   listvl

   Is the shortest acceptable abbreviation of **listvldb**.

   volume name or ID

   Specifies the volume's name or volume ID number.

8. Issue the **vos remove** command to remove the user's volume. It automatically removes the backup version of the volume, if it exists. It is not conventional to replicate user volumes, so the command usually also completely removes the volume's entry from the Volume Location Database (VLDB). If there are ReadOnly replicas of the volume, you must repeat the **vos remove** command to remove each one individually.

   ```
   % vos remove <machine name> <partition name> <volume name or ID>
   ```

   where

   remo

   Is the shortest acceptable abbreviation of **remove**.

   machine name

   Names the file server machine that houses the volume, as specified in the output from Step "7" on page 483.

partition name

Names the partition that houses the volume, as specified in the output from Step "7" on page 483.

volume name or ID

Specifies the volume's name or ID number.

9. Issue the **fs rmmount** command to remove the volume's mount point.

   If you mounted the user's backup volume as a subdirectory of the home directory, then this command is sufficient to unmount the backup version as well. If you mounted the backup version at an unrelated location in the filespace, repeat the **fs rmmount** command for it.

   ```
   % fs rmmount <directory>
   ```

   where

   rmm

   Is the shortest acceptable abbreviation of **rmmount**.

   directory

   Names the mount point for the volume's previous name (the former home directory). Partial pathnames are interpreted relative to the current working directory.

   Specify the read/write path to the mount point, to avoid the failure that results when you attempt to delete a mount point from a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "Mounting Volumes" on page 149.

10. Issue the **pts delete** command to remove the user's Protection Database entry. A complete description of this command appears in Step "5" on page 482.

    ```
    % pts delete <user or group name or id>
    ```

11. If the deleted user home directory resided in a replicated volume, use the **vos release** command to release the volume, as described in "To replicate a read/write volume (create a read-only volume)" on page 142.

    ```
    % vos release <volume name or ID>
    ```

    Note: This step can be necessary even if the home directory's parent directory is not itself a mount point for a replicated volume (and is easier to overlook in that case). For example, the ABC Corporation template puts the mount points for user volumes in the **/afs/abc.com/usr** directory. Because that is a regular directory rather than a mount point, it resides in the **root.cell**

volume mounted at the **/afs/abc.com** directory. That volume is replicated, so after changing it by deleting a mount point the administrator must issue the **vos release** command.

Chapter 14. Administering the Protection Database

This chapter explains how to create and maintain user, machine, and group entries in the Protection Database.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Display Protection Database entry	**pts examine**
Map user, machine or group name to AFS ID	**pts examine**
Display entry's owner or creator	**pts examine**
Display number of users or machines belonging to group	**pts examine**
Display number of groups user or machine belongs to	**pts examine**
Display group-creation quota	**pts examine**
Display entry's privacy flags	**pts examine**
Display members of group, or groups that user or machine belongs to	**pts membership**
Display groups that user or group owns	**pts listowned**
Display all entries in Protection Database	**pts listentries**
Create machine entry	**pts createuser**
Create group entry	**pts creategroup**
Add users and machines to groups	**pts adduser**
Remove users and machines from groups	**pts removeuser**
Delete machine or group entry	**pts delete**
Change a group's owner	**pts chown**
Change an entry's name	**pts rename**
Set group creation quota	**pts setfields**
Set entry's privacy flags	**pts setfields**
Display AFS ID counters	**pts listmax**
Set AFS ID counters	**pts setmax**

About the Protection Database

The Protection Database stores information about AFS users, client machines, and groups which the File Server process uses to determine whether clients are authorized to access AFS data.

To obtain authenticated access to an AFS cell, a user must have an entry in the cell's Protection Database. The first time that a user requests access to the data stored on a file server machine, the File Server on that machine contacts the Protection Server to request the user's *current protection subgroup* (*CPS*), which lists all the groups to which the user belongs. The File Server scans the access control list (ACL) of the directory that houses the data, looking for groups on the CPS. It grants access in accordance with the permissions that the ACL extends to those groups or to the user individually. (The File Server stores the CPS and uses it as long as the user has the same tokens. When a user's group membership changes, he or she must reauthenticate for the File Server to recognize the change.)

Only administrators who belong to the cell's **system:administrators** group can create user entries (the group is itself defined in the Protection Database, as discussed in "The System Groups" on page 488). Members of the **system:administrators** group can also create machine entries, which can then be used to control access based on the machine from which the access request originates. After creating a machine entry, add it to a Protection Database group and place the group on ACLs (a machine cannot appear on ACLs directly). A machine entry can represent a single machine or multiple machines with consecutive IP addresses as specified by a wildcard notation. For instructions, see "Creating User and Machine Entries" on page 495. Because all replicas of a volume share the same ACL (the one on the volume's root directory mount point), machine entries enable you to replicate the volume that houses a program's binary file while still complying with a machine-based license agreement as required by the program's manufacturer. See "Creating User and Machine Entries" on page 495.

A group entry is a list of user entries, machine entries, or both (groups cannot belong to other groups). Putting a group on an ACL is a convenient way to extend or deny access to a set of users without listing them on the ACL individually. Similarly, adding users to a group automatically grants them access to all files and directories for which the associated ACL lists that group. Both administrators and regular users can create groups.

The System Groups

In addition to the groups that users and administrators can create, AFS defines the following three system groups. The Protection Server creates them automatically when it builds the first version of a cell's Protection Database, and always assigns them the same AFS GIDs.

system:anyuser

Represents all users able to access the cell's filespace from the local and foreign cells, authenticated or not. Its AFS GID is **-101**. The group has no stable membership listed in the Protection Database. Accordingly, the **pts examine** command displays **0** in its `membership` field, and the **pts membership** command does not list any members for it.

Placing this group on an ACL is a convenient way to extend access to all users. The File Server automatically places this group on the CPS of any user who requests access to data stored on a file server machine. (Every unauthenticated user is assigned the identity **anonymous** and this group is the only entry on the CPS for **anonymous**.)

system:authuser

Represents all users who are able to access the cell's filespace from the local and foreign cells and who have successfully obtained an AFS token in the local cell (are authenticated). Its AFS GID is **-102**. Like the **system:anyuser** group, it has no stable membership listed in the Protection Database. Accordingly, the **pts examine** command displays **0** in its `membership` field, and the **pts membership** command does not list any members for it.

Placing this group on an ACL is therefore a convenient way to extend access to all authenticated users. The File Server automatically places this group on the CPS of any authenticated user who requests access to data stored on a file server machine.

system:administrators

Represents the small number of cell administrators authorized to issue privileged **pts** commands and the **fs** commands that set quota. The ACL on the root directory of every newly created volume grants all permissions to the group. Even if you remove that entry, the group implicitly retains the **a** (**administer**), and by default also the **l** (**lookup**), permission on every ACL. Its AFS GID is **-204**. For instructions on administering this group, see "Administering the system:administrators Group" on page 532.

Displaying Information from the Protection Database

This section describes the commands you can use to display Protection Database entries and associated information. In addition to name and AFS ID, the Protection Database stores the following information about each user, machine, or group entry.

- The entry's owner, which is the user or group of users who can administer the entry
- The entry's creator, which serves mostly as an audit trail
- A membership count, which indicates how many groups a user or machine belongs to, or how many members belong to a group
- A set of privacy flags, which control which users can administer or display information about the entry
- A group-creation quota, which defines how many groups a user can create
- A list of the groups to which a user or machine belongs, or of the users and machines that belong to a group
- A list of the groups that a user or group owns

To display a Protection Database entry

1. Verify that you belong to the **system:administrators** group, which enables you to display an entry regardless of the setting of its first (**s**) privacy flag. By default, any user can display a Protection Database entry. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts examine** command to display one or more Protection Database entries.

   ```
   % pts examine <user or group name or id>+
   ```

   where

e

Is the shortest acceptable abbreviation of **examine** (and **check** is an alias).

user or group name or id

Specifies the name or AFS ID of each entry to display. Precede any AFS GID with a hyphen (**-**) because it is a negative integer.

The output includes the following fields. Examples follow.

`Name`

Specifies the entry's name.

- For a user, this is the name used when authenticating with AFS and the name that appears on ACL entries.
- For a machine, this is the IP address of a single machine, or a wildcard notation that represents a group of machines with consecutive IP addresses, as described in "Creating User and Machine Entries" on page 495.
- For a group, this is the name that appears on ACL entries and in the list of groups output by the **pts membership** command. The names of *regular* groups have two parts, separated by a colon (**:**). The part before the colon indicates the group's owner, and the part after is the unique name. A *prefix-less* group's name does not have the owner prefix; only members of the **system:administrators** group can create prefix-less groups. For further discussion of group names, see "Creating Groups" on page 497.

`id`

Specifies the entry's unique AFS identification number. For user and machine entries, the AFS user ID (AFS UID) is a positive integer; for groups, the AFS group ID (AFS GID) is a negative integer. AFS UIDs and GIDs have the same function as their counterparts in the UNIX file system, but are used by the AFS servers and the Cache Manager only.

Normally, the Protection Server assigns an AFS UID or GID automatically when you create Protection Database entries. Members of the **system:administrators** group can specify an ID if desired. For further discussion, see "Creating User and Machine Entries" on page 495 and "Creating Groups" on page 497.

`owner`

Names the user or group who owns the entry and therefore can administer it (for more information about a group owning another group, see "Using Groups Effectively" on page 498). Other users possibly have administrative privileges, too, depending on the setting of the entry's privacy flags. For instructions on changing the owner, see "Changing a Group's Owner" on page 503.

`creator`

Names the user who created the entry, and serves as an audit trail. If the entry is deleted from the Protection Database, the creator's group creation quota increases by one, even if the creator no longer owns the entry; see "Setting Group-Creation Quota" on page 506.

The value `anonymous` in this field generally indicates that the entry was created when the Protection Server was running in no-authentication mode, probably during initial configuration of the cell's first file server machine. For a description of no-authentication mode, see "Managing Authentication and Authorization Requirements" on page 93.

`membership`

Specifies the number of groups to which the user or machine belongs, or the number of users or machines that belong to the group.

`flags`

Specifies who can display or change information in a Protection Database entry. The five flags, each representing a different capability, always appear in the same order.

- For user entries, the default value is `S----`, which indicates that anyone can issue the **pts examine** command on the entry, but only the user and members of the **system:administrators** group can perform any other action.
- For machine entries, the default value is `S----`, which indicates that anyone can issue the **pts examine** command on the entry, but only members of the **system:administrators** group can perform any other action.
- For group entries, the default value is `S-M--`, which indicates that anyone can issue the **pts examine** and **pts membership** commands on the entry, but only the group's owner and members of the **system:administrators** group can perform any other action.

For a complete description of possible values for the flags, see "Setting the Privacy Flags on Database Entries" on page 507.

`group quota`

Specifies how many more groups a user can create in the Protection Database. The value for a newly created user entry is 20, but members of the **system:administrators** group can issue the **pts setfields** command at any time to change the value; see "Setting Group-Creation Quota" on page 506.

Group creation quota has no meaning for a machine or group entry: the Protection Server recognizes the issuer of the **pts creategroup** command only as an authenticated user or as the **anonymous** user, never as a machine or group. The default value for group entries is 0 (zero), and there is no reason to change it.

The following examples show the output for a user called **pat**, a machine with IP address **192.12.108.133** and a group called **terry:friends**:

```
% pts examine pat
Name: pat, id: 1020, owner: system:administrators, creator: admin,
  membership: 12, flags: S----, group quota: 15.
% pts ex 192.12.108.133
Name: 192.12.108.133, id: 5151, owner: system:administrators, creator: admin,
  membership: 1, flags: S----, group quota: 20.
% pts examine terry:friends
Name: terry:friends, id: -567, owner: terry, creator: terry,
  membership: 12, flags: SOm--, group quota: 0.
```

To display group membership

1. Verify that you belong to the **system:administrators** group, which enables you to display an entry's group membership information regardless of the setting of its third (**m**) privacy flag. By default the owner and the user can display group membership for a user entry, the owner for a machine entry, and anyone for a group entry. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts membership** command to display the list of groups to which a user or machine belongs, or the list of users and machines that belong to a group.

   ```
   % pts membership <user or group name or id>+
   ```

   where

   m

   Is the shortest acceptable abbreviation of **membership**.

   user or group name or id

   Specifies the name or AFS UID of each user or machine for which to list the groups it belongs to, or the name or AFS GID of each group for which to list the members.

For user and machine entries, the output begins with the following string, and then each group appears on its own line:

```
Groups user_or_machine (id: AFS_UID) is a member of:
```

For group entries, the output begins with the following string, and then each member appears on its own line:

```
Members of group (id: AFS_GID) are:
```

For the system groups **system:anyuser** and **system:authuser**, the output includes the initial header string only, because these groups do not have a stable membership listed in their Protection Database entry. See "The System Groups" on page 488.

The following examples show the output for a user called **terry** and a group called **terry:friends**:

```
% pts mem terry
```

```
Groups terry (id: 5347) is a member of:
  pat:friends
  sales
  acctg:general
% pts mem terry:friends
Members of terry:friends (id: -567) are:
  pat
  smith
  johnson
```

To list the groups that a user or group owns

1. Verify that you belong to the **system:administrators** group, which enables you to display an entry's group ownership information regardless of the setting of its second (**o**) privacy flag. By default the owner can list the groups owned by group, and a user the groups he or she owns. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts listowned** command to list the groups owned by each user or group.

   ```
   % pts listowned <user or group name or id>+
   ```

   where

   listo

   Is the shortest acceptable abbreviation of **listowned**.

   user or group name or id

   Specifies the name or AFS UID of each user, or the name or AFS GID or each group, for which to list the groups owned.

The output begins with the following string, and then each group appears on its own line:

```
Groups owned by user_or_group (id: AFS_ID) are:
```

The following examples show the output for a user called **terry** and a group called **terry:friends**:

```
% pts listo terry
Groups owned by terry (id: 5347) are:
  terry:friends
  terry:co-workers
% pts listo terry:friends
Groups owned by terry:friends (id: -567) are:
  terry:pals
  terry:buddies
```

To display all Protection Database entries

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts listentries** command to display all Protection Database entries.

   ```
   % pts listentries [-users] [-groups]
   ```

   where

   liste

   Is the shortest acceptable abbreviation of **listentries**.

   -users

   Displays user and machine entries. The same output results if you omit both this flag and the **-groups** flag.

   -groups

   Displays group entries.

The output is a table that includes the following columns. Examples follow.

`Name`

Specifies the entry's name.

`ID`

Specifies the entry's AFS identification number. For user and machine entries, the AFS user ID (AFS UID) is a positive integer; for groups, the AFS group ID (AFS GID) is a negative integer.

`Owner`

Specifies the AFS ID of the user or group who owns the entry and therefore can administer it.

`Creator`

Specifies the AFS UID of the user who created the entry.

The following example is from the ABC Corporation cell. The issuer provides no options, so the output includes user and machine entries.

```
% pts listentries
Name                          ID  Owner Creator
anonymous                  32766   -204    -204
admin                          1   -204   32766
```

```
pat                          1000   -204       1
terry                        1001   -204       1
smith                        1003   -204       1
jones                        1004   -204       1
192.12.105.33                2000   -204       1
192.12.105.46                2001   -204       1
```

Creating User and Machine Entries

An entry in the Protection Database is one of the two required components of every AFS user account, along with an entry in the Authentication Database. It is best to create a Protection Database user entry only in the context of creating a complete user account, by using the **uss add** or **uss bulk** command as described in "Creating and Deleting User Accounts with the uss Command Suite" on page 413, or the **pts createuser** command as described in "Creating AFS User Accounts" on page 463.

You can also use the **pts createuser** command to create Protection Database machine entries, which can then be used to control access based on the machine from which the access request originates. After creating a machine entry, add it to a Protection Database group and place the group on ACLs (a machine cannot appear on ACLs directly). Because all replicas of a volume share the same ACL (the one on the volume's root directory mount point), you can replicate the volume that houses a program's binary file while still complying with a machine-based license agreement as required by the program's manufacturer. If you do not place any other entries on the ACL, then only users working on the designated machines can access the file.

Keep in mind that creating an ACL entry for a group with machine entries in it extends access to both authenticated and unauthenticated users working on the machine. However, you can deny access to unauthenticated users by omitting an entry for the **system:anyuser** group from the ACLs of the parent directories in the file's pathname. Conversely, if you want to enable unauthenticated users on the machine to access a file, then the ACL on every directory leading to it must include an entry for either the **system:anyuser** group or a group to which the machine entry belongs. For more information on the **system:anyuser** group, see "The System Groups" on page 488.

Because a machine entry can include unauthenticated users, it is best not to add both machine entries and user entries to the same group. In general, it is easier to use and administer nonmixed groups. A machine entry can represent a single machine, or multiple machines with consecutive IP addresses (that is, all machines on a network or subnet) specified by a wildcard notation. See the instructions in "To create machine entries in the Protection Database" on page 496.

By default, the Protection Server assigns the next available AFS UID to a new user or machine entry. It is best to allow this, especially for machine entries. For user entries, it makes sense to assign an AFS UID only if the user already has a UNIX UID that the AFS UID needs to match (see "Assigning AFS and UNIX UIDs that Match" on page 460). When automatically allocating an AFS UID, the Protection Server increments the `max user id` counter by one and assigns the result to the new entry. Use the **pts listmax** command to display the counter, as described in "Displaying and Setting the AFS UID and GID Counters" on page 509.

Do not reuse the AFS UIDs of users who have left your cell permanently or machine entries you have removed, even though doing so seems to avoid the apparent waste of IDs. When you remove a user or machine entry from the Protection Database, the **fs listacl** command displays the AFS UID associated with the former entry, rather than the name. If you then assign the AFS UID to a new user or machine, the new user or machine automatically inherits permissions that were granted to the previous possessor of the ID. To remove obsolete AFS UIDs from ACLs, use the **fs cleanacl** command described in "Removing Obsolete AFS IDs from ACLs" on page 527.

In addition to the name and AFS UID, the Protection Server records the following values in the indicated fields of a new user or machine's entry. For more information and instructions on displaying an entry, see "To display a Protection Database entry" on page 489.

- It sets the `owner` field to the **system:administrators** group, indicating that the group's members administer the entry.
- It sets the `creator` field to the username of the user who issued the **pts createuser** command (or the **uss add** or **uss bulk** command).
- It sets the `membership` field to **0** (zero), because the new entry does not yet belong to any groups.
- It sets the `flags` field to **S----**; for explanation, see "Setting the Privacy Flags on Database Entries" on page 507.
- It sets the `group quota` field to **20**, meaning that the new user can create 20 groups. This field has no meaning for machine entries. For further discussion, see "Setting Group-Creation Quota" on page 506.

To create machine entries in the Protection Database

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts createuser** command to create one or more machine entries.

   ```
   % pts createuser -name <user name>+
   ```

   where

   cu

   Is an alias for **createuser** (and **createu** is the shortest acceptable abbreviation).

   -name

   Specifies an IP address in dotted-decimal notation for each machine entry. An entry can represent a single machine or a set of several machines with consecutive IP addresses, using the wildcard notation described in the following list. The letters **W**, **X**, **Y**, and **Z** each represent an actual number value in the field:

   - **W.X.Y.Z** represents a single machine, for example **192.12.108.240**.
   - **W.X.Y.0** matches all machines whose IP addresses start with the first three numbers. For example, **192.12.108.0** matches both **192.12.108.119** and **192.12.108.120**, but does not match **192.12.105.144**.

- **W.X.0.0** matches all machines whose IP addresses start with the first two numbers. For example, the address **192.12.0.0** matches both **192.12.106.23** and **192.12.108.120**, but does not match **192.5.30.95**.
- **W.0.0.0** matches all machines whose IP addresses start with the first number in the specified address. For example, the address **192.0.0.0** matches both **192.5.30.95** and **192.12.108.120**, but does not match **138.255.63.52**.

Do not define a machine entry with the name **0.0.0.0** to match every machine. The **system:anyuser** group is equivalent.

The following example creates a machine entry that includes all of the machines in the **192.12** network.

```
% pts cu 192.12.0.0
```

Creating Groups

Before you can add members to a group, you must create the group entry itself. The instructions in this section explain how to create both regular and prefix-less groups:

- A *regular group*'s name is preceded by a prefix that indicates who owns the group, in the following format:

  owner_name**:**group_name

  Any user can create a regular group. Group names must always be typed in full, so a short group_name that indicates the group's purpose or its members' common interest is practical. Groups with names like **terry:1** and **terry:2** are less useful because their purpose is unclear. For more details on the required format for regular group names, see the instructions in "To create groups" on page 499.

- A *prefix-less group*, as its name suggests, has only one field in its name, equivalent to a regular group's group_name field.

  Only members of the **system:administrators** group can create prefix-less groups. For a discussion of their purpose, see "Using Prefix-Less Groups" on page 500.

By default, the Protection Server assigns the next available AFS GID to a new group entry, and it is best to allow this. When automatically allocating an AFS GID (which is a negative integer), the Protection Server decrements the `max group id` counter by one and assigns the result to the new group. Use the **pts listmax** command to display the counter, as described in "Displaying and Setting the AFS UID and GID Counters" on page 509.

In addition to the name and AFS GID, the Protection Server records the following values in the indicated fields of a new group's entry. See "To display a Protection Database entry" on page 489.

- It sets the `owner` field to the issuer of the **pts creategroup** command, or to the user or group specified by the **-owner** argument.
- It sets the `creator` field to the username of the user who issued the **pts creategroup** command.
- It sets the `membership` field to **0** (zero), because the group currently has no members.
- It sets the `flags` field to **S-M--**; for explanation, see "Setting the Privacy Flags on Database Entries" on page 507.
- It sets the `group quota` field to **0**, because this field has no meaning for group entries.

Using Groups Effectively

The main reason to create groups is to place them on ACLs, which enables you to control access for multiple users without having to list them individually on the ACL. There are three basic ways to use groups, each suited to a different purpose:

- *Private use*: you create a group and place it on the ACL of directories you own, without necessarily informing the group's members that they belong to it. Members notice only that they can or cannot access the directory in a certain way. You retain sole administrative control over the group, since you are the owner.

  The existence of the group and the identity of its members is not necessarily secret. Other users can use the **fs listacl** command and see the group's name on a directory's ACL, or use the **pts membership** command to list the groups they themselves belong to. You can set the group's third privacy flag to limit who can use the **pts membership** command to list the group's membership, but a member of the **system:administrators** group always can; see "Setting the Privacy Flags on Database Entries" on page 507.

- *Shared use*: you inform the group's members that they belong to the group, but you still remain the sole administrator. For example, the manager of a work group can create a group of all the members in the work group, and encourage them to use it on the ACLs of directories that house information they want to share with other members of the group.

  Note: If you place a group owned by someone else on your ACLs, the group's owner can change the group's membership without informing you. Someone new can gain or lose access in a way you did not intend and without your knowledge.

- *Group use*: you create a group and then use the **pts chown** command to assign ownership to a group, either another group or the group itself (the latter type is a self-owned group). You inform the members of the owning group that they all can administer the owned group.

  The main advantage of designating a group as an owner is that it spreads responsibility for administering a group among several people. A single person does not have to perform all administrative tasks, and if the original creator leaves the group, ownership does not have to be transferred.

However, everyone in the owner group can make changes that affect others negatively, such as adding or removing people from the group inappropriately or changing the group's ownership to themselves exclusively. These problems can be particularly sensitive in a *self-owned* group. Using an owner group works best if all the members know and trust each other; it is probably wise to keep the number of people in an owner group small.

To create groups

1. If creating a prefix-less group, verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts creategroup** command to create each group. All of the groups have the same owner.

   ```
   % pts creategroup  -name <group name>+ [-owner <owner of the group>]
   ```

   where

   cg

   Is an alias for **creategroup** (and **createg** is the shortest acceptable abbreviation).

   -name

   Names each group to create. The name can include up to 63 lowercase letters or numbers, but it is best not to include punctuation characters, especially those that have a special meaning to the shell.

   A prefix-less group name cannot include the colon (**:**), because it is used to separate the two parts of a regular group name:

   owner_name**:**group_name

   The Protection Server requires that the owner_name prefix of a regular group name accurately indicate the group's owner. By default, you are recorded as the owner, and the owner_name must be your AFS username. You can include the **-owner** argument to designate another AFS user, a regular group, or a prefix-less group as the owner, providing the required value in the owner_name field:

   - If the owner is a user, it must be the AFS username.
   - If the owner is another regular group, it must match the owning group's owner_name field. For example, if the owner is the group **terry:associates**, the owner field must be **terry**.
   - If the owner is a prefix-less group, it must be the owning group's name.

(For a discussion of why it is useful for a group to own another group, see "Using Groups Effectively" on page 498.)

-owner

Is optional and designates an owner other than the issuer of the command. Specify either an AFS username or the name of a regular or prefix-less group that already has at least one member. Do not include this argument if you want to make the group self-owned as described in "Using Groups Effectively" on page 498. For instructions, see "To create a self-owned group" on page 500.

Do not designate a machine as a group's owner. Because a machine cannot authenticate, there is no way for a machine to administer the group.

To create a self-owned group

1. Issue the **pts creategroup** command to create a group. Do not include the **-owner** argument, because you must own a group to reassign ownership. For complete instructions, see "To create groups" on page 499.

   ```
   % pts creategroup  <group name>
   ```

2. Issue the **pts adduser** command to add one or more members to the group (a group must already have at least one member before owning another group). For complete instructions, see "Adding and Removing Group Members" on page 501.

   ```
   % pts adduser -user <user name>+ -group <group name>+
   ```

3. Issue the **pts chown** command to assign group ownership to the group itself. For complete instructions, see "To change a group's owner" on page 504.

   ```
   % pts chown <group name> <new owner>
   ```

Using Prefix-Less Groups

Members of the **system:administrators** group can create prefix-less groups, which are particularly suitable for *group use*, which is described in "Using Groups Effectively" on page 498.

Suppose, for example, that the manager of the ABC Corporation's Accounting Department, user **smith**, creates a group that includes all of the corporation's accountants and places the group on the ACLs of directories that house departmental records. Using a prefix-less group rather than a regular group is appropriate for the following reasons:

- The fact that **smith** created and owns the group is irrelevant, and a regular group must be called **smith:acctg**. A prefix-less name like **acctg** is more appropriate.
- If another user (say **jones**) ever replaces **smith** as manager of the Accounting Department, **jones** needs to become the new owner of the group. If the group is a regular one, its owner_name prefix automatically changes to **jones**, but the change in the owner_name prefix does not propagate to any regular groups owned by the group. Someone must use the **pts rename** command to change each one's owner_name prefix from **smith** to **jones**.

A possible solution is to create an authentication account for a fictional user called **acctg** and make it the owner of regular groups which have **acctg** as their owner_name prefix. However, if the **acctg** account is also used for other purposes, then the number of people who need to know user **acctg**'s password is possibly larger than the number of people who need to administer the groups it owns.

A prefix-less group called **acctg** solves the problem of inappropriate owner names. The groups that it owns have **acctg** as their owner_name prefix, which more accurately reflects their purpose than having the manager's name there. Prefix-less groups are also more accountable than dummy authentication accounts. Belonging to the group enables individuals to exercise the permissions granted to the group on ACLs, but users continue to perform tasks under their own names rather than under the dummy username. Even if the group owns itself, only a finite number of people can administer the group entry.

Adding and Removing Group Members

Users and machines can be members of groups; groups cannot belong to other groups. Newly created groups have no members at all. To add them, use the **pts adduser** command; to remove them, use the **pts removeuser** command.

To add users and machines to groups

1. Verify that you belong to the **system:administrators** group, which enables you to add members to a group regardless of the setting of its fourth (**a**) privacy flag. By default the group's owner also has the necessary privilege. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts adduser** command to add one or more members to one or more groups.

   ```
   % pts adduser -user <user name>+ -group <group name>+
   ```

   where

   ad

   Is the shortest acceptable abbreviation of **adduser**.

-user

Specifies each username or machine IP address to add as a member of each group named by the **-group** argument. A group cannot belong to another group.

group name

Names each group to which to add the new members.

To remove users and machines from groups

1. Verify that you belong to the **system:administrators** group, which enables you to remove members from a group regardless of the setting of its fifth (**r**) privacy flag. By default the group's owner also has the necessary privilege. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts removeuser** command to remove one or more members from one or more groups.

   ```
   % pts removeuser -user  <user name>+  -group <group name>+
   ```

where

rem

Is the shortest acceptable abbreviation of **removeuser**.

-user

Specifies each user or machine IP address to remove from each group named by the **-group** argument.

-group

Names each group from which to remove members.

Deleting Protection Database Entries

It is best to delete a Protection Database user entry only if you are removing the complete user account. Use either the **uss delete** command as described in "Deleting Individual Accounts with the uss delete Command" on page 448, or the **pts delete** command as described in "Removing a User Account" on page 481.

To remove machine and group entries, use the **pts delete** command as described in this section. The operation has the following results:

- When you delete a machine entry, its name (IP address wildcard) is removed from groups.

- When you delete a group entry, its AFS GID appears on ACLs instead of the name. The *group-creation quota* of the user who created the group increases by one, even if the user no longer owns the group.

  To remove obsolete AFS IDs from ACLs, use the **fs cleanacl** command as described in "Removing Obsolete AFS IDs from ACLs" on page 527.

To delete Protection Database entries

1. Verify that you belong to the **system:administrators** group or own the group you are deleting. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts delete** command to delete one or more entries from the Protection Database.

   ```
   % pts delete <user or group name or id>+
   ```

   where

   del

   Is the shortest acceptable abbreviation of **delete**.

   user or group name or id

   Specifies the IP address or AFS UID of each machine or the name or AFS GID or each group to remove.

Changing a Group's Owner

For user and machine entries, the Protection Server automatically assigns ownership to the **system:administrators** group at creation time, and this cannot be changed. For group entries, you can change ownership. This transfers administrative responsibility for it to another user or group (for information on group ownership of other groups, see "Using Groups Effectively" on page 498).

When you create a regular group, its owner_name prefix must accurately reflect its owner, as described in "To create groups" on page 499:

- If the owner is a user, owner_name is the username.
- If the owner is a regular group, owner_name is the owning group's owner_name prefix.
- If the owner is a prefix-less group, owner_name is the owner group's name.

When you change a regular group's owner, the Protection Server automatically changes its owner_name prefix appropriately. For example, if the user **pat** becomes the new owner of the group **terry:friends**, its name automatically changes to **pat:friends**, both in the Protection Database and on ACLs.

However, the Protection Server does not automatically change the owner_name prefix of any regular groups that the group owns. To continue with the previous example, suppose that the group **terry:friends** owns the group **terry:pals**. When **pat** becomes the new owner of **terry:friends**, the name **terry:pals** does not change. To change the owner_name prefix of a regular group that is owned by another group (in the example, to change the group's name to **pat:pals**), use the **pts rename** command as described in "Changing a Protection Database Entry's Name" on page 505.

To change a group's owner

1. Verify that you belong to the **system:administrators** group or own the group for which you are changing the owner. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. **(Optional)** If you are changing the group's owner to another group (or to itself) and want to retain administrative privilege on the owned group, verify that you belong to the new owner group. If necessary, issue the **pts membership** command, which is fully described in "To display group membership" on page 492.

   ```
   % pts membership <user or group name or id>
   ```

   Use the **pts adduser** command to add yourself if necessary, as fully described in "To add users and machines to groups" on page 501.

   ```
   % pts adduser <user name> <group name>
   ```

3. Issue the **pts chown** command to change the group's owner.

   ```
   % pts chown <group name> <new owner>
   ```

   where

   cho

   Is the shortest acceptable abbreviation of **chown**.

   group name

   Specifies the current name of the group.

   new owner

   Names the user or group to become the group's owner.

4. **(Optional)** Issue the **pts listowned** command to display any groups that the group owns. As discussed in the introduction to this section, the **pts chown** command does not automatically change the owner_name prefix of any regular groups that a group owns.

   ```
   % pts listowned <user or group name or id>
   ```

   If you want to change their names to match the new owning group, use the **pts rename** command on each one, as described in "To change the name of a machine or group entry" on page 506.

   ```
   % pts rename <old name> <new name>
   ```

Changing a Protection Database Entry's Name

To change the name of a Protection Database entry, use the **pts rename** command. It is best to change a user entry's name only when renaming the entire user account, since so many components of the account (Authentication Database entry, volume name, home directory mount point, and so on) share the name. For instructions, see "Changing Usernames" on page 478. A machine entry's name maps to the actual IP address of one or more machine, so changing the entry's name is appropriate only if the IP addresses have changed.

It is likely, then, that most often you need to change group names. The following types of name changes are possible:

- Changing a regular group's name to another regular group name. The most common reason for this type of change is that you have used the **pts chown** command to change the owner of the group. That operation does not change the owner_name prefix of a regular group owned by the group whose name has been changed. Therefore, you must use the **pts rename** command to change it appropriately. For example, when user **pat** becomes the owner of the **terry:friends** group, its name changes automatically to **pat:friends**, but the name of a group it owns, **terry:pals**, does not change. Use the **pts rename** command to rename **terry:pals** to **pat:pals**. The Protection Server does not accept changes to the owner_name prefix that do not reflect the true ownership (changing **terry:pals** to **smith:pals** is not possible).

  You can also use the **pts rename** command to change the group_name portion of a regular group name, with or without changing the owner_name prefix.

  Both the group's owner and the members of the **system:administrators** group can change its name to another regular group name.

- Changing a regular group's name to a prefix-less name. If you change a group's name in this way, you must also use the **pts rename** command to change the name of any regular group that the group owns. Only members of the **system:administrators** group can make this type of name change.
- Changing a prefix-less name to another prefix-less name. As with other name changes, the owner_name prefix of any regular groups that the prefix-less group owns does not change automatically. You must issue the **pts rename** command on them to maintain consistency.

  Both the group's owner and the members of the **system:administrators** group can change its name to another prefix-less name.

- Changing a prefix-less name to a regular name. The owner_name prefix on the new name must accurately reflect the group's ownership. As with other name changes, the owner_name prefix of any regular groups that the prefix-less group owns does not change automatically. You must issue the **pts rename** command on them to maintain consistency.

  Only members of the **system:administrators** group can make this type of name change.

To change the name of a machine or group entry

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts rename** command to change the entry's name.

   ```
   % pts rename <old name> <new name>
   ```

   where

   ren

   Is the shortest acceptable abbreviation of **rename**.

   old name

   Specifies the entry's current name.

   new name

   Specifies the new name. If the new name is for a regular group, the owner_name prefix must correctly indicate the owner.

Setting Group-Creation Quota

To prevent abuse of system resources, the Protection Server imposes a group-creation quota that limits how many more groups a user can create. When a new user entry is created, the quota is set to 20, but members of the **system:administrators** group can use the **pts setfields** command to increase or decrease it at any time.

It is pointless to change group-creation quota for machine or group entries. It is not possible to authenticate as a group or machine and then create groups.

To display the group-creation quota, use the **pts examine** command to display a user entry's `group quota field`, as described in "To display a Protection Database entry" on page 489.

To set group-creation quota

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts setfields** command to specify how many more groups each of one or more users can create.

   ```
   % pts setfields -nameorid <user or group name or id>+  \
                   -groupquota <set limit on group creation>
   ```

   where

   setf

   Is the shortest acceptable abbreviation of **setfields**.

   -nameorid

   Specifies the name or AFS UID of each user for which to set group-creation quota.

   -groupquota

   Defines how many groups each user can create in addition to existing groups (in other words, groups that already exist do not count against the quota). The value you specify overwrites the current value, rather than incrementing it.

Setting the Privacy Flags on Database Entries

Members of the **system:administrators** group can always display and administer Protection Database entries in any way, and regular users can display and administer their own entries and any group entries they own. The *privacy flags* on a Protection Database entry determine who else can display certain information from the entry, and who can add and remove members in a group.

To display the flags, use the **pts examine** command as described in "To display a Protection Database entry" on page 489. The flags appear in the output's `flags` field. To set the flags, include the **-access** argument to the **pts setfields** command.

The five flags always appear, and always must be set, in the following order:

s

Controls who can issue the **pts examine** command to display the entry.

o

Controls who can issue the **pts listowned** command to display the groups that a user or group owns.

m

Controls who can issue the **pts membership** command to display the groups a user or machine belongs to, or which users or machines belong to a group.

a

Controls who can issue the **pts adduser** command to add a user or machine to a group. It is meaningful only for groups, but a value must always be set for it even on user and machine entries.

r

Controls who can issue the **pts removeuser** command to remove a user or machine from a group. It is meaningful only for groups, but a value must always be set for it even on user and machine entries.

Each flag can take three possible types of values to enable a different set of users to issue the corresponding command:

- A hyphen (**-**) designates the members of the **system:administrators** group and the entry's owner. For user entries, it designates the user in addition.
- The lowercase version of the letter applies meaningfully to groups only, and designates members of the group in addition to the individuals designated by the hyphen.
- The uppercase version of the letter designates everyone.

For example, the flags `SOmar` on a group entry indicate that anyone can examine the group's entry and display the groups that it owns, and that only the group's members can display, add, or remove its members.

The default privacy flags for user and machine entries are `S----`, meaning that anyone can display the entry. The ability to perform any other functions is restricted to members of the **system:administrators** group and the entry's owner (as well as the user for a user entry).

The default privacy flags for group entries are `S-M--`, meaning that all users can display the entry and the members of the group, but only the entry owner and members of the **system:administrators** group can perform other functions.

To set a Protection Database entry's privacy flags

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts setfields** command to set the privacy flags.

   ```
   % pts setfields <user or group name or id>+ -access <set privacy flags>
   ```

where

setf

Is the shortest acceptable abbreviation of **setfields**.

user or group name or id

Specifies the name or AFS UID of each user, the IP address or AFS UID of each machine, or the name or AFS GID of each group for which to set the privacy flags.

-access

Specifies the set of privacy flags to associate with each entry. Provide a value for each of the five flags, observing the following constraints:

- Provide a value for all five flags, even though the fourth and fifth flags are not meaningful for user and machine entries.
- For self-owned groups, the hyphen is equivalent to a lowercase letter, because all the members of a self-owned group own it.
- Set the first flag to lowercase **s** or uppercase **S** only. For user and machine entries, the Protection Server interprets the lowercase **s** as equivalent to the hyphen.
- Set the second flag to the hyphen (**-**) or uppercase **O** only. For groups, the Protection Server interprets the hyphen as equivalent to lowercase **o** (that is, members of a group can always list the groups that it owns).
- Set the third flag to the hyphen (**-**), lowercase **m**, or uppercase **M**. For user and machine entries, the lowercase **m** does not have a meaningful interpretation, because they have no members.
- Set the fourth flag to the hyphen (**-**), lowercase **a**, or uppercase **A**. Although this flag does not have a meaningful interpretation for user and machine entries (because they have no members), it must be set, preferably to the hyphen.
- Set the fifth flag to the hyphen (**-**) or lowercase **r** only. Although this flag does not have a meaningful interpretation for user and machine entries (because they have no members), it must be set, preferably to the hyphen.

Displaying and Setting the AFS UID and GID Counters

When you use the **pts createuser** command to create a user or machine entry in the Protection Database, the Protection Server by default automatically allocates an AFS user ID (AFS UID) for it; similarly, it allocates an AFS group ID (AFS GID) for each group entry you create with the **pts creategroup** command. It tracks the next available AFS UID (which is a positive integer) and AFS GID (which is a negative integer) with the `max user id` and `max group id` counters, respectively.

Members of the **system:administrators** group can include the **-id** argument to either **pts** creation command to assign a specific ID to a new user, machine, or group. It often makes sense to assign AFS UIDs explicitly when creating AFS accounts for users with existing UNIX accounts, as discussed in

"Assigning AFS and UNIX UIDs that Match" on page 416. It is also useful if you want to establish ranges of IDs that correspond to departmental affiliations (for example, assigning AFS UIDs from 300 to 399 to members of one department, AFS UIDs from 400 to 499 to another department, and so on).

To display the current value of the counters, use the **pts listmax** command. When you next create a user or machine entry and do not specify its AFS UID, the Protection Server increments the `max user id` counter by one and assigns that number to the new entry. When you create a new group and do not specify its AFS GID, the Protection Server decrements the `max group id` counter by one (makes it more negative), and assigns that number to the new group.

You can change the value of either counter, or both, in one of two ways:

- Directly, using the **pts setmax** command.
- Indirectly, by using the **-id** argument to the **pts createuser** command to assign an AFS UID that is larger than the `max user id` counter, or by using the **-id** to the **pts creategroup** command to assign an AFS GID that is less (more negative) than the max group id counter. In either case, the Protection Server changes the counter to the value of the **-id** argument. The Protection Server does not use the IDs between the previous value of the counter and the new one when allocating IDs automatically, unless you use the **pts setmax** command to move the counter back to its old value.

  If the value you specify with the **-id** argument is less than the `max user id` counter or greater (less negative) than the `max group id` counter, then the counter does not change.

To display the AFS ID counters

1. Issue the **pts listmax** command to display the counters.

   ```
   % pts listmax
   ```

   where **listm** is an acceptable abbreviation of **listmax**.

The following example illustrates the output's format. In this case, the next automatically assigned AFS UID is 5439 and AFS GID is -469.

```
% pts listmax
Max user id is 5438 and max group id is -468.
```

To set the AFS ID counters

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts setmax** command to set the `max user id` counter, the `max group id` counter, or both.

   ```
   % pts setmax [-group <group max>] [-user <user max>]
   ```

   where

   setm

   Is the shortest acceptable abbreviation of **setmax**.

   -group

   Specifies an integer one greater (less negative) than the AFS GID that the Protection Server is to assign to the next group entry. Because the value is a negative integer, precede it with a hyphen (**-**).

   -user

   Specifies an integer one less than the AFS UID that the Protection Server is to assign to the next user or machine entry.

Chapter 15. Managing Access Control Lists

To control access to a directory and all of the files in it, AFS associates an *access control list* (*ACL*) with it, rather than the mode bits that the UNIX file system (UFS) associates with individual files or directories. AFS ACLs provide more refined access control because there are seven access permissions rather than UFS's three, and there is room for approximately 20 user or group entries on an ACL, rather than just the three UFS entries (**owner**, **group**, and **other**).

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Examine access control list	**fs listacl**
Edit ACL's normal permissions section	**fs setacl**
Edit ACL's negative permissions section	**fs setacl** with **-negative** flag
Replace an ACL	**fs setacl** with **-clear** flag
Copy an ACL	**fs copyacl**
Remove obsolete AFS UIDs	**fs cleanacl**

Protecting Data in AFS

This section describes the main differences between the AFS and UFS file protection systems, discusses the implications of directory-level protections, and describes the seven access permissions.

Differences Between UFS and AFS Data Protection

The UFS mode bits data protection system and the AFS ACL system differ in the following ways:

- Protection at the file level (UFS) versus the directory level (AFS)

  UFS associates a set of nine mode bits with each file element, three (**rwx**) for each of the element's owner, owning group, and all other users. A similar set of mode bits on the file's directory applies to the file only in an oblique way.

  An AFS ACL instead protects all files in a directory in the same way. If a certain file is more sensitive than others, store it in a directory with a more restrictive ACL.

  Defining access at the directory level has important consequences:

  - The permissions on a directory's ACL apply to all of the files in the directory. When you move a file to a different directory, you effectively change the access permissions that apply to it to those on its new directory's ACL. Changing a directory's ACL changes the protection on all the files in it.
  - When you create a subdirectory, its initial ACL is created as a copy of its parent directory's ACL. You can then change the subdirectory's ACL independently. However, the parent directory's ACL continues to control access to the subdirectory in the following way: the parent directory's ACL

must grant the **l** (**lookup**) permission to a user (or a group the user belongs to) in order for the user to access the subdirectory at all.

In general, then, it is best to assign fairly liberal access permissions to high-level directories (including user home directories). In particular, it often makes sense to grant at least the **l** permission to the **system:anyuser** or **system:authuser** group on high-level directories. For further discussion, see "Using Groups on ACLs" on page 518.

- How the mode bits are interpreted

  Mode bits are the only file-protection system in UFS. AFS allows you to set the UNIX mode bits on a file in addition to the ACL on its directory, but it interprets them differently. See "How AFS Interprets the UNIX Mode Bits" on page 529.

- Three access permissions (UFS) versus seven (AFS)

  UFS defines three access permissions in the form of mode bits: **r** (**read**), **w** (**write**), and **x** (**execute**). AFS defines seven permissions, which makes access control more precise. For detailed descriptions, see "The AFS ACL Permissions" on page 515.
  a (**administer**)
  d (**delete**)
  i (**insert**)
  k (**lock**)
  l (**lookup**)
  r (**read**)
  w (**write**)

- Three defined users and groups (UFS) versus many (AFS)

  UFS controls access for one user and two groups by providing a set of mode bits for each: the user who owns the file or directory, a single defined group, and everyone who has an account on the system.

  AFS, in contrast, allows you to place many entries (individual users or groups) on an ACL, granting a different set of access permissions to each one. The number of possible entries is about 20, and depends on how much space each entry occupies in the memory allocated for the ACL itself.

  AFS defines two system groups, **system:anyuser** and **system:authuser**, which represent all users and all authenticated users, respectively; for further discussion, see "Using Groups on ACLs" on page 518. In addition, users can define their own groups in the Protection Database, consisting of individual users or machine IP addresses. Users who have the **a** permission on an ACL can create entries for the system groups as well as groups defined by themselves or other users. For information on defining groups, see "Administering the Protection Database" on page 487.

  When a user requests access to a file or directory, the File Server sums together all of the permissions that the relevant ACL extends to the user and to groups to which the user belongs. Placing group entries on ACLs therefore can control access for many more users than the ACL can accommodate as individual entries.

The AFS ACL Permissions

Functionally, the seven standard ACL permissions fall into two groups: one that applies to the directory itself and one that applies to the files it contains.

The Four Directory Permissions

The four permissions in this group are meaningful with respect to the directory itself. For example, the **i** (**insert**) permission does not control addition of data to a file, but rather creation of a new file or subdirectory.

The l (lookup) permission

This permission functions as something of a gate keeper for access to the directory and its files, because a user must have it in order to exercise any other permissions. In particular, a user must have this permission to access anything in the directory's subdirectories, even if the ACL on a subdirectory grants extensive permissions.

This permission enables a user to issue the following commands:

- The **ls** command to list the names of the files and subdirectories in the directory
- The **ls -ld** command to obtain complete status information for the directory element itself
- The **fs listacl** command to examine the directory's ACL

This permission does not enable a user to read the contents of a file in the directory, to issue the **ls -l** command on a file in the directory, or to issue the **fs listacl** command with the filename as the **-path** argument. Those operations require the **r** (**read**) permission which is described in "The Three File Permissions" on page 516.

Similarly, this permission does not enable a user to issue the **ls**, **ls -l**, **ls -ld**, or **fs listacl** commands against a subdirectory of the directory. Those operations require the **l** permission on the ACL of the subdirectory itself.

The i (insert) permission

This permission enables a user to add new files to the directory, either by creating or copying, and to create new subdirectories. It does not extend into any subdirectories, which are protected by their own ACLs.

The d (delete) permission

This permission enables a user to remove files and subdirectories from the directory or move them into other directories (assuming that the user has the **i** permission on the ACL of the other directories).

The a (administer) permission

This permission enables a user to change the directory's ACL. Members of the **system:administrators** group implicitly have this permission on every directory (that is, even if that group does not appear on the ACL). Similarly, the owner of a directory implicitly has this permission on its ACL and those of all directories below it that he or she owns.

The Three File Permissions

The three permissions in this group are meaningful with respect to files in a directory, rather than the directory itself or its subdirectories.

The r (read) permission

This permission enables a user to read the contents of files in the directory and to issue the **ls -l** command to stat the file elements.

The w (write) permission

This permission enables a user to modify the contents of files in the directory and to issue the **chmod** command to change their UNIX mode bits.

The k (lock) permission

This permission enables the user to run programs that issue system calls to lock files in the directory.

The Eight Auxiliary Permissions

AFS provides eight additional permissions that do not have a defined meaning, denoted by the uppercase letters **A**, **B**, **C**, **D**, **E**, **F**, **G**, and **H**.

You can write application programs that assign a meaning to one or more of the permissions, and then place them on ACLs to control file access by those programs. For example, you can modify a print program to recognize and interpret the permissions, and then place them on directories that house files that the program accesses. Use the **fs listacl** and **fs setacl** commands to display and set the auxiliary permissions on ACLs just like the standard seven.

Shorthand Notation for Sets of Permissions

You can combine the seven permissions in any way in an ACL entry, but certain combinations are more useful than others. Four of the more common combinations have corresponding shorthand forms. When using the **fs setacl** command to define ACL entries, you can provide either one or more of the individual letters that represent the permissions, or one of the following shorthand forms:

all

Represents all seven standard permissions (**rlidwka**).

none

Removes the entry from the ACL, leaving the user or group with no permissions.

read

Represents the **r** (**read**) and **l** (**lookup**) permissions.

write

Represents all permissions except **a** (**administer**): **rlidwk**.

Using Normal and Negative Permissions

ACLs enable you both to grant and to deny access to a directory and the files in it. To grant access, use the **fs setacl** command to create an ACL entry that associates a set of permissions with a user or group, as described in "Setting ACL Entries" on page 521. When you use the **fs listacl** command to display an ACL (as described in "Displaying ACLs" on page 519), such entries appear underneath the following header, which uses the term *rights* to refer to permissions:

```
Normal rights
```

There are two ways to deny access:

1. The recommended method is simply to omit an entry for the user or group from the ACL, or to omit the appropriate permissions from the entry. Use the **fs setacl** command to remove or edit an existing entry, using the instructions in "To add, remove, or edit normal ACL permissions" on page 521. In most circumstances, this method is enough to prevent access of certain kinds or by certain users. You must take care, however, not to grant the undesired permissions to any groups to which such users belong.
2. The more explicit method for denying access is to use the **-negative** flag to the **fs setacl** command to create an entry that associates *negative permissions* with the user or group; for instructions, see "To add, remove, or edit negative ACL permissions" on page 523. The output from the **fs listacl** command lists negative entries underneath the following header:

   ```
   Negative rights
   ```

   When determining what type of access to grant to a user, the File Server first compiles a set of permissions by examining all of the entries in the `Normal rights` section of the ACL. It then subtracts any permissions associated with the user (or with groups to which the user belongs) on the

`Negative rights` section of the ACL. Therefore, negative permissions always cancel out normal permissions.

Using negative permissions reverses the usual semantics of the **fs setacl** command, introducing the potential for confusion. In particular, combining the **none** shorthand and the **-negative** flag constitutes a double negative: by removing an entry from the `Negative rights` section of the ACL, you enable a user once again to obtain permissions via entries in the `Normal rights` section. Combining the **all** shorthand with the **-negative** flag explicitly denies all permissions.

Note also that it is pointless to create an entry in the `Negative rights` section if an entry in the `Normal rights` section grants the denied permissions to the **system:anyuser** group. In this case, users can obtain the permissions simply by using the **unlog** command to discard their tokens. When they do so, the File Server recognizes them as the **anonymous** user, who belongs to the **system:anyuser** group but does not match the entries on the `Negative rights` section of the ACL.

Using Groups on ACLs

As previously mentioned, placing a group entry on an ACL enables you to control access for many users at once. You can grant a new user access to many files and directories simply by adding the user to a group that appears on the relevant ACLs. You can also create groups of machines, in which case any user logged on to the machine obtains the access that is granted to the group. On directories where they have the **a** permission on the ACL, users can define their own groups and can create ACL entries for any groups, not just groups that they create or own themselves. For instructions on creating groups of users or machines, and a discussion of the most effective ways to use different types of groups, see "Administering the Protection Database" on page 487.

AFS also defines the following two system groups, which can be very useful on ACLs because they potentially represent a large group of people. For more information about these groups, see "The System Groups" on page 488.

system:anyuser

Includes anyone who can access the cell's file tree, including users who have logged in as the local superuser **root**, have connected to a local machine from somewhere outside the cell, and AFS users who belong to a foreign cell. This group includes users who do not have tokens that are valid for the local AFS servers; the servers recognize them as the user **anonymous**.

Note that creating an ACL entry for this group is the only way to extend access to AFS users from foreign cells, unless you create local authentication accounts for them.

system:authuser

Includes all users who have a valid AFS token obtained from the local cell's authentication service.

It is particularly useful to grant the **l** (**lookup**) permission to the **system:anyuser** group on the ACL of most directories in the file system, especially at the upper levels. This permission enables users only to learn the names of files and subdirectories in a directory, but without it they cannot traverse their way through the directories in the path to a target file.

A slightly more restrictive alternative is to grant the **l** permission to the **system:authuser** group. If that is still not restrictive enough, you can grant the **l** to specific users or groups, which cannot exceed about 20 in number on a given ACL.

Another reason to grant certain permissions to the **system:anyuser** group is to enable the correct operation of processes that provide services such as printing and mail delivery. For example, in addition to the **l** permission, a print process possibly needs the **r** (**read**) permission in order to access the contents of files, and a mail delivery process possibly requires the **i** (**insert**) permission to deliver new pieces of mail.

The ACL on the root directory of every newly created volume grants all permissions to the **system:administrators** group. You can remove this entry if you wish, but members of the **system:administrators** group always implicitly have the **a** (**administer**), and by default also the **l**, permission on every directory's ACL. The **a** permission enables them to grant themselves other permissions explicitly when necessary. To learn about changing this default set of permissions, see "Administering the system:administrators Group" on page 532.

Displaying ACLs

To display the ACL associated with a file, directory or symbolic link, issue the **fs listacl** command. The output for a symbolic link displays the ACL that applies to its target file or directory, rather than the ACL on the directory that houses the symbolic link.

Note for AFS/DFS Migration Toolkit users: If the machine on which you issue the **fs listacl** command is configured to access a DCE cell's DFS filespace via the AFS/DFS Migration Toolkit, you can use the command to display the ACL on DFS files and directories. To display a DFS directory's Initial Container and Initial Object ACL instead of the regular one, include the **fs listacl** command's **-id** or **-if** flag. For instructions, see the *IBM AFS/DFS Migration Toolkit Administration Guide and Reference*. The **fs** command interpreter ignores the **-id** and **-if** flags if you include them when displaying an AFS ACL.

To display an ACL

1. Issue the **fs listacl** command.

   ```
   % fs listacl [<dir/file path>+]
   ```

   where

   la

   Is an acceptable alias for **listacl** (and **lista** is the shortest acceptable abbreviation).

dir/file path

Names one or more files or directories for which to display the ACL. For files, the output displays the ACL for its directory. If you omit this argument, the output is for the current working directory. Partial pathnames are interpreted relative to the current working directory. You can also use the following notation on its own or as part of a pathname:

.

(A single period). Specifies the current working directory.

..

(Two periods). Specifies the current working directory's parent directory.

*****

(The asterisk). Specifies each file and subdirectory in the current working directory. The ACL displayed for a file is always the same as for its directory, but the ACL for each subdirectory can differ.

The following error message indicates that you do not have the permissions needed to display an ACL. To specify a directory name as the dir/file path argument, you must have the **l** (**lookup**) permission on the ACL. To specify a filename, you must also have the **r** (**read**) permission on its directory's ACL.

```
fs: You don't have the required access permissions on 'dir/file path'
```

Members of the **system:administrators** group and the directory's owner (as reported by the **ls -ld** command) implicitly have the **a** (**administer**) permission on every directory's ACL, and can use the **fs setacl** command to grant themselves the required permissions; for instructions, see "Setting ACL Entries" on page 521.

The output for each file or directory specified as dir/file path begins with the following header to identify it:

```
Access list for  dir/file path is
```

The `Normal rights` header appears on the next line, followed by lines that each pair a user or group name and a set of permissions. The permissions appear as the single letters defined in "The AFS ACL Permissions" on page 515, and always in the order **rlidwka**. If there are any negative permissions, the `Negative rights` header appears next, followed by pairs of negative permissions.

The following example displays the ACL on user **terry**'s home directory in the ABC Corporation cell:

```
% fs la /afs/abc.com/usr/terry
Access list for /afs/abc.com/usr/terry is
Normal permissions:
   system:authuser rl
   pat rlw
   terry rlidwka
Negative permissions:
   terry:other-dept rl
   jones rl
```

where **pat**, **terry**, and **jones** are individual users, **system:authuser** is a system group, and **terry:other-dept** is a group that **terry** owns. The list of normal permissions grants all permissions to **terry**, the **r** (**read**), **l** (**lookup**), and **w** (**write**) permissions to **pat**, and the **r** and **l** permissions to the members of the **system:authuser** group.

The list of negative permissions denies the **r** and **l** permissions to **jones** and the members of the **terry:other-dept** group. These entries effectively prevent them from accessing **terry**'s home directory in any way, because they cancel out the **r** and **l** permissions extended to the **system:authuser** group, which is the only entry on the `Normal rights` section of the ACL that possibly applies to them.

Setting ACL Entries

To add, remove, or edit ACL entries, use the **fs setacl** command. By default, the command manipulates entries on the normal permissions section of the ACL. To manipulate entries on the negative permissions section, include the **-negative** flag.

You must have the **a** (**administer**) permission on an ACL to edit it. The owner of a directory (as reported by the **ls -ld**) command and members of the **system:administrators** group always implicitly have it on every ACL. By default, members of the **system:administrators** group also implicitly have the **l** (**lookup**) permission.

Note for AFS/DFS Migration Toolkit users: If the machine on which you issue the **fs setacl** command is configured to access a DCE cell's DFS filespace via the AFS/DFS Migration Toolkit, you can use the command to set the ACL on DFS files and directories. To set a DFS directory's Initial Container and Initial Object ACL instead of the regular one, include the **fs setacl** command's **-id** or **-if** flag. For instructions, see the *IBM AFS/DFS Migration Toolkit Administration Guide and Reference*. The **fs** command interpreter ignores the **-id** and **-if** flags if you include them when setting an AFS ACL.

To add, remove, or edit normal ACL permissions

1. Verify that you have the **a** (**administer**) permission on each directory for which you are editing the ACL. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

2. Issue the **fs setacl** command to edit entries in the normal permissions section of the ACL. To remove an entry, specify the **none** shorthand as the permissions. If an ACL entry already exists, the permissions you specify completely replace those in the existing entry.

   ```
   % fs setacl  -dir <directory>+ -acl <access list entries>+
   ```

   where

   sa

   Is an acceptable alias for **setacl** (and **seta** is the shortest acceptable abbreviation).

-dir

Names one or more directories to which to apply the ACL entries defined by the **-acl** argument. Partial pathnames are interpreted relative to the current working directory.

Specify the read/write path to each directory, to avoid the failure that results when you attempt to change a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

You can also use the following notation on its own or as part of a pathname:

.

(A single period). If used by itself, sets the ACL on the current working directory.

..

(Two periods). If used by itself, sets the ACL on the current working directory's parent directory.

*****

(The asterisk). Sets the ACL on each of the subdirectories in the current working directory. You must precede it with the **-dir** switch, since it potentially designates multiple directories. The **fs** command interpreter generates the following error message for each file in the directory:

```
fs: 'filename': Not a directory
```

If you specify only one directory or file name, you can omit the **-dir** and **-acl** switches.

-acl

Specifies one or more ACL entries, each of which pairs a user or group name and a set of permissions. Separate the pairs, and the two parts of each pair, with one or more spaces.

To define the permissions, provide either:

- One or more of the letters that represent the standard or auxiliary permissions (**rlidwka** and **ABCDEFGH**), in any order
- One of the four shorthand notations:
  - **all** (equals **rlidwka**)
  - **none** (removes the entry)
  - **read** (equals **rl**)
  - **write** (equals **rlidwk**)

For a more detailed description of the permissions and shorthand notations, see "The AFS ACL Permissions" on page 515.

On a single command line, you can combine user and group entries. You can also use individual letters in some pairs and the shorthand notations in other pairs, but cannot combine letters and shorthand notation within a single pair.

Either of the following examples grants user **pat** the **r** (**read**) and **l** (**lookup**) permissions on the ACL of the **notes** subdirectory in the issuer's home directory. They illustrate how it is possible to omit the **-dir** and **-acl** switches when you name only one directory.

```
% fs sa ~/notes pat rl
% fs sa ~/notes pat read
```

The following example edits the ACL for the current working directory. It removes the entry for the **system:anyuser** group, and adds two entries: one grants all permissions except **a** (**administer**) to the members of the **terry:colleagues** group and the other grants the **r** (**read**) and **l** (**lookup**) permissions to the **system:authuser** group. The command appears on two lines here only for legibility.

```
% fs  sa  -dir . -acl  system:anyuser  none  terry:colleagues  write  \
       system:authuser  rl
```

To add, remove, or edit negative ACL permissions

1. Verify that you have the **a** (**administer**) permission on each directory for which you are editing the ACL. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

2. Issue the **fs setacl** command with the **-negative** flag to edit entries in the negative permissions section of the ACL. To remove an entry, specify the **none** shorthand as the permissions. If an ACL entry already exists for a user or group, the permissions you specify completely replace those in the existing entry.

   ```
   % fs setacl -dir <directory>+ -acl <access list entries>+  -negative
   ```

   where

   sa

   Is an acceptable alias for **setacl** (and **seta** is the shortest acceptable abbreviation).

   -dir

   Names one or more directories to which to apply the negative ACL entries defined by the **-acl** argument. Specify the read/write path to each directory, to avoid the failure that results when you attempt to change a read-only volume. For a detailed description of acceptable values, see "To add, remove, or edit normal ACL permissions" on page 521.

-acl

Specifies one or more ACL entries, each of which pairs a user or group name and a set of permissions. Separate the pairs, and the two parts of each pair, with one or more spaces. For a detailed description of acceptable values, see "To add, remove, or edit normal ACL permissions" on page 521. Keep in mind that the usual meaning of each permission is reversed.

-negative

Places the entries defined by the **-acl** argument on the negative permissions section of the ACL for each directory named by the **-dir** argument.

The following example denies user **pat** the **w** (**write**) and **d** (**delete**) permissions for the **project** subdirectory of the current working directory.

```
% fs sa project pat wd -neg
```

Completely Replacing an ACL

It is sometimes simplest to clear an ACL completely before defining new permissions on it, for instance if the mix of normal and negative permissions makes it difficult to understand how their interaction affects a user's access to the directory. To clear an ACL completely while you define new entries, include the **-clear** flag on the **fs setacl** command. When you include this flag, you can create entries on either the normal permissions or the negative permissions section of the ACL, but not on both at once.

Remember to create an entry that grants appropriate permissions to the directory's owner. The owner implicitly has the **a** (**administer**) permission required to replace a deleted entry, but the effects of a missing ACL entry (particularly the lack of the **lookup** permission) can be so confusing that it becomes difficult for the owner to realize that the missing entry is causing the problems.

To replace an ACL completely

1. Verify that you have the **a** (**administer**) permission on each directory for which you are editing the ACL. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

2. Issue the **fs setacl** command with the **-clear** flag to clear the ACL completely before setting either normal or negative permissions. Because you need to grant the owner of the directory all permissions, it is better in most cases to set normal permissions at this point.

   ```
   % fs setacl -dir <directory>+ -acl <access list entries>+ -clear  \
               [-negative]
   ```

   where

sa

Is an acceptable alias for **setacl** (and **seta** is the shortest acceptable abbreviation).

-dir

Names one or more directories to which to apply the negative ACL entries defined by the **-acl** argument. Specify the read/write path to each directory, to avoid the failure that results when you attempt to change a read-only volume. For a detailed description of acceptable values, see "To add, remove, or edit normal ACL permissions" on page 521.

-acl

Specifies one or more ACL entries, each of which pairs a user or group name and a set of permissions. Separate the pairs, and the two parts of each pair, with one or more spaces. Remember to grant all permissions to the owner of the directory. For a detailed description of acceptable values, see "To add, remove, or edit normal ACL permissions" on page 521.

-clear

Removes all entries from each ACL before creating the entries indicated by the **-acl** argument.

-negative

Places the entries defined by the **-acl** argument on the negative permissions section of each ACL.

Copying ACLs Between Directories

The **fs copyacl** command copies a source directory's ACL to one or more destination directories. It does not affect the source ACL at all, but changes each destination ACL as follows:

- If an entry on the source ACL does not exist on the destination ACL, the command copies it to the destination ACL.
- If an entry on the destination ACL does not also exist on the source ACL, the command does not remove it unless you include the **-clear** flag to overwrite the destination ACL completely.
- If an entry is on both ACLs, the command changes the permissions on the destination ACL entry to match the source ACL entry.

Note for AFS/DFS Migration Toolkit users: If the machine is configured to enable AFS users to access a DCE cell's DFS filespace via the AFS/DFS Migration Toolkit, then you can use the **fs copyacl** command to copy ACLs between DFS files and directories also. The command includes **-id** and **-if** flags for altering a DFS directory's Initial Container and Initial Object ACLs as well as its regular ACL; see the *IBM AFS/DFS Migration Toolkit Administration Guide and Reference*. You cannot copy ACLs between AFS and DFS directories, because they use different ACL formats. The **fs** command interpreter ignores the **-id** and **-if** flags if you include them when copying AFS ACLs.

To copy an ACL between directories

1. Verify that you have the **l** (**lookup**) permission on the source ACL and the **a** (**administer**) permission on each destination ACL. To identify the source directory by naming a file in it, you must also have the **r** (**read**) permission on the source ACL. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

2. Issue the **fs copyacl** command to copy a source ACL to the ACL on one or more destination directories. (The command appears here on two lines only for legibility.)

   ```
   % fs copyacl -fromdir <source directory> -todir <destination directory>+  \
                 [-clear]
   ```

   where

   co

   Is the shortest acceptable abbreviation for **copyacl**.

   -fromdir

   Names the source directory from which to copy the ACL. Partial pathnames are interpreted relative to the current working directory. If this argument names a file, the ACL is copied from its directory.

   -todir

   Names each destination directory to which to copy the source ACL. Partial pathnames are interpreted relative to the current working directory. Filenames are not acceptable.

   Specify the read/write path to each directory, to avoid the failure that results when you attempt to change a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

   -clear

   Completely overwrites each destination directory's ACL with the source ACL.

The following example copies the ACL from the current working directory's **notes** subdirectory to the **plans** subdirectory. The issuer does not include the **-clear** flag, so the entry for user **pat** remains on the **plans** directory's ACL although there is no corresponding entry on the **notes** directory's ACL.

```
% fs la notes plans
Access list for notes is
Normal permissions:
   terry rlidwka
   smith rl
   jones rl
```

```
Access list for plans is
Normal permissions:
   terry rlidwk
   pat rlidwk
% fs copyacl notes plans
% fs la notes plans
Access list for notes is
Normal permissions:
   terry rlidwka
   smith rl
   jones rl
Access list for plans is
Normal permissions:
   terry rlidwka
   pat rlidwk
   smith rl
   jones rl
```

Removing Obsolete AFS IDs from ACLs

When you remove a user or group entry from the Protection Database, the **fs listacl** command displays the user's AFS UID (or group's AFS GID) in ACL entries, rather than the name. In the following example, user **terry** has an ACL entry for the group **terry:friends** (AFS GID -567) on her home directory in the ABC Corporation cell, and then removes the group from the Protection Database.

```
% fs listacl /afs/abc.com/usr/terry
Access list for /afs/abc.com/usr/terry is
Normal permissions:
  terry:friends rlik
  system:anyuser l
  terry rlidwka
% pts delete terry:friends
% fs listacl /afs/abc.com/usr/terry
Access list for /afs/abc.com/usr/terry is
Normal permissions:
  -567 rlik
  system:anyuser l
  terry rlidwka
```

Leaving AFS IDs on ACLs serves no function, because the ID no longer corresponds to an active user or group. Furthermore, if the ID is ever assigned to a new user or group, then the new possessor of the ID gains access that the owner of the directory actually intended for the previous possessor. (Reusing AFS IDs is not recommended precisely for this reason.)

To remove obsolete AFS UIDs from ACLs, use the **fs cleanacl** command.

To clean obsolete AFS IDs from an ACL

1. Verify that you have the **a** (**administer**) permission on each directory for which you are cleaning the ACL. If necessary, issue the **fs listacl** command, which is fully described in "Displaying ACLs" on page 519.

   ```
   % fs listacl [<dir/file path>]
   ```

2. Issue the **fs cleanacl** command to remove entries for obsolete AFS IDs.

   ```
   % fs cleanacl [<dir/file path>+]
   ```

   where

   cl

   Is the shortest acceptable abbreviation of **cleanacl**.

   dir/file path

   Names each directory for which to clean the ACL. If this argument names a file, its directory's ACL is cleaned. Omit this argument to clean the current working directory's ACL.

   Specify the read/write path to each directory, to avoid the failure that results when you attempt to change a read-only volume. By convention, you indicate the read/write path by placing a period before the cell name at the pathname's second level (for example, **/afs/.abc.com**). For further discussion of the concept of read/write and read-only paths through the filespace, see "The Rules of Mount Point Traversal" on page 149.

   You can also use the following notation on its own or as part of a pathname:

   .

   (A single period). If used by itself, cleans the current working directory's ACL.

   ..

   (Two periods). If used by itself, cleans the ACL on the current working directory's parent directory.

   *****

   (The asterisk). Cleans the ACL of each of the subdirectories in the current working directory. However, if you use the asterisk and there are obsolete AFS IDs on any directory's ACL, the following error message appears for every file in the directory:

   ```
   fs: 'filename': Not a directory
   ```

If there are obsolete AFS IDs on a directory, the command interpreter displays its cleaned ACL under the following header.

```
Access list for directory is now
```

If a directory's ACL has no obsolete AFS IDs on it, the following message appears for each.

```
Access list for directory is fine.
```

How AFS Interprets the UNIX Mode Bits

Although AFS uses ACLs to protect file data rather than the mode bits that UFS uses, it does not ignore the mode bits entirely. When you issue the **chmod** command on an AFS file or directory, AFS changes the bits appropriately. To change a file's mode bits, you must have the AFS **w** (**write**) permission on the ACL of the file's directory. To change a directory's mode bits, you must have the **d** (**delete**), **i** (**insert**), and **l** (**lookup**) permissions on its ACL.

AFS also uses the UNIX mode bits as follows:

- It uses the initial bit to determine the element's type. This is the bit that appears first in the output from the **ls -l** command and shows the hyphen (**-**) for a file or the letter **d** for a directory.
- It does not use any of the mode bits on a directory.
- For a file, the first (owner) set of bits interacts with the ACL entries that apply to the file in the following way:
  - If the first **r** mode bit is not set, no one (including the owner) can read the file, no matter what permissions they have on the ACL. If the bit is set, users also need the **r** (**read**) and **l** permissions on the ACL of the file's directory to read the file.
  - If the first **w** mode bit is not set, no one (including the owner) can modify the file. If the **w** bit is set, users also need the **w** and **l** permissions on the ACL of the file's directory to modify the file.
  - There is no ACL permission directly corresponding to the **x** mode bit, but to execute a file stored in AFS, the user must also have the **r** and **l** permissions on the ACL of the file's directory.

Chapter 16. Managing Administrative Privilege

This chapter explains how to enable system administrators and operators to perform privileged AFS operations.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Display members of **system:administrators** group	**pts membership**
Add user to **system:administrators** group	**pts adduser**
Remove user from **system:administrators** group	**pts removeuser**
Display `ADMIN` flag in Authentication Database entry	**kas examine**
Set or remove `ADMIN` flag on Authentication Database entry	**kas setfields**
Display users in **UserList** file	**bos listusers**
Add user to **UserList** file	**bos adduser**
Remove user from **UserList** file	**bos removeuser**

An Overview of Administrative Privilege

A fully privileged AFS system administrator has the following characteristics:

- Membership in the cell's **system:administrators** group. See "Administering the system:administrators Group" on page 532.
- The `ADMIN` flag on his or her entry in the cell's Authentication Database. See "Granting Privilege for kas Commands: the ADMIN Flag" on page 534.
- Inclusion in the file **/usr/afs/etc/UserList** on the local disk of each AFS server machine in the cell. See "Administering the UserList File" on page 535.

This section describes the three privileges and explains why more than one privilege is necessary.

Note: Never grant any administrative privilege to the user **anonymous**, even when a server outage makes it impossible to mutually authenticate. If you grant such privilege, then any user who can access a machine in your cell can issue privileged commands. The alternative solution is to put the affected server machine into no-authentication mode and use the **-noauth** flag available on many commands to prevent mutual authentication attempts. For further discussion, see "Managing Authentication and Authorization Requirements" on page 93.

The Reason for Separate Privileges

Often, a cell's administrators require full administrative privileges to perform their jobs effectively. However, separating the three types of privilege makes it possible to grant only the minimum set of privileges that a given administrator needs to complete his or her work.

The **system:administrators** group privilege is perhaps the most basic, and most frequently used during normal operation (when all the servers are running normally). When the Protection Database is unavailable due to machine or server outage, it is not possible to issue commands that require this type of privilege.

The `ADMIN` flag privilege is separate because of the extreme sensitivity of the information in the Authentication Database, especially the server encryption key in the **afs** entry. When the Authentication Database is unavailable due to machine or server outage, it is not possible to issue commands that require this type of privilege.

The ability to issue privileged **bos** and **vos** command is recorded in the **/usr/afs/etc/UserList** file on the local disk of each AFS server machine rather than in a database, so that in case of serious server or network problems administrators can still log onto server machines and use those commands while solving the problem.

Administering the system:administrators Group

The first type of AFS administrative privilege is membership . Members of the **system:administrators** group in the Protection Database have the following privileges:

- Permission to issue all **pts** commands, which are used to administer the Protection Database. See "Administering the Protection Database" on page 487.
- Permission to issue the **fs setvol** and **fs setquota** commands, which set the space quota on volumes as described in "Setting and Displaying Volume Quota and Current Size" on page 176.
- Implicit **a** (**administer**) and by default **l** (**lookup**) permissions on the access control list (ACL) on every directory in the cell's AFS filespace. Members of the group can use the **fs setacl** command to grant themselves any other permissions they require, as described in "Setting ACL Entries" on page 521.

  You can change the ACL permissions that the File Server on a given file server machine implicitly grants to the members of the **system:administrators** group for the data in volumes that it houses. When you issue the **bos create** command to create and start the **fs** process on the machine, include the **-implicit** argument to the **fileserver** initialization command. For syntax details, see the **fileserver** reference page in the *IBM AFS Administration Reference*. You can grant additional permissions, or remove the **l** permission. However, the File Server always implicitly grants the **a** permission to members of the group, even if you set the value of the **-implicit** argument to **none**.

To display the members of the system:administrators group

1. Issue the **pts membership** command to display the **system:administrators** group's list of members. Any user can issue this command as long as the first privacy flag on the **system:administrators** group's Protection Database entry is not changed from the default value of uppercase `S`.

   ```
   % pts membership system:administrators
   ```

   where **m** is the shortest acceptable abbreviation of **membership**.

To add users to the system:administrators group

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts adduser** group to add one or more users.

   ```
   % pts adduser -user <user name>+ -group system:administrators
   ```

   where

   ad

   Is the shortest acceptable abbreviation of **adduser**.

   -user

   Names each user to add to the **system:administrators** group.

To remove users from the system:administrators group

1. Verify that you belong to the **system:administrators** group. If necessary, issue the **pts membership** command, which is fully described in "To display the members of the system:administrators group" on page 532.

   ```
   % pts membership system:administrators
   ```

2. Issue the **pts removeuser** command to remove one or more users.

   ```
   % pts removeuser -user <user name>+ -group system:administrators
   ```

   where

rem

Is the shortest acceptable abbreviation of **removeuser**.

-user

Names each user to remove from the **system:administrators** group.

Granting Privilege for kas Commands: the ADMIN Flag

Administrators who have the `ADMIN` flag on their Authentication Database entry can issue all **kas** commands, which enable them to administer the Authentication Database.

To check if the ADMIN flag is set

1. Issue the **kas examine** command to display an entry from the Authentication Database.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UFS) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin_username** argument (here abbreviated to **-admin**) to name a user identity that has the `ADMIN` flag on its Authentication Database entry.

   ```
   % kas examine <name of user>   \
                 -admin  <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ```

   where

   e

   Is the shortest acceptable abbreviation of **examine**.

   name of user

   Names the entry to display.

   -admin

   Names an administrative account with the `ADMIN` flag on its Authentication Database entry, such as the **admin** account. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

If the `ADMIN` flag is turned on, it appears on the first line, as in this example:

```
% kas e terry -admin admin
Administrator's (admin) password: <admin_password>
```

```
User data for terry (ADMIN)
  key version is 0, etc...
```

To set or remove the ADMIN flag

1. Issue the **kas setfields** command to turn on the `ADMIN` flag in an Authentication Database entry.

   The Authentication Server performs its own authentication rather than accepting your existing AFS token. By default, it authenticates your local (UNIX) identity, which possibly does not correspond to an AFS-privileged administrator. Include the **-admin** argument to name an identity that has the `ADMIN` flag on its Authentication Database entry. To verify that an entry has the flag, issue the **kas examine** command as described in "To check if the ADMIN flag is set" on page 534.

   The following command appears on two lines only for legibility.

   ```
   % kas setfields <name of user>  {ADMIN |  NOADMIN} \
                  -admin <admin principal to use for authentication>
   Administrator's (admin_user) password: <admin_password>
   ```

   where

   sf

   Is an alias for **setfields** (and **setf** is the shortest acceptable abbreviation).

   name of user

   Names the entry for which to set or remove the `ADMIN` flag.

   ADMIN | NOADMIN

   Sets or removes the `ADMIN` flag, respectively.

   -admin

   Names an administrative account with the `ADMIN` flag on its Authentication Database entry, such as the **admin** account. The password prompt echoes it as admin_user. Enter the appropriate password as admin_password.

Administering the UserList File

Inclusion in the file **/usr/afs/etc/UserList** on the local disk of each AFS server machine enables an administrator to issue commands from the indicated suites.

- The **bos** commands enable the administrator to manage server processes and the server configuration files that define the cell's database server machines, server encryption keys, and privileged users. See

"Administering Server Machines" on page 59 and "Monitoring and Controlling Server Processes" on page 105.

- The **vos** commands enable the administrator to manage volumes and the Volume Location Database (VLDB). See "Managing Volumes" on page 131.
- The **backup** commands enable the administrator to use the AFS Backup System to copy data to permanent storage. See "Configuring the AFS Backup System" on page 195 and "Backing Up and Restoring AFS Data" on page 241.

Although each AFS server machine maintains a separate copy of the file on its local disk, it is conventional to keep all copies the same. It can be confusing for an administrator to have the privilege on some machines but not others.

If your cell runs the United States edition of AFS and uses the Update Server to distribute the contents of the system control machine's **/usr/afs/etc** directory, then edit only the copy of the **UserList** file stored on the system control machine. If you have forgotten which machine is the system control machine, see "The Four Roles for File Server Machines" on page 68.

If your cell runs the international edition of AFS, or does not use a system control machine, then you must edit the **UserList** file on each server machine individually.

To avoid making formatting errors that can result in performance problems, never edit the **UserList** file directly. Instead, use the **bos adduser** or **bos removeuser** commands as described in this section.

To display the users in the UserList file

1. Issue the **bos listusers** command to display the contents of the **/usr/afs/etc/UserList** file.

   ```
   % bos listusers <machine name>
   ```

   where

   listu

   Is the shortest acceptable abbreviation of **listusers**.

   machine name

   Names an AFS server machine. In the normal case, any machine is acceptable because the file is the same on all of them.

To add users to the UserList file

1. Verify you are listed in the **/usr/afs/etc/UserList** file. If not, you must have a qualified administrator add you before you can add entries to it yourself. If necessary, issue the **bos listusers** command,

which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **bos adduser** command to add one or more users to the **UserList** file.

```
% bos adduser <machine name> <user names>+
```

where

addu

Is the shortest acceptable abbreviation of **adduser**.

machine name

Names the system control machine if you use the Update Server to distribute the contents of the **/usr/afs/etc** directory (possible only in cells running the United States edition of AFS). By default, it can take up to five minutes for the Update Server to distribute the changes, so newly added users must wait that long before attempting to issue privileged commands.

If you are running the international edition of AFS, or do not use the Update Server, repeat the command, substituting the name of each AFS server machine for machine name in turn.

user names

Specifies the username of each administrator to add to the **UserList** file.

To remove users from the UserList file

1. Verify you are listed in the **/usr/afs/etc/UserList** file. If not, you must have a qualified administrator add you before you can remove entries from it yourself. If necessary, issue the **bos listusers** command, which is fully described in "To display the users in the UserList file" on page 536.

```
% bos listusers <machine name>
```

2. Issue the **bos removeuser** command to remove one or more users from the **UserList** file.

```
% bos removeuser <machine name> <user names>+
```

where

removeu

Is the shortest acceptable abbreviation of **removeuser**.

machine name

Names the system control machine if you use the Update Server to distribute the contents of the **/usr/afs/etc** directory (possible only in cells running the United States edition of AFS). By default, it can take up to five minutes for the Update Server to distribute the change, so newly removed users can continue to issue privileged commands during that time.

If you are running the international edition of AFS, or do not use the Update Server, repeat the command, substituting the name of each AFS server machine for machine name in turn.

user names

Specifies the username of each administrator to add to the **UserList** file.

Appendix A. Managing the NFS/AFS Translator

The NFS(R)/AFS(R) Translator enables users working on NFS client machines to access, create and remove files stored in AFS. This chapter assumes familiarity with both NFS and AFS.

Summary of Instructions

This chapter explains how to perform the following tasks by using the indicated commands:

Mount directory on translator machine	**mount**
Examine value of @**sys** variable	**fs sysname**
Enable/disable reexport of AFS, set other parameters	**fs exportafs**
Assign AFS tokens to user on NFS client machine	**knfs**

Overview

The NFS/AFS Translator enables users on NFS client machines to access the AFS filespace as if they are working on an AFS client machine, which facilitates collaboration with other AFS users.

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

- Its AFS client functionality enables it to access the AFS filespace. The Cache Manager requests and caches files from AFS file server machines, and can even maintain tokens for NFS users, if you have made the configuration changes that enable NFS users to authenticate with AFS.
- Its NFS server functionality makes it possible for the translator machine to export the AFS filespace to NFS client machines. When a user on an NFS client machine mounts the translator machine's **/afs** directory (or one of its subdirectories, if that feature is enabled), access to AFS is immediate and transparent. The NFS client machine does not need to run any AFS software.

Enabling Unauthenticated or Authenticated AFS Access

By configuring the translation environment appropriately, you can provide either unauthenticated or authenticated access to AFS from NFS client machines. The sections of this chapter on configuring translator machines, NFS client machines, and AFS user accounts explain how to configure the translation environment appropriately.

- If you configure the environment for unauthenticated access, the AFS File Server considers the NFS users to be the user **anonymous**. They can access only those AFS files and directories for which the access control list (ACL) extends the required permissions to the **system:anyuser** group. They can issue only those AFS commands that do not require privilege, and then only if their NFS client machine is a system type for which AFS binaries are available and accessible by the **system:anyuser** group. Such users presumably do not have AFS accounts.

- If you configure the environment for authenticated access, you must create entries in the AFS Authentication and Protection Databases for the NFS users. The authentication procedure they use depends on whether the NFS client machine is a supported system type (one for which AFS binaries are available):
  - If AFS binaries are available for the NFS client machine, NFS users can issue the **klog** command on the NFS client machine. They can access the filespace and issue AFS commands to the same extent as authenticated users working on AFS client machines.
  - If AFS binaries are not available for the NFS client machine, NFS users must establish a connection with the translator machine (using the **telnet** utility, for example) and then issue the **klog** and **knfs** commands on the translator machine to make its Cache Manager use the tokens correctly while users work on the NFS client. They can access the AFS filespace as authenticated users, but cannot issue AFS commands. For instructions, see "Authenticating on Unsupported NFS Client Machines" on page 551.

Setting the AFSSERVER and AFSCONF Environment Variables

If you wish to enable your NFS users to issue AFS commands, you must define the AFSSERVER and AFSCONF environment variables in their command shell. This section explains the variables' function and outlines the various methods for setting them.

Issuing AFS commands also requires that the NFS client machine is a supported system type (one for which AFS binaries are available and accessible). Users working on NFS client machines of unsupported system types can access AFS as authenticated users, but they cannot issue AFS commands. It is not necessary to define the AFSSERVER and AFSCONF variables for such users. For instructions on using the **knfs** command to obtain authenticated access on unsupported system types, see "Authenticating on Unsupported NFS Client Machines" on page 551.

The AFSSERVER Variable

The AFSSERVER variable designates the AFS client machine that performs two functions for NFS clients:

- It acts as the NFS client's *remote executor* by executing AFS-specific system calls on its behalf, such as those invoked by the **klog** and **tokens** commands and by many commands in the AFS suites.
- Its stores the tokens that NFS users obtain when they authenticate with AFS. This implies that the remote executor machine and the translator machine must be the same if the user needs authenticated access to AFS.

The choice of remote executor most directly affects commands that display or change Cache Manager configuration, such as the **fs getcacheparms**, **fs getcellstatus**, and **fs setcell** commands. When issued on an NFS client, these commands affect the Cache Manager on the designated remote executor machine. (Note, however, that several such commands require the issuer to be logged into the remote executor's local file system as the local superuser **root**. The ability of NFS client users to log in as **root** is controlled by NFS, not by the NFS/AFS Translator, so setting the remote executor properly does not necessarily enable users on the NFS client to issue such commands.)

The choice of remote executor is also relevant for AFS commands that do not concern Cache Manager configuration but rather have the same result on every machine, such as the **fs** commands that display or set ACLs and volume quota. These commands take an AFS path as one of their arguments. If the Cache Manager on the remote executor machine mounts the AFS filespace at the **/afs** directory, as is conventional for AFS clients, then the pathname specified on the NFS client must begin with the string **/afs** for the Cache Manager to understand it. This implies that the remote executor must be the NFS client's primary translator machine (the one whose **/afs** directory is mounted at **/afs** on the NFS client).

The AFSCONF Variable

The AFSCONF environment variable names the directory that houses the **ThisCell** and **CellServDB** files to use when running AFS commands issued on the NFS client machine. As on an AFS client, these files determine the default cell for command execution.

For predictable performance, it is best that the files in the directory named by the AFSCONF variable match those in the **/usr/vice/etc** directory on the translator machine. If your cell has an AFS directory that serves as the central update source for files in the **/usr/vice/etc** directory, it is simplest to set the AFSCONF variable to refer to it. In the conventional configuration, this directory is called **/afs/**cellname**/common/etc**.

Setting Values for the Variables

To learn the values of the AFSSERVER and AFSCONF variables, AFS command interpreters consult the following three sources in sequence:

1. The current command shell's environment variable definitions
2. The **.AFSSERVER** or **.AFSCONF** file in the issuer's home directory
3. The **/.AFSSERVER** or **/.AFSCONF** file in the NFS client machine's root (/) directory. If the client machine is diskless, its root directory can reside on an NFS server machine.

(Actually, before consulting these sources, the NFS client looks for the **CellServDB** and **ThisCell** files in its own **/usr/vice/etc** directory. If the directory exists, the NFS client does not use the value of the AFSCONF variable. However, the **/usr/vice/etc** directory usually exists only on AFS clients, not NFS clients.)

As previously detailed, correct performance generally requires that the remote executor machine be the NFS client's primary translator machine (the one whose **/afs** directory is mounted at the **/afs** directory on the NFS client). The requirement holds for all users accessing AFS from the NFS client, so it is usually simplest to create the **.AFSSERVER** file in the NFS client's root directory. The main reason to create the file in a user's home directory or to set the AFSSERVER environment variable in the current command shell is that the user needs to switch to a different translator machine, perhaps because the original one has become inaccessible.

Similarly, it generally makes sense to create the **.AFSCONF** file in the NFS client's root directory. Creating it in the user's home directory or setting the AFSCONF environment variable in the current command shell is useful mostly when there is a reason to specify a different set of database server machines for the cell, perhaps in a testing situation.

Delayed Writes for Files Saved on NFS Client Machines

When an application running on an AFS client machine issues the **close** or **fsync** system call on a file, the Cache Manager by default performs a synchronous write of the data to the File Server. (For further discussion, see "AFS Implements Save on Close" on page 18 and "Enabling Asynchronous Writes" on page 385.)

To avoid degrading performance for the AFS users working on a translator machine, AFS does not perform synchronous writes for applications running on the translator machine's NFS clients. Instead, one of the Cache Manager daemons (the maintenance daemon) checks every 60 seconds for chunks in the cache that contain data saved on NFS clients, and writes their contents to the File Server. This does not guarantee that data saved on NFS clients is written to the File Server within 60 seconds, but only that the *maintenance daemon* checks for and begins the write of data at that interval.

Furthermore, AFS always ignores the **fsync** system call as issued on an NFS client. The call requires an immediate and possibly time-consuming response from the File Server, which potentially causes delays for other AFS clients of the File Server. NFS version 3 automatically issues the **fsync** system call directly after the **close** call, but the Cache Manager ignores it and handles the operation just like a regular **close**.

The delayed write mechanism means that there is usually a delay between the time when an NFS application issues the **close** or **fsync** system call on a file and the time when the changes are recorded at the File Server, which is when they become visible to users working on other AFS client machines (either directly or on its NFS clients). The delay is likely to be longer than for files saved by users working directly on an AFS client machine.

The exact amount of delay is difficult to predict. The NFS protocol itself allows a standard delay before saved data must be transferred from the NFS client to the NFS server (the translator machine). The modified data remains in the translator machine's AFS client cache until the maintenance daemon's next scheduled check for such data, and it takes additional time to transfer the data to the File Server. The maintenance daemon uses a single thread, so there can be additional delay if it takes more than 60 seconds to write out all of the modified NFS data. That is, if the maintenance daemon is still writing data at the time of the next scheduled check, it cannot notice any additional modified data until the scheduled time after it completes the long write operation.

The Cache Manager's response to the **write** system call is the same whether it is issued on an AFS client machine or on an NFS client of a translator machine: it records the modifications in the local AFS client cache only.

Configuring NFS/AFS Translator Machines

To act as an NFS/AFS translator machine, a machine must configured as follows:

- It must be an AFS client. Many system types supported as AFS clients can be translator machines. To learn about possible restrictions in a specific release of AFS, see the *IBM AFS Release Notes*.
- It must be an NFS server. The appropriate number of NFS server daemons (**nfsd** and others) depends on the anticipated NFS client load.
- It must export the local directory on which the AFS filespace is mounted, **/afs** by convention.

If users on a translator machine's NFS clients are to issue AFS commands, the translator machine must also meet the requirements discussed in "Configuring the Translator Machine to Accept AFS Commands" on page 543.

Loading NFS and AFS Kernel Extensions

The AFS distribution for system types that can act as NFS/AFS Translator machines usually includes two versions of the AFS kernel extensions file, one for machines where the kernel supports NFS server functionality, and one for machines not using NFS (the latter AFS kernel extensions file generally has the string **nonfs** in its name). A translator machine must use the NFS-enabled version of the AFS extensions file. On some system types, you select the appropriate file by moving it to a certain location, whereas on other system types you set a variable that results in automatic selection of the correct file. See the instructions in the *IBM AFS Quick Beginnings* for incorporating AFS into the kernel on each system type.

On many system types, NFS is included in the kernel by default, so it is not necessary to load NFS kernel extensions explicitly. On system types where you must load NFS extensions, then in general you must load them before loading the AFS kernel extensions. The *IBM AFS Quick Beginnings* describes how to incorporate the AFS initialization script into a machine's startup sequence so that it is ordered correctly with respect to the script that handles NFS.

In addition, the AFS extensions must be loaded into the kernel before the **afsd** command runs. The AFS initialization script included in the AFS distribution correctly orders the loading and **afsd** commands.

Configuring the Translator Machine to Accept AFS Commands

For users working on a translator machine's NFS clients to issue AFS commands, the **-rmtsys** flag must be included on the **afsd** command which initializes the translator machine's Cache Manager. The flag starts an additional daemon (the *remote executor* daemon), which executes AFS-specific system calls on behalf of NFS clients. For a discussion of the implications of NFS users issuing AFS commands, see "Setting the AFSSERVER and AFSCONF Environment Variables" on page 540.

The instructions in the IBM AFS Quick Beginnings for configuring the Cache Manager explain how to add options such as the **-rmtsys** flag to the **afsd** command in the AFS initialization script. On many system types, it is simplest to list the flag on the line in the script that defines the OPTIONS variable. The *remote executor daemon* does not consume many resources, so it is simplest to add it to the **afsd** command on every translator machine, even if not all users on the machine's NFS clients issue AFS commands.

Controlling Optional Translator Features

After an AFS client machine is configured as a translator machine, it by default exports the AFS filespace to NFS clients. You can disable and reenable translator functionality by using the **fs exportafs** command's **-start** argument. The command's other arguments control other aspects of translator behavior.

- The **-convert** argument controls whether the second and third (**group** and **other**) sets of UNIX mode bits on an AFS file or directory being exported to NFS are set to match the first (**owner**) mode bits. By default, the mode bits are set to match.

  Unlike AFS, NFS uses all three sets of mode bits when determining whether a user can read or write a file, even one stored in AFS. Some AFS files possibly do not have any **group** and **other** mode bits turned on, because AFS uses only the **owner** bits in combination with the ACL on the file's directory. If only the **owner** mode bits are set, NFS allows only the file's owner of the file to read or write it. Setting the **-convert** argument to the value **on** enables other users to access the file in the same manner as the owner. Setting the value **off** preserves the mode bits set on the file as stored in AFS.

- The **-uidcheck** argument controls whether tokens can be assigned to an NFS user whose local UID on the NFS client machine differs from the local UID associated with the tokens on the translator machine. By default, this is possible.

  If you turn on UID checking by setting the value **on**, then tokens can be assigned only to an NFS user whose local UID matches the local UID of the process on the translator machine that is assigning the tokens. One consequence is that there is no point in including the **-id** argument to the **knfs** command: the only acceptable value is the local UID of the command's issuer, which is the value used when the **-id** argument is omitted. Requiring matching UIDs in this way is effective only when users have the same local UID on the translator machine as on NFS client machines. In that case, it guarantees that users assign their tokens only to their own NFS sessions. For instructions, see "Authenticating on Unsupported NFS Client Machines" on page 551.

  > **Note:** Turning on UID checking also prevents users on supported NFS clients from using the **klog** command to authenticate on the NFS client directly. They must authenticated and use the **knfs** command on the translator machine instead. This is because after the **klog** command interpreter obtains the token on the NFS client, it passes it to the Cache Manager's remote executor daemon, which makes the system call that stores the token in a credential structure on the translator machine. The remote executor generally runs as the local superuser **root**, so in most cases its local UID (normally zero) does not match the local UID of the user who issued the **klog** command on the NFS client machine.
  >
  > On the other hand, although using the **knfs** command instead of the **klog** command is possibly less convenient for users, it eliminates a security exposure: the **klog** command interpreter passes the token across the network to the remote executor daemon in clear text mode.

  If you disable UID checking by assigning the value **off** , the issuer of the **knfs** command can assign tokens to a user who has a different local UID on the NFS client machine, such as the local superuser **root**. Indeed, more than one issuer of the **knfs** command can assign tokens to the same user on the NFS client machine. Each time a different user issues the **knfs** command with the same value for the

-id argument, that user's tokens overwrite the existing ones. This can result in unpredictable access for the NFS user.

- The **-submounts** argument controls whether users on the NFS client can mount AFS directories other than the top-level **/afs** directory. By default, the translator does not permit these submounts.

  Submounts can be useful in a couple of circumstances. If, for example, NFS users need to access their own AFS home directories only, then creating a submount to it eliminates the need for them to know or enter the complete path. Similarly, you can use a submount to prevent users from accessing parts of the filespace higher in the AFS hierarchy than the submount.

To configure an NFS/AFS translator machine

The following instructions configure the translator to enable users to issue AFS commands. Omit Step "6" on page 546 if you do not want to enable this functionality.

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Configure the NFS/AFS translator machine as an NFS server, if it is not already. Follow the instructions provided by your NFS supplier. The appropriate number of NFS server daemons (such as **nfsd**) depends on the number of potential NFS clients.
3. Configure the NFS/AFS translator machine as an AFS client, if it is not already. For the most predictable performance, the translator machine's local copies of the **/usr/vice/etc/CellServDB** and **/usr/vice/etc/ThisCell** files must be the same as on other client machines in the cell.
4. Modify the file that controls mounting of directories on the machine by remote NFS clients.
   - On systems that use the **/etc/exports** file, edit it to enable export of the **/afs** directory to NFS clients. You can list the names of specific NFS client machines if you want to provide access only to certain users. For a description of the file's format, see the NFS manual page for **exports(5)**.

     The following example enables any NFS client machine to mount the machine's **/afs**, **/usr**, and **/usr2** directories:

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

   - On system types that use the **share** command, edit the **/etc/dfs/dfstab** file or equivalent to include **share** instructions that enable remote mounts of the **/afs** directory. Most distributions include the binary as **/usr/sbin/share**. The following example commands enable remote mounts of the root (**/**) and **/afs** directories. To verify the correct syntax, consult the manual page for the **share** command.

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

5. Edit the machine's AFS initialization file to invoke the standard UNIX **exportfs** command after the **afsd** program runs. On some system types, the modifications you made in Step "4" on page 545 are not enough to enable exporting the AFS filespace via the **/afs** directory, because the resulting configuration changes are made before the **afsd** program runs during machine initialization. Only after the **afsd** program runs does the **/afs** directory become the mount point for the entire AFS filespace; before, it is a local directory like any other.
6. Modify the **afsd** command in the AFS initialization file to include the **-rmtsys** flag.

   For system types other than IRIX, the instructions in the *IBM AFS Quick Beginnings* for configuring the Cache Manager explain how to add the **-rmtsys** flag, for example by adding it to the line in the script that defines the value for the OPTIONS variable.

   On IRIX systems, the AFS initialization script automatically adds the **-rmtsys** flag if you have activated the **afsxnfs** configuration variable as instructed in the *IBM AFS Quick Beginnings* instructions for incorporating AFS extensions into the kernel. If the variable is not already activated, issue the following command.

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

7. **(Optional)** Depending on the number of NFS clients you expect this machine to serve, it can be beneficial to add other arguments to the **afsd** command in the machine's initialization file, such as the **-daemons** argument to set the number of background daemons. See "Administering Client Machines and the Cache Manager" on page 351 and the **afsd** reference page in the *IBM AFS Administration Reference*.
8. Reboot the machine. On many system types, the appropriate command is **shutdown**; consult your operating system administrator's guide.

   ```
   # shutdown appropriate_options
   ```

To disable or enable Translator functionality, or set optional features

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Issue the **fs exportafs** command.

   ```
   # fs exportafs nfs [-start {on | off}} ]  [-convert {on | off}]
                      [-uidcheck {on | off}]   [-submounts {on | off}]
   ```

   -start

   Disables translator functionality if the value is **off** or reenables it if the value is **on**. Omit this argument to display the current setting of all parameters set by this command.

-convert

Controls the setting of the second and third (**group** and **other**) sets of UNIX mode bits on AFS files and directories as exported to NFS clients If the value is **on**, they are set to match the **owner** mode bits. If the value is **off**, the bits are not changed. If this argument is omitted, the default value is **on**.

-uidcheck

Controls whether issuers of the **knfs** command can specify a value for its **-id** argument that does not match their AFS UID:

- If the value is **on**, the value of the **-id** argument must match the issuer's local UID.
- If the value is **off**, the issuer of the **knfs** command can use the **-id** argument to assign tokens to a user who has a different local UID on the NFS client machine, such as the local superuser **root**.

If this argument is omitted, the default value is **off**.

-submounts

Controls whether the translator services an NFS mount of any directory in the AFS filespace other than the top-level **/afs** directory. If the value is **on**, such submounts are allowed. If the value is off, only mounts of the **/afs** directory are allowed. If this argument is omitted, the default value is **off**.

Configuring NFS Client Machines

Any NFS client machine that meets the following requirements can access files in AFS via the NFS/AFS Translator. It does not need to be configured as an AFS client machine.

- It must NFS-mount a translator machine's **/afs** directory on a local directory, which by convention is also called **/afs**. The following instructions explain how to add the **mount** command to the NFS client machine's **/etc/fstab** file or equivalent.

  The directory on which an NFS client mounts the translator's machine's **/afs** directory can be called something other than **/afs**. For instance, to make it easy to switch to another translator machine if the original one becomes inaccessible, you can mount more than one translator machine's **/afs** directory. Name the mount **/afs** for the translator machine that you normally use, and use a different name the mount to each alternate translator machine.

  Mounting the AFS filespace on a directory other than **/afs** introduces another requirement, however: when issuing a command that takes an AFS pathname argument, you must specify the full pathname, starting with **/afs**, rather than a relative pathname. Suppose, for example, that a translator machine's AFS filespace is mounted at **/afs2** on an NFS client machine and you issue the following command to display the ACL on the current working directory, which is in AFS:

  ```
  % fs listacl .
  ```

The **fs** command interpreter on the NFS client must construct a full pathname before passing the request to the Cache Manager on the translator machine. The AFS filespace is mounted at **/afs2**, so the full pathname starts with that string. However, the Cache Manager on the translator cannot find a directory called **/afs2**, because its mount of the AFS filespace is called **/afs**. The command fails. To prevent the failure, provide the file's complete pathname, starting with the string **/afs**.

- It must run an appropriate number of NFS client **biod** daemons, which improve performance by handling pre-reading and delayed writing. Most NFS vendors recommend running four such daemons, and most NFS initialization scripts start them automatically. Consult your NFS documentation.

To enable users to issue AFS commands, the NFS client machine must also be a supported system type (one for which AFS binaries are available) and able to access the AFS command binaries. The *IBM AFS Release Notes* list the supported system types in each release.

In addition, the AFSSERVER and AFSCONF environment variables must be set appropriately, as discussed in "Setting the AFSSERVER and AFSCONF Environment Variables" on page 540.

To configure an NFS client machine to access AFS

Note: The following instructions enable NFS users to issue AFS commands. Omit Step "5" on page 549 and Step "6" on page 549 if you do not want to enable this functionality.

1. Become the local superuser **root** on the machine, if you are not already, by issuing the **su** command.

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

2. Configure the machine as an NFS client machine, if it is not already. Follow the instructions provided by your NFS vendor. The number of NFS client (**biod**) daemons needs to be appropriate for the expected load on this machine. The usual recommended number is four.
3. Create a directory called **/afs** on the machine, if one does not already exist, to act as the mount point for the translator machine's **/afs** directory. It is acceptable to use other names, but doing so introduces the limitation discussed in the introduction to this section.

   ```
   # mkdir /afs
   ```

4. Modify the machine's file systems registry file (**/etc/fstab** or equivalent) to include a command that mounts a translator machine's **/afs** directory. To verify the correct syntax of the **mount** command, see the operating system's **mount(5)** manual page. The following example includes options that are appropriate on many system types.

   ```
   mount -o hard,intr,timeo=300  translator_machine:/afs /afs
   ```

   where

`hard`

Indicates that the NFS client retries NFS requests until the NFS server (translator machine) responds. When using the translator, file operations possibly take longer than with NFS alone, because they must also pass through the AFS Cache Manager. With a soft mount, a delayed response from the translator machine can cause the request to abort. Many NFS versions use hard mounts by default; if your version does not, it is best to add this option.

`intr`

Enables the user to use a keyboard interrupt signal (such as **<Ctrl-c>**) to break the mount when the translator machine is inaccessible. Include this option only if the `hard` option is used, in which case the connection does not automatically break off when a translator machine goes down.

`timeo`

Sets the maximum time (in tenths of seconds) the translator can take to respond to the NFS client's request before the client considers the request timed out. With a hard mount, setting this option to a high number like 300 reduces the number of error messages like the following, which are generated when the translator does not respond immediately.

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

With a soft mount, it reduces the number of actual errors returned on timed-out requests.

`translator_machine`

Specifies the fully-qualified hostname of the translator machine whose **/afs** directory is to be mounted on the client machine's **/afs** directory.

Note: To mount the translator machine's /**afs** directory onto a directory on the NFS client other than /**afs**, substitute the alternate directory name for the second instance of `/afs` in the **mount** command.

5. **(Optional)** If appropriate, create the **/.AFSSERVER** file to set the AFSSERVER environment variable for all of the machine's users. For a discussion, see "Setting the AFSSERVER and AFSCONF Environment Variables" on page 540. Place a single line in the file, specifying the fully-qualified hostname of the translator machine that is to serve as the remote executor. To enable users to issue commands that handle tokens, it must be the machine named as translator_machine in Step "4" on page 548.
6. **(Optional)** If appropriate, create the **/.AFSCONF** file to set the AFSCONF environment variable for all of the machine's users. For a discussion, see "Setting the AFSSERVER and AFSCONF Environment Variables" on page 540. Place a single line in the file, specifying the name of the directory where the **CellServDB** and **ThisCell** files reside. If you use a central update source for these files (by convention, **/afs/**cellname**/common/etc**), name it here.

Configuring User Accounts

There are no requirements for NFS users to access AFS as unauthenticated users. To take advantage of more AFS functionality, however, they must meet the indicated requirements.

- To access AFS as authenticated users, they must of course authenticate with AFS, which requires an entry in the Protection and Authentication Databases.
- To create and store files, they need the required ACL permissions. If you are providing a home directory for storage of personal files, it is conventional to create a dedicated volume and mount it at the user's home directory location in the AFS filespace.
- To issue AFS commands, they must meet several additional requirements:
  - They must be working on an NFS client machine of a supported system type and from which the AFS command binaries are accessible.
  - Their command shell must define values for the AFSSERVER and AFSCONF environment variables, as described in "Setting the AFSSERVER and AFSCONF Environment Variables" on page 540. It is often simplest to define the variables by creating **/.AFSSERVER** and **/.AFSCONF** file in the NFS client machine's root directory, but you can also either set the variables in each user's shell initialization file (**.cshrc** or equivalent), or create files called **.AFSSERVER** and **.AFSCONF** in each user's home directory.
  - They must have an entry in the AFS Protection and Authentication Databases, so that they can authenticate if the command requires AFS privilege. Other commands instead require assuming the local **root** identity on the translator machine; for further discussion, see "The AFSSERVER Variable" on page 540.
  - Their PATH environment variable must include the pathname to the appropriate AFS binaries. If a user works on NFS client machines of different system types, include the **@sys** variable in the pathname rather than an actual system type name.

To configure a user account for issuing AFS commands

1. Create entries for the user in the Protection and Authentication Databases, or create a complete AFS account. See the instructions for account creation in "Creating and Deleting User Accounts with the uss Command Suite" on page 413 or "Administering User Accounts" on page 459.
2. Modify the user's PATH environment variable to include the pathname of AFS binaries, such as **/afs/**cellname/sysname**/usr/afsws/bin**. If the user works on NFS client machines of different system types, considering replacing the specific sysname value with the @**sys** variable. The PATH variable is commonly defined in a login or shell initialization file (such as the **.login** or **.cshrc** file).
3. **(Optional)** Set the AFSSERVER and AFSCONF environment variables if appropriate. This is required if the NFS client machines on which the user works do not have the **/.AFSSERVER** and **/.AFSCONF** files in their root directories, or if you want user-specific values to override those settings.

Either define the variables in the user's login or shell initialization file, or create the files **.AFSSERVER** and **.AFSCONF** files in the user's home directory.

For the AFSSERVER variable, specify the fully-qualified hostname of the translator machine that is to serve as the remote executor. For the AFSCONF variable, specify the name of the directory where the **CellServDB** and **ThisCell** files reside. If you use a central update source for these files (by convention, **/afs/**cellname**/common/etc**), name it here.

4. If the pathname you defined in Step "2" on page 550 includes the **@sys** variable, instruct users to check that their system name is defined correctly before they issue AFS commands. They issue the following command:

```
% fs sysname
```

Authenticating on Unsupported NFS Client Machines

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

To authenticate on an unsupported system type, establish a connection to the translator machine (using a facility such as **telnet**), and issue the **klog** command to obtain tokens for all the cells you wish to contact during the upcoming NFS session. Then issue the **knfs** command, which stores the tokens in a credential structure associated with your NFS session. The Cache Manager uses the tokens when performing AFS access requests that originate from your NFS session.

More specifically, the credential structure is identified by a process authentication group (PAG) number associated with a particular local UID on a specific NFS client machine. By default, the NFS UID recorded in the credential structure is the same as your local UID on the translator machine. You can include the **-id** argument to specify an alternate NFS UID, unless the translator machine's administrator has used the **fs exportafs** command's **-uidcheck** argument to enable UID checking. In that case, the value of the **-id** argument must match your local UID on the translator machine (so there is not point to including the **-id** argument). Enforcing matching UIDs prevents someone else from placing their tokens in your credential structure, either accidentally or on purpose. However, it means that your cell's administrators must set your local UID on the NFS client to match your local UID on the translator machine. It also makes it impossible to authenticate by issuing the **klog** command on supported NFS clients, meaning that all NFS users must use the **knfs** command. See "Controlling Optional Translator Features" on page 543.

After issuing the **knfs** command, you can begin working on the NFS client with authenticated access to AFS. When you are finished working, it is a good policy to destroy your tokens by issuing the **knfs** command on the translator machine again, this time with the **-unlog** flag. This is simpler if you have left the connection to the translator machine open, but you can always establish a new connection if you closed the original one.

If your NFS client machine is a supported system type and you wish to issue AFS commands on it, include the **-sysname** argument to the **knfs** command. The remote executor daemon on the translator machine substitutes its value for the **@sys** variable in pathnames when executing AFS commands that you issue on the NFS client machine. If your PATH environment variable uses the @**sys** variable in the pathnames for directories that house AFS binaries (as recommended), then setting this argument enables the remote executor daemon to access the AFS binaries appropriate for your NFS client machine even if its system type differs from the translator machine's.

If you do not issue the **knfs** command (or the **klog** command on the NFS client machine itself, if it is a supported system type), then you are not authenticated with AFS. For a description of unauthenticated access, see "Enabling Unauthenticated or Authenticated AFS Access" on page 539.

To authenticate using the knfs command

1. Log on to the relevant translator machine, either on the console or remotely by using a program such as **telnet**.
2. Obtain tokens for every cell you wish to access while working on the NFS client. AFS-modified login utilities acquire a token for the translator machine's local cell by default; use **klog** command to obtain tokens for other cells if desired.
3. Issue the **knfs** command to create a credential structure in the translator machine's kernel memory for storing the tokens obtained in the previous step. Include the **-id** argument to associate the structure with a UID on the NFS client that differs from your local UID on the translator machine. This is possible unless the translator machine's administrator has enabled UID checking on the translator machine; see "Controlling Optional Translator Features" on page 543. If the NFS client machine is a supported system type and you wish to issue AFS commands on it, include the **-sysname** argument to specify its system type.

   ```
   % knfs -host <host name>  [-id <user ID (decimal)>]  \
               [-sysname  <host's '@sys' value>]
   ```

   where

   -host

   Specifies the fully-qualified hostname of the NFS client machine on which you are working.

   -id

   Specifies a local UID number on the NFS client machine with which to associate the tokens, if different from your local UID on the translator machine. If this argument is omitted, the tokens are associated with an NFS UID that matches your local UID on the translator machine. In both cases, the NFS client software marks your AFS access requests with the NFS UID when it forwards them to the Cache Manager on the translator machine.

   -sysname

   Specifies the value that the local machine's remote executor daemon substitutes for the @**sys** variable in pathnames when executing AFS commands issued on the NFS client machine (which must be a supported system type).

   The following error message indicates that the translator machine's administrator has enabled UID checking and you have provided a value that differs from your local UID on the translator machine.

```
knfs: Translator in 'passwd sync' mode; remote uid must be the same as local uid
```

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

To display tokens using the knfs command

1. Log on to the relevant translator machine, either on the console or remotely by using a program such as **telnet**.
2. Issue the **knfs** command with the **-tokens** flag to display the tokens associated with either the NFS UID that matches your local UID on the translator machine or the NFS UID specified by the **-id** argument.

   ```
   % knfs -host <host name>  [-id <user ID (decimal)>] -tokens
   ```

   where

   -host

   Specifies the fully-qualified hostname of the NFS client machine on which you are working.

   -id

   Specifies the local UID on the NFS client machine for which to display tokens, if different from your local UID on the translator machine. If this argument is omitted, the tokens are for the NFS UID that matches your local UID on the translator machine.

   -tokens

   Displays the tokens.

3. Close the connection to the translator machine if desired.

To discard tokens using the knfs command

1. If you closed your connection to the translator machine after issuing the **knfs** command, reopen it.
2. Issue the **knfs** command with the **-unlog** flag.

   ```
   % knfs -host  <host name>  [-id <user ID (decimal)>]  -unlog
   ```

   where

   -host

   Specifies the fully-qualified hostname of the NFS client machine you are working on.

-id

Specifies the local UID number on the NFS client machine for which to discard the associated tokens, if different from your local UID on the translator machine. If this argument is omitted, the tokens associated with an NFS UID that matches your local UID on the translator machine are discarded.

-unlog

Discards the tokens.

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

Appendix B. Using AFS Commands

This section describes the components of AFS commands and how to make entering commands more efficient by using shortened forms. It has the following sections:

AFS Command Syntax

AFS commands that belong to suites have the following structure:

```
command_suite operation_code-switch <value>[+]  [-flag]
```

Command Names

Together, the **command_suite** and **operation_code** make up the *command name*.

The **command_suite** specifies the group of related commands to which the command belongs, and indicates which command interpreter and server process perform the command. AFS has several command suites, including **bos**, **fs**, **kas**, **package**, **pts**, **scout**, **uss** and **vos**. Some of these suites have an interactive mode in which the issuer omits the **command_suite** portion of the command name.

The **operation_code** tells the command interpreter and server process which action to perform. Most command suites include several operation codes. The *IBM AFS Administration Reference* describes each operation code in detail, and the *IBM AFS Administration Guide* describes how to use them in the context of performing administrative tasks.

Several AFS commands do not belong to a suite and so their names do not have a **command_suite** portion. Their structure is otherwise similar to the commands in the suites.

Options

The term *option* refers to both arguments and flags, which are described in the following sections.

Arguments

One or more arguments can follow the command name. Arguments specify the entities on which to act while performing the command (for example, which server machine, server process, or file). To minimize the potential for error, provide a command's arguments in the order prescribed in its syntax definition.

Each argument has two parts, which appear in the indicated order:

- The *switch* specifies the argument's type and is preceded by a hyphen (**-**). For instance, the switch **-server** usually indicates that the argument names a server machine. Switches can often be omitted, subject to the rules outlined in "Conditions for Omitting Switches" on page 557.

- The *value* names a particular entity of the type specified by the preceding switch. For example, the proper value for a **-server** switch is a server machine name like **fs3.abc.com**. Unlike switches (which have a required form), values vary depending on what the issuer wants to accomplish. Values appear surrounded by angle brackets (**< >**) in command descriptions and the online help to show that they are user-supplied variable information.

Some arguments accept multiple values, as indicated by trailing plus sign (**+**) in the command descriptions and online help. How many of a command's arguments take multiple values, and their ordering with respect to other arguments, determine when it is acceptable to omit switches. See "Conditions for Omitting Switches" on page 557.

Some commands have optional as well as required arguments; the command descriptions and online help show optional arguments in square brackets ([ ]).

Flags

Some commands have one or more flags, which specify the manner in which the command interpreter and server process perform the command, or what kind of output it produces. Flags are preceded by hyphens like switches, but they take no values. Although the command descriptions and online help generally list a command's flags after its arguments, there is no prescribed order for flags. They can appear anywhere on the command line following the operation code, except in between the parts of an argument. Flags are always optional.

An Example Command

The following example illustrates the different parts of a command that belongs to an AFS command suite.

```
% bos getdate -server fs1.abc.com -file ptserver kaserver
```

where

- **bos** is the command suite. The BOS Server executes most of the commands in this suite.
- **getdate** is the operation code. It tells the BOS Server on the specified server machine (in this case **fs1.abc.com**) to report the modification dates of binary files in the local **/usr/afs/bin** directory.
- **-server fs1.abc.com** is one argument, with **-server** as the switch and **fs1.abc.com** as the value. This argument specifies the server machine on which BOS Server is to collect and report binary dates.
- **-file ptserver kaserver** is an argument that takes multiple values. The switch is **-file** and the values are **ptserver** and **kaserver**. This argument tells the BOS Server to report the modification dates on the files **/usr/afs/bin/kaserver** and **/usr/afs/bin/ptserver**.

Rules for Entering AFS Commands

Enter each AFS command on a single line (press **<Return>** only at the end of the command). Some commands in this document appear broken across multiple lines, but that is for legibility only.

Use a space to separate each element on a command line from its neighbors. Spaces rather than commas also separate multiple values of an argument.

In many cases, the issuer of a command can reduce the amount of typing necessary by using one or both of the following methods:

- Omitting switches
- Using accepted abbreviations for operation codes, switches (if they are included at all), and some types of values

The following sections explain the conditions for omitting or shortening parts of the command line. It is always acceptable to type a command in full, with all of its switches and no abbreviations.

Conditions for Omitting Switches

It is always acceptable to type the switch part of an argument, but in many cases it is not necessary. Specifically, switches can be omitted if the following conditions are met.

- All of the command's required arguments appear in the order prescribed by the syntax statement
- No switch is provided for any argument
- There is only one value for each argument (but note the important exception discussed in the following paragraph)

Omitting switches is possible only because there is a prescribed order for each command's arguments. When the issuer does not include switches, the command interpreter relies instead on the order of arguments; it assumes that the first element after the operation code is the command's first argument, the next element is the command's second argument, and so on. The important exception is when a command's final required argument accepts multiple values. In this case, the command interpreter assumes that the issuer has correctly provided one value for each argument up through the final one, so any additional values at the end belong to the final argument.

The following list describes the rules for omitting switches from the opposite perspective: an argument's switch must be provided when any of the following conditions apply.

- The command's arguments do not appear in the prescribed order
- An optional argument is omitted but a subsequent optional argument is provided
- A switch is provided for a preceding argument
- More than one value is supplied for a preceding argument (which must take multiple values, of course); without a switch on the current argument, the command interpreter assumes that the current argument is another value for the preceding argument

An Example of Omitting Switches

Consider again the example command from "An Example Command" on page 556.

```
% bos getdate -server fs1.abc.com -file ptserver kaserver
```

This command has two required arguments: the server machine name (identified by the **-server** switch) and binary file name (identified by the **-file** switch). The second argument accepts multiple values. By complying with all three conditions, the issuer can omit the switches:

```
% bos getdate fs1.abc.com ptserver kaserver
```

Because there are no switches, the **bos** command interpreter relies on the order of arguments. It assumes that the first element following the operation code, **fs1.abc.com**, is the server machine name, and that the next argument, **ptserver**, is a binary file name. Then, because the command's second (and last) argument accepts multiple values, the command interpreter correctly interprets **kaserver** as an additional value for it.

On the other hand, the following is not acceptable because it violates the first two conditions in "Conditions for Omitting Switches" on page 557: even though there is only one value per argument, the arguments do not appear in the prescribed order, and a switch is provided for one argument but not the other.

```
% bos getdate ptserver -server fs1.abc.com
```

Rules for Using Abbreviations and Aliases

This section explains how to abbreviate operation codes, option names, server machine names, partition names, and cell names. It is not possible to abbreviate other types of values.

Abbreviating Operation Codes

It is acceptable to abbreviate an operation code to the shortest form that still distinguishes it from the other operation codes in its suite.

For example, it is acceptable to shorten **bos install** to **bos i** because there are no other operation codes in the **bos** command suite that begin with the letter **i**. In contrast, there are several **bos** operation codes that start with the letter **s**, so the abbreviations must be longer to remain unambiguous:

bos sa for **bos salvage**
bos seta for **bos setauth**
bos setc for **bos setcellname**
bos setr for **bos setrestart**
bos sh for **bos shutdown**
bos start for **bos start**
bos startu for **bos startup**
bos stat for **bos status**
bos sto for **bos stop**

In addition to abbreviations, some operation codes have an *alias*, a short form that is not derived by abbreviating the operation code to its shortest unambiguous form. For example, the alias for the **fs setacl** command is **fs sa**, whereas the shortest unambiguous abbreviation is **fs seta**.

There are two usual reasons an operation code has an alias:

- Because the command is frequently issued, it is convenient to have a form shorter than the one derived by abbreviating. The **fs setacl** command is an example.
- Because the command's name has changed, but users of previous versions of AFS know the former name. For example, **bos listhosts** has the alias **bos getcell**, its former name. It is acceptable to abbreviate aliases to their shortest unambiguous form (for example, **bos getcell** to **bos getc**).

Even if an operation code has an alias, it is still acceptable to use the shortest unambiguous form. Thus, the **fs setacl** command has three acceptable forms: **fs setacl** (the full form), **fs seta** (the shortest abbreviation), and **fs sa** (the alias).

Abbreviating Switches and Flags

It is acceptable to shorten a switch or flag to the shortest form that distinguishes it from the other switches and flags for its operation code. It is often possible to omit switches entirely, subject to the conditions listed in "Conditions for Omitting Switches" on page 557.

Abbreviating Server Machine Names

AFS server machines must have fully-qualified Internet-style host names (for example, **fs1.abc.com**), but it is not always necessary to type the full name on the command line. AFS commands accept unambiguous shortened forms, but depend on the cell's name service (such as the Domain Name Service) or a local host table to resolve a shortened name to the fully-qualified equivalent when the command is issued.

Most commands also accept the dotted decimal form of the machine's IP address as an identifier.

Abbreviating Partition Names

Partitions that house AFS volumes must have names of the form **/vicep***x* or **/vicep***xx*, where the variable final portion is one or two lowercase letters. By convention, the first server partition created on a file server machine is called **/vicepa**, the second **/vicepb**, and so on. The *IBM AFS Quick Beginnings* explains how to configure and name a file server machine's partitions in preparation for storing AFS volumes on them.

When issuing AFS commands, you can abbreviate a partition name using any of the following forms:

```
/vicepa     =     vicepa     =     a     =     0
/vicepb     =     vicepb     =     b     =     1
```

After **/vicepz** (for which the index is 25) comes

```
/vicepaa    =     vicepaa    =     aa    =     26
```

```
/vicepab   =   vicepab   =   ab   =   27
```

and so on through

```
/vicepiv   =   vicepiv   =   iv   =   255
```

Abbreviating Cell Names

A cell's full name usually matches its Internet domain name (such as **stateu.edu** for the State University or **abc.com** for ABC Corporation). Some AFS commands accept unambiguous shortened forms, usually with respect to the local **/usr/vice/etc/CellServDB file** but sometimes depending on the ability of the local name service to resolve the corresponding domain name.

Displaying Online Help for AFS Commands

To display online help for AFS commands that belong to suites, use the **help** and **apropos** operation codes. A **-help** flag is also available on every almost every AFS command.

The online help entry for a command consists of two or three lines:

- The first line names the command and briefly describes what it does
- If the command has aliases, they appear on the next line
- The final line, which begins with the string `Usage:`, lists the command's options in the prescribed order; online help entries use the same typographical symbols (brackets and so on) as this documentation.

If no operation code is specified, the **help** operation code displays the first line (short description) for every operation code in the suite:

```
% command_suite help
```

If the issuer specifies one or more operation codes, the **help** operation code displays each command's complete online entry (short description, alias if any, and syntax):

```
% command_suite help operation_code+
```

The **-help** flag displays a command's syntax but not the short description or alias:

```
% command_name -help
```

The **apropos** operation code displays the short description of any command in a suite whose operation code or short description includes the specified keyword:

```
% command_suite apropos "<help string>"
```

The following example command displays the complete online help entry for the **fs setacl** command:

```
% fs help setacl
```

```
fs setacl: set access control list
aliases: sa
Usage: fs setacl -dir <directory>+ -acl <access list entries>+
[-clear] [-negative] [-id] [-if] [-help]
```

To see only the syntax statement, use the **-help** flag:

```
% fs setacl -help
Usage: fs setacl -dir <directory>+ -acl <access list entries>+
[-clear] [-negative] [-id] [-if] [-help]
```

In the following example, a user wants to display the quota for her home volume. She knows that the relevant command belongs to the **fs** suite, but cannot remember the operation code. She uses **quota** as the keyword:

```
% fs apropos quota
listquota: list volume quota
quota: show volume quota usage
setquota: set volume quota
```

The following illustrates the error message that results if no command name or short description contains the keyword:

```
% fs apropos "list quota"
Sorry, no commands found
```

Appendix C. The afsmonitor Program Statistics

This appendix lists the statistics you can gather with the **afsmonitor** program, grouping them by category and section, and briefly describing each field, group, and section. For instructions on using the **afsmonitor** program, see "Monitoring and Auditing AFS Performance" on page 295

The Cache Manager Statistics

Cache Manager statistics fields are classified into the following sections and groups:

- PerfStats_section - Performance Statistics Section.
  - PerfStats_group - Performance Statistics Group.
  - misc_group - Miscellaneous Group.
- Server_UpDown_section - Server Up/Down Statistics Section.
  - FS_upDown_SC_group - File Server Up/Down Statistics in Same Cell Group.
  - FS_upDown_OC_group - File Server Up/Down Statistics in Other Cells Group.
  - VL_upDown_SC_group - VL Server Up/Down Statistics in Same Cell Group.
  - VL_upDown_OC_group - VL Server Up/Down Statistics in Other Cells Group.
- RPCop_section - RPC Operation Measurements Section.
  - FS_RPCopTimes_group - File Server RPC Operation Timings Group.
  - FS_RPCopErrors_group - File Server RPC Operation Errors Group.
  - FS_RPCopBytes_group - File Server RPC Transfer Timings Group.
  - CM_RPCopTimes_group - Cache Manager RPC Operation Timings Group.
- Auth_Access_section - Authentication and Replicated File Access Section.
  - Auth_Stats_group - Authentication Information for Cache Manager Group.
  - Access_Stats_group - Unreplicated File Access Group.

All Cache Manager variables categorized under these sections and groups names are listed below.

Performance Statistics Section (PerfStats_section)

Performance Statistics Group (PerfStats_group)

- dlocalAccesses: Number of data accesses to files within local cell.
- vlocalAccesses: Number of stat accesses to files within local cell.
- dremoteAccesses: Number of data accesses to files outside of local cell.

- vremoteAccesses: Number of stat accesses to files outside of local cell.
- cacheNumEntries: Number of cache entries.
- cacheBlocksTotal: Number of (1K) blocks configured for cache.
- cacheBlocksInUse: Number of cache blocks actively in use.
- cacheBlocksOrig: Number of cache blocks at bootup.
- cacheMaxDirtyChunks: Maximum number of dirty cache chunks tolerated.
- cacheCurrDirtyChunks: Current number of dirty cache chunks.
- dcacheHits: Number of data files found in local cache.
- vcacheHits: Number of stat entries found in local cache.
- dcacheMisses: Number of data files **not** found in local cache.
- vcacheMisses: Number of stat entries **not** found in local cache.
- cacheFlushes: Number of files flushed from cache.
- cacheFilesReused: Number of cache files reused.
- dcacheXAllocs: Additionally allocated dcaches.
- vcacheXAllocs: Additionally allocated vcaches.
- bufAlloced: Number of buffers allocated by AFS.
- bufHits: Number of pages found on buffer cache.
- bufMisses: Number of pages **not** found on buffer cache.
- bufFlushDirty: Number of cached dirty buffers flushed because all were busy.
- LargeBlocksActive: Number of currently used large free pool entries.
- LargeBlocksAlloced: Number of allocated large free pool entries.
- SmallBlocksActive: Number of currently used small free pool entries.
- SmallBlocksAlloced: Number of allocated used small free pool entries.
- OutStandingMemUsage: Amount of allocated memory.
- OutStandingAllocs: Outstanding osi_allocs (no osi_frees yet).
- CallBackAlloced: Number of callback structures allocated.
- CallBackFlushes: Number of callback flush operations performed.
- srvRecords: Number of servers currently on record.
- srvRecordsHWM: Server record high water mark.
- srvNumBuckets: Number of server hash chain buckets.
- srvMaxChainLength: Maximum server hash chain length.
- srvMaxChainLengthHWM: Server hash chain high water mark.
- sysName_ID: Sysname ID for host hardware.

Miscellaneous Group (misc_group)

- numPerfCalls: Number of performance calls received.
- epoch: Cache Manager epoch time.
- numCellsVisible: Number of cells we know about.
- numCellsContacted: Number of cells contacted.

Server Up/Down Statistics Section (Server_UpDown_section)

File Server Up/Down Statistics in Same Cell Group (FS_upDown_SC_group)

> **Note:** The *records* referred to in this section are the internal records kept by the **afsmonitor** program to track the processes from which data is being gathered.

- fs_sc_numTtlRecords: Number of fileserver records, active or inactive.
- fs_sc_numUpRecords: Number of (active) fileserver records currently marked up.
- fs_sc_numDownRecords: Number of (active) fileserver records currently marked down.
- fs_sc_sumOfRecordAges: Sum of fileserver record lifetimes.
- fs_sc_ageOfYoungestRecord: Age of youngest fileserver record.
- fs_sc_ageOfOldestRecord: Age of oldest fileserver record.
- fs_sc_numDowntimeIncidents: Number of (completed) downtime incidents.
- fs_sc_numRecordsNeverDown: Number of fileserver records never marked down.
- fs_sc_maxDowntimesInARecord: Maximum downtimes seen by any fileserver record.
- fs_sc_sumOfDowntimes: Sum of all (completed) downtimes, in seconds.
- fs_sc_shortestDowntime: Shortest downtime, in seconds.
- fs_sc_longestDowntime: Longest downtime, in seconds.
- fs_sc_down_0_10_min: Down time incidents: 0-10 minutes.
- fs_sc_down_10_30_min: Down time incidents: 10-30 minutes.
- fs_sc_down_half_1_hr: Down time incidents: 30-60 minutes.
- fs_sc_down_1_2_hr: Down time incidents: 1-2 hours.
- fs_sc_down_2_4_hr: Down time incidents: 2-4 hours.
- fs_sc_down_4_8_hr: Down time incidents: 4-8 hours.
- fs_sc_down_more_8_hr: Down time incidents: more than 8 hours.
- fs_sc_downDst_0: Down time incidents: 0 times.
- fs_sc_downDst_1: Down time incidents: 1 time.
- fs_sc_downDst_2_5: Down time incidents: 2-5 times.

- fs_sc_downDst_6_10: Down time incidents: 6-10 times.
- fs_sc_downDst_10_50: Down time incidents: 10-50 times.
- fs_sc_downDst_more_50: Down time incidents: more than 50 times.

File Server Up/Down Statistics in Other Cells Group (FS_upDown_OC_group)

- fs_oc_numTtlRecords: Number of fileserver records, active or inactive.
- fs_oc_numUpRecords: Number of (active) fileserver records currently marked up.
- fs_oc_numDownRecords: Number of (active) fileserver records currently marked down.
- fs_oc_sumOfRecordAges: Sum of server record lifetimes.
- fs_oc_ageOfYoungestRecord: Age of youngest fileserver record.
- fs_oc_ageOfOldestRecord: Age of oldest fileserver record.
- fs_oc_numDowntimeIncidents: Number of (completed) downtime incidents.
- fs_oc_numRecordsNeverDown: Number of fileserver records never marked down.
- fs_oc_maxDowntimesInARecord: Maximum downtimes seen by any fileserver.
- fs_oc_sumOfDowntimes: Sum of all (completed) downtimes, in seconds.
- fs_oc_shortestDowntime: Shortest downtime, in seconds.
- fs_oc_longestDowntime: Longest downtime, in seconds.
- fs_oc_down_0_10_min: Down time incidents: 0-10 minutes.
- fs_oc_down_10_30_min: Down time incidents: 10-30 minutes.
- fs_oc_down_half_1_hr: Down time incidents: 30-60 minutes.
- fs_oc_down_1_2_hr: Down time incidents: 1-2 hours.
- fs_oc_down_2_4_hr: Down time incidents: 2-4 hours.
- fs_oc_down_4_8_hr: Down time incidents: 4-8 hours.
- fs_oc_down_more_8_hr: Down time incidents: more than 8 hours.
- fs_oc_downDst_0: Down time incidents: 0 times.
- fs_oc_downDst_1: Down time incidents: 1 time.
- fs_oc_downDst_2_5: Down time incidents: 2-5 times.
- fs_oc_downDst_6_10: Down time incidents: 6-10 times.
- fs_oc_downDst_10_50: Down time incidents: 10-50 times.
- fs_oc_downDst_more_50: Down time incidents: more than 50 times.

VL Server Up/Down Statistics in Same Cell Group (VL_upDown_SC_group)

- vl_sc_numTtlRecords: Number of vlserver records, active or inactive.
- vl_sc_numUpRecords: Number of (active) vlserver records currently marked up.

- vl_sc_numDownRecords: Number of (active) vlserver records currently marked down.
- vl_sc_sumOfRecordAges: Sum of vlserver record lifetimes.
- vl_sc_ageOfYoungestRecord: Age of youngest vlserver record.
- vl_sc_ageOfOldestRecord: Age of oldest vlserver record.
- vl_sc_numDowntimeIncidents: Number of (completed) downtime incidents.
- vl_sc_numRecordsNeverDown: Number of vlserver records never marked down.
- vl_sc_maxDowntimesInARecord: Maximum downtimes seen by any vlserver record.
- vl_sc_sumOfDowntimes: Sum of all (completed) downtimes, in seconds.
- vl_sc_shortestDowntime: Shortest downtime, in seconds.
- vl_sc_longestDowntime: Longest downtime, in seconds.
- vl_sc_down_0_10_min: Down time incidents: 0-10 minutes.
- vl_sc_down_10_30_min: Down time incidents: 10-30 minutes.
- vl_sc_down_half_1_hr: Down time incidents: 30-60 minutes.
- vl_sc_down_1_2_hr: Down time incidents: 1-2 hours.
- vl_sc_down_2_4_hr: Down time incidents: 2-4 hours.
- vl_sc_down_4_8_hr: Down time incidents: 4-8 hours.
- vl_sc_down_more_8_hr: Down time incidents: more than 8 hours.
- vl_sc_downDst_0: Down time incidents: 0 times.
- vl_sc_downDst_1: Down time incidents: 1 time.
- vl_sc_downDst_2_5: Down time incidents: 2-5 times.
- vl_sc_downDst_6_10: Down time incidents: 6-10 times.
- vl_sc_downDst_10_50: Down time incidents: 10-50 times.
- vl_sc_downDst_more_50: Down time incidents: more than 50 times.

VL Server Up/Down Statistics in Other Cells Group (VL_upDown_DC_group)

- vl_oc_numTtlRecords: Number of vlserver records, active or inactive.
- vl_oc_numUpRecords: Number of (active) vlserver records currently marked up.
- vl_oc_numDownRecords: Number of (active) vlserver records currently marked down.
- vl_oc_sumOfRecordAges: Sum of vlserver record lifetimes.
- vl_oc_ageOfYoungestRecord: Age of youngest vlserver record.
- vl_oc_ageOfOldestRecord: Age of oldest vlserver record.
- vl_oc_numDowntimeIncidents: Number of (completed) downtime incidents.
- vl_oc_numRecordsNeverDown: Number of vlserver records never marked down.
- vl_oc_maxDowntimesInARecord: Maximum downtimes seen by any vlserver record.

- vl_oc_sumOfDowntimes: Sum of all (completed) downtimes, in seconds.
- vl_oc_shortestDowntime: Shortest downtime, in seconds.
- vl_oc_longestDowntime: Longest downtime, in seconds.
- vl_oc_down_0_10_min: Down time incidents: 0-10 minutes.
- vl_oc_down_10_30_min: Down time incidents: 10-30 minutes.
- vl_oc_down_half_1_hr: Down time incidents: 30-60 minutes.
- vl_oc_down_1_2_hr: Down time incidents: 1-2 hours.
- vl_oc_down_2_4_hr: Down time incidents: 2-4 hours.
- vl_oc_down_4_8_hr: Down time incidents: 4-8 hours.
- vl_oc_down_more_8_hr: Down time incidents: more than 8 hours.
- vl_oc_downDst_0: Down time incidents: 0 times.
- vl_oc_downDst_1: Down time incidents: 1 time.
- vl_oc_downDst_2_5: Down time incidents: 2-5 times.
- vl_oc_downDst_6_10: Down time incidents: 6-10 times.
- vl_oc_downDst_10_50: Down time incidents: 10-50 times.
- vl_oc_downDst_more_50: Down time incidents: more than 50 times.

RPC Operation Measurements Section (RPCop_section)

File Server RPC Operation Timings Group (FS_RPCopTimes_group)

- FetchData_ops: Number of FetchData operations executed.
- FetchData_ops_ok: Number of successful FetchData operations.
- FetchData_sum: Sum of timings for FetchData operations.
- FetchData_sqr: Sum of squares of sample timings for FetchData operations.
- FetchData_min: Minimum execution time observed for FetchData operations.
- FetchData_max: Maximum execution time observed for FetchData operations.
- FetchACL_ops: Number of FetchACL operations executed.
- FetchACL_ops_ok: Number of successful FetchACL operations.
- FetchACL_sum: Sum of timings for FetchACL operations.
- FetchACL_sqr: Sum of squares of sample timings for FetchACL operations.
- FetchACL_min: Minimum execution time observed for FetchACL operations.
- FetchACL_max: Maximum execution time observed for FetchACL operations.
- FetchStatus_ops: Number of FetchStatus operations executed.
- FetchStatus_ops_ok: Number of successful FetchStatus operations.

- FetchStatus_sum: Sum of timings for FetchStatus operations.
- FetchStatus_sqr: Sum of squares of sample timings for FetchStatus operations.
- FetchStatus_min: Minimum execution time observed for FetchStatus operations.
- FetchStatus_max: Maximum execution time observed for FetchStatus operations.
- StoreData_ops: Number of StoreData operations executed.
- StoreData_ops_ok: Number of successful StoreData operations.
- StoreData_sum: Sum of timings for StoreData operations.
- StoreData_sqr: Sum of squares of sample timings for StoreData operations.
- StoreData_min: Minimum execution time observed for StoreData operations.
- StoreData_max: Maximum execution time observed for StoreData operations.
- StoreACL_ops: Number of StoreACL operations executed.
- StoreACL_ops_ok: Number of successful StoreACL operation.
- StoreACL_sum: Sum of timings for StoreACL operations.
- StoreACL_sqr: Sum of squares of sample timings for StoreACL operations.
- StoreACL_min: Minimum execution time observed for StoreACL operations.
- StoreACL_max: Maximum execution time observed for StoreACL operations.
- StoreStatus_ops: Number of StoreStatus operations executed.
- StoreStatus_ops_ok: Number of successful StoreStatus operations.
- StoreStatus_sum: Sum of timings for StoreStatus operations.
- StoreStatus_sqr: Sum of squares of sample timings for StoreStatus operations.
- StoreStatus_min: Minimum execution time observed for StoreStatus operations.
- StoreStatus_max: Maximum execution time observed for StoreStatus operations.
- RemoveFile_ops: Number of RemoveFile operations executed.
- RemoveFile_ops_ok: Number of successful RemoveFile operations.
- RemoveFile_sum: Sum of timings for RemoveFile operations.
- RemoveFile_sqr: Sum of squares of sample timings for RemoveFile operations.
- RemoveFile_min: Minimum execution time observed for RemoveFile operations.
- RemoveFile_max: Maximum execution time observed for RemoveFile operations.
- CreateFile_ops: Number of CreateFile operations executed.
- CreateFile_ops_ok: Number of successful CreateFile operations.
- CreateFile_sum: Sum of timings for CreateFile operations.
- CreateFile_sqr: Sum of squares of sample timings for CreateFile operations.
- CreateFile_min: Minimum execution time observed for CreateFile operations.
- CreateFile_max: Maximum execution time observed for CreateFile operations.
- Rename_ops: Number of Rename operations executed.

- Rename_ops_ok: Number of successful Rename operations.
- Rename_sum: Sum of timings for Rename operations.
- Rename_sqr: Sum of squares of sample timings for Rename operations.
- Rename_min: Minimum execution time observed for Rename operations.
- Rename_max: Maximum execution time observed for Rename operations.
- Symlink_ops: Number of Symlink operations executed.
- Symlink_ops_ok: Number of successful Symlink operations.
- Symlink_sum: Sum of timings for Symlink operations.
- Symlink_sqr: Sum of squares of sample timings for Symlink operations.
- Symlink_min: Minimum execution time observed for Symlink operations.
- Symlink_max: Maximum execution time observed for Symlink operations.
- Link_ops: Number of Link operations executed.
- Link_ops_ok: Number of successful Link operations.
- Link_sum: Sum of timings for Link operations.
- Link_sqr: Sum of squares of sample timings for Link operations.
- Link_min: Minimum execution time observed for Link operations.
- Link_max: Maximum execution time observed for Link operations.
- MakeDir_ops: Number of MakeDir operations executed.
- MakeDir_ops_ok: Number of successful MakeDir operations.
- MakeDir_sum: Sum of timings for MakeDir operations.
- MakeDir_sqr: Sum of squares of sample timings for MakeDir operations.
- MakeDir_min: Minimum execution time observed for MakeDir operations.
- MakeDir_max: Maximum execution time observed for MakeDir operations.
- RemoveDir_ops: Number of RemoveDir operations executed.
- RemoveDir_ops_ok: Number of successful RemoveDir operations.
- RemoveDir_sum: Sum of timings for RemoveDir operations.
- RemoveDir_sqr: Sum of squares of sample timings for RemoveDir operations.
- RemoveDir_min: Minimum execution time observed for RemoveDir operations.
- RemoveDir_max: Maximum execution time observed for RemoveDir operations.
- SetLock_ops: Number of SetLock operations executed.
- SetLock_ops_ok: Number of successful SetLock operations.
- SetLock_sum: Sum of timings for SetLock operations.
- SetLock_sqr: Sum of squares of sample timings for SetLock operations.
- SetLock_min: Minimum execution time observed for SetLock operations.
- SetLock_max: Maximum execution time observed for SetLock operations.

- ExtendLock_ops: Number of ExtendLock operations executed.
- ExtendLock_ops_ok: Number of successful ExtendLock operations.
- ExtendLock_sum: Sum of timings for ExtendLock operations.
- ExtendLock_sqr: Sum of squares of sample timings for ExtendLock operations.
- ExtendLock_min: Minimum execution time observed for ExtendLock operations.
- ExtendLock_max: Maximum execution time observed for ExtendLock operations.
- ReleaseLock_ops: Number of ReleaseLock operations executed.
- ReleaseLock_ops_ok: Number of successful ReleaseLock operations.
- ReleaseLock_sum: Sum of timings for ReleaseLock operations.
- ReleaseLock_sqr: Sum of squares of sample timings for StoreStatus operations.
- ReleaseLock_min: Minimum execution time observed for ReleaseLock operations.
- ReleaseLock_max: Maximum execution time observed for ReleaseLock operations.
- GetStatistics_ops: Number of GetStatistics operations executed.
- GetStatistics_ops_ok: Number of successful GetStatistics operations.
- GetStatistics_sum: Sum of timings for GetStatistics operations.
- GetStatistics_sqr: Sum of squares of sample timings for GetStatistics operations.
- GetStatistics_min: Minimum execution time observed for GetStatistics operations.
- GetStatistics_max: Maximum execution time observed for GetStatistics operations.
- GiveUpCallbacks_ops: Number of GiveUpCallbacks operations executed.
- GiveUpCallbacks_ops_ok: Number of successful GiveUpCallbacks operations.
- GiveUpCallbacks_sum: Sum of timings for GiveUpCallbacks operations.
- GiveUpCallbacks_sqr: Sum of squares of sample timings for GiveUpCallbacks operations.
- GiveUpCallbacks_min: Minimum execution time observed for GiveUpCallbacks operations.
- GiveUpCallbacks_max: Maximum execution time observed for GiveUpCallbacks operations.
- GetVolumeInfo_ops: Number of GetVolumeInfo operations executed.
- GetVolumeInfo_ops_ok: Number of successful GetVolumeInfo operations.
- GetVolumeInfo_sum: Sum of timings for GetVolumeInfo operations.
- GetVolumeInfo_sqr: Sum of squares of sample timings for GetVolumeInfo operations.
- GetVolumeInfo_min: Minimum execution time observed for GetVolumeInfo operations.
- GetVolumeInfo_max: Maximum execution time observed for GetVolumeInfo operations.
- GetVolumeStatus_ops: Number of GetVolumeStatus operations executed.
- GetVolumeStatus_ops_ok: Number of successful GetVolumeStatus operations.
- GetVolumeStatus_sum: Sum of timings for GetVolumeStatus operations.
- GetVolumeStatus_sqr: Sum of squares of sample timings for GetVolumeStatus operations.
- GetVolumeStatus_min: Minimum execution time observed for GetVolumeStatus operations.

- GetVolumeStatus_max: Maximum execution time observed for GetVolumeStatus operations.
- SetVolumeStatus_ops: Number of SetVolumeStatus operations executed.
- SetVolumeStatus_ops_ok: Number of successful SetVolumeStatus operations.
- SetVolumeStatus_sum: Sum of timings for SetVolumeStatus operations.
- SetVolumeStatus_sqr: Sum of squares of sample timings for SetVolumeStatus operations.
- SetVolumeStatus_min: Minimum execution time observed for SetVolumeStatus operations.
- SetVolumeStatus_max: Maximum execution time observed for SetVolumeStatus operations.
- GetRootVolume_ops: Number of GetRootVolume operations executed.
- GetRootVolume_ops_ok: Number of successful GetRootVolume operations.
- GetRootVolume_sum: Sum of timings for GetRootVolume operations.
- GetRootVolume_sqr: Sum of squares of sample timings for GetRootVolume operations.
- GetRootVolume_min: Minimum execution time observed for GetRootVolume operations.
- GetRootVolume_max: Maximum execution time observed for GetRootVolume operations.
- CheckToken_ops: Number of CheckToken operations executed.
- CheckToken_ops_ok: Number of successful CheckToken operations.
- CheckToken_sum: Sum of timings for CheckToken operations.
- CheckToken_sqr: Sum of squares of sample timings for CheckToken operations.
- CheckToken_min: Minimum execution time observed for CheckToken operations.
- CheckToken_max: Maximum execution time observed for CheckToken operations.
- GetTime_ops: Number of GetTime operations executed.
- GetTime_ops_ok: Number of successful GetTime operations.
- GetTime_sum: Sum of timings for GetTime operations.
- GetTime_sqr: Sum of squares of sample timings for GetTime operations.
- GetTime_min: Minimum execution time observed for GetTime operations.
- GetTime_max: Maximum execution time observed for GetTime operations.
- NGetVolumeInfo_ops: Number of NGetVolumeInfo operations executed.
- NGetVolumeInfo_ops_ok: Number of successful NGetVolumeInfo operations.
- NGetVolumeInfo_sum: Sum of timings for NGetVolumeInfo operations.
- NGetVolumeInfo_sqr: Sum of squares of sample timings for NGetVolumeInfo operations.
- NGetVolumeInfo_min: Minimum execution time observed for NGetVolumeInfo operations.
- NGetVolumeInfo_max: Maximum execution time observed for NGetVolumeInfo operations.
- BulkStatus_ops: Number of BulkStatus operations executed.
- BulkStatus_ops_ok: Number of successful BulkStatus operations.
- BulkStatus_sum: Sum of timings for BulkStatus operations.
- BulkStatus_sqr: Sum of squares of sample timings for BulkStatus operations.

- BulkStatus_min: Minimum execution time observed for BulkStatus operations.
- BulkStatus_max: Maximum execution time observed for BulkStatus operations.
- XStatsVersion_ops: Number of XStatsVersion operations executed.
- XStatsVersion_ops_ok: Number of successful XStatsVersion operations.
- XStatsVersion_sum: Sum of timings for XStatsVersion operations.
- XStatsVersion_sqr: Sum of squares of sample timings for XStatsVersion operations.
- XStatsVersion_min: Minimum execution time observed for XStatsVersion operations.
- XStatsVersion_max: Maximum execution time observed for XStatsVersion operations.
- GetXStats_ops: Number of GetXStats operations executed.
- GetXStats_ops_ok: Number of successful GetXStats operations.
- GetXStats_sum: Sum of timings for GetXStats operations.
- GetXstats_sqr: Sum of squares of sample timings for GetXStats operations.
- GetXStats_min: Minimum execution time observed for GetXStats operations.
- GetXStats_max: Maximum execution time observed for GetXStats operations.

File Server RPC Operation Errors Group (FS_RPCopErrors_group)

- FetchData_srv_err: Number of server-down errors during FetchData operations.
- FetchData_net_err: Number of network errors during FetchData operations.
- FetchData_prot_err_err: Number of protection violations during FetchData operations.
- FetchData_vol_err: Number of volume related errors during FetchData operations.
- FetchData_busy_err: Number of volume busy conditions during FetchData operations.
- FetchData_other_err: Number of miscellaneous other errors during FetchData operations.
- FetchACL_srv_err: Number of server-down errors during FetchACL operations.
- FetchACL_net_err: Number of network errors during FetchACL operations.
- FetchACL_prot_err: Number of protection violations during FetchACL operations.
- FetchACL_vol_err: Number of volume related errors during FetchACL operations.
- FetchACL_busy_err: Number of volume busy conditions encountered during FetchACL operations.
- FetchACL_other_err: Number of miscellaneous other errors during FetchACL operations.
- FetchStatus_srv_err: Number of server-down errors during FetchStatus operations.
- FetchStatus_net_err: Number of network errors during FetchStatus operations.
- FetchStatus_prot_err: Number of protection violations during FetchStatus operations.
- FetchStatus_vol_err: Number of volume related errors during FetchStatus operations.
- FetchStatus_busy_err: Number of volume busy conditions encountered during FetchStatus operations.
- FetchStatus_other_err: Number of miscellaneous other errors during FetchStatus operations.

- StoreData_srv_err: Number of server-down errors during StoreData operations.
- StoreData_net_err: Number of network errors during StoreData operations.
- StoreData_prot_err: Number of protection violations during StoreData operations.
- StoreData_vol_err: Number of volume related errors during StoreData operations.
- StoreData_busy_err: Number of volume busy conditions encountered during StoreData operations.
- StoreData_other_err: Number of miscellaneous other errors during StoreData operations.
- StoreACL_srv_err: Number of server-down errors during StoreACL operations.
- StoreACL_net_err: Number of network errors during StoreACL operations.
- StoreACL_prot_err: Number of protection violations during StoreACL operations.
- StoreACL_vol_err: Number of volume related errors during StoreACL operations.
- StoreACL_busy_err: Number of volume busy conditions encountered during StoreACL operations.
- StoreACL_other_err: Number of miscellaneous other errors during StoreACL operations.
- StoreStatus_srv_err: Number of server-down errors during StoreStatus operations.
- StoreStatus_net_err: Number of network errors during StoreStatus operations.
- StoreStatus_prot_err: Number of protection violations during StoreStatus operations.
- StoreStatus_vol_err: Number of volume related errors during StoreStatus operations.
- StoreStatus_busy_err: Number of volume busy conditions encountered during StoreStatus operations.
- StoreStatus_other_err: Number of miscellaneous other errors during StoreStatus operations.
- RemoveFile_srv_err: Number of server-down errors during RemoveFile operations.
- RemoveFile_net_err: Number of network errors during RemoveFile operations.
- RemoveFile_prot_err: Number of protection violations during RemoveFile operations.
- RemoveFile_vol_err: Number of volume related errors during RemoveFile operations.
- RemoveFile_busy_err: Number of volume busy conditions encountered during RemoveFile operations.
- RemoveFile_other_err: Number of miscellaneous other errors during RemoveFile operations.
- CreateFile_srv_err: Number of server-down errors during CreateFile operations.
- CreateFile_net_err: Number of network errors during CreateFile operations.
- CreateFile_prot_err: Number of protection violations during CreateFile operations.
- CreateFile_vol_err: Number of volume related errors during CreateFile operations.
- CreateFile_busy_err: Number of volume busy conditions encountered during CreateFile operations.
- CreateFile_other_err: Number of miscellaneous other errors during CreateFile operations.
- Rename_srv_err: Number of server-down errors during Rename operations.
- Rename_net_err: Number of network errors during Rename operations.
- Rename_prot_err: Number of protection violations during Rename operations.
- Rename_vol_err: Number of volume related errors during Rename operations.

- Rename_busy_err: Number of volume busy conditions encountered during Rename operations.
- Rename_other_err: Number of miscellaneous other errors during Rename operations.
- Symlink_srv_err: Number of server-down errors during Symlink operations.
- Symlink_net_err: Number of network errors during Symlink operations.
- Symlink_prot_err: Number of protection violations during Symlink operations.
- Symlink_vol_err: Number of volume related errors during Symlink operations.
- Symlink_busy_err: Number of volume busy conditions encountered during Symlink operations.
- Symlink_other_err: Number of miscellaneous other errors during Symlink operations.
- Link_srv_err: Number of server-down errors during Link operations.
- Link_net_err: Number of network errors during Link operations.
- Link_prot_err: Number of protection violations during Link operations.
- Link_vol_err: Number of volume related errors during Link operations.
- Link_busy_err: Number of volume busy conditions encountered during Link operations.
- Link_other_err: Number of miscellaneous other errors during Link operations.
- MakeDir_srv_err: Number of server-down errors during MakeDir operations.
- MakeDir_net_err: Number of network errors during MakeDir operations.
- MakeDir_prot_err: Number of protection violations during MakeDir operations.
- MakeDir_vol_err: Number of volume related errors during MakeDir operations.
- MakeDir_busy_err: Number of volume busy conditions encountered during MakeDir operations.
- MakeDir_other_err: Number of miscellaneous other errors during MakeDir operations.
- RemoveDir_srv_err: Number of server-down errors during RemoveDir operations.
- RemoveDir_net_err: Number of network errors during RemoveDir operations.
- RemoveDir_prot_err: Number of protection violations during RemoveDir operations.
- RemoveDir_vol_err: Number of volume related errors during RemoveDir operations.
- RemoveDir_busy_err: Number of volume busy conditions encountered during RemoveDir operations.
- RemoveDir_other_err: Number of miscellaneous other errors during RemoveDir operations.
- SetLock_srv_err: Number of server-down errors during SetLock operations.
- SetLock_net_err: Number of network errors during SetLock operations.
- SetLock_prot_err: Number of protection violations during SetLock operations.
- SetLock_vol_err: Number of volume related errors during SetLock operations.
- SetLock_busy_err: Number of volume busy conditions encountered during SetLock operations.
- SetLock_other_err: Number of miscellaneous other errors during SetLock operations.
- ExtendLock_srv_err: Number of server-down errors during ExtendLock operations.
- ExtendLock_net_err: Number of network errors during ExtendLock operations.
- ExtendLock_prot_err: Number of protection violations during ExtendLock operations.

- ExtendLock_vol_err: Number of volume related errors during ExtendLock operations.
- ExtendLock_busy_err: Number of volume busy conditions encountered during ExtendLock operations.
- ExtendLock_other_err: Number of miscellaneous other errors during ExtendLock operations.
- ReleaseLock_srv_err: Number of server-down errors during ReleaseLock operations.
- ReleaseLock_net_err: Number of network errors during ReleaseLock operations.
- ReleaseLock_prot_err: Number of protection violations during ReleaseLock operations.
- ReleaseLock_vol_err: Number of volume related errors during ReleaseLock operations.
- ReleaseLock_busy_err: Number of volume busy conditions encountered during ReleaseLock operations.
- ReleaseLock_other_err: Number of miscellaneous other errors during ReleaseLock operations.
- GetStatistics_srv_err: Number of server-down errors during GetStatistics operations.
- GetStatistics_net_err: Number of network errors during GetStatistics operations.
- GetStatistics_prot_err: Number of protection violations during GetStatistics operations.
- GetStatistics_vol_err: Number of volume related errors during GetStatistics operations.
- GetStatistics_busy_err: Number of volume busy conditions encountered during GetStatistics operations.
- GetStatistics_other_err: Number of miscellaneous other errors during GetStatistics operations.
- GiveUpCallbacks_srv_err: Number of server-down errors during GiveUpCallbacks operations.
- GiveUpCallbacks_net_err: Number of network errors during GiveUpCallbacks operations.
- GiveUpCallbacks_prot_err: Number of protection violations during GiveUpCallbacks operations.
- GiveUpCallbacks_vol_err: Number of volume related errors during GiveUpCallbacks operations.
- GiveUpCallbacks_busy_err: Number of volume busy conditions encountered during GiveUpCallbacks operations.
- GiveUpCallbacks_other_err: Number of miscellaneous other errors during GiveUpCallbacks operations.
- GetVolumeInfo_srv_err: Number of server-down errors during GetVolumeInfo operations.
- GetVolumeInfo_net_err: Number of network errors during GetVolumeInfo operations.
- GetVolumeInfo_prot_err: Number of protection violations during GetVolumeInfo operations.
- GetVolumeInfo_vol_err: Number of volume related errors during GetVolumeInfo operations.
- GetVolumeInfo_busy_err: Number of volume busy conditions encountered during GetVolumeInfo operations.
- GetVolumeInfo_other_err: Number of miscellaneous other errors during GetVolumeInfo operations.
- GetVolumeStatus_srv_err: Number of server-down errors during GetVolumeStatus operations.
- GetVolumeStatus_net_err: Number of network errors during GetVolumeStatus operations.
- GetVolumeStatus_prot_err: Number of protection violations during GetVolumeStatus operations.

- GetVolumeStatus_vol_err: Number of volume related errors during GetVolumeStatus operations.
- GetVolumeStatus_busy_err: Number of volume busy conditions encountered during GetVolumeStatus operations.
- GetVolumeStatus_other_err: Number of miscellaneous other errors during GetVolumeStatus operations.
- SetVolumeStatus_srv_err : Number of server-down errors during SetVolumeStatus operations.
- SetVolumeStatus_net_err: Number of network errors during SetVolumeStatus operations.
- SetVolumeStatus_prot_err: Number of protection violations during SetVolumeStatus operations.
- SetVolumeStatus_vol_err: Number of volume related errors during SetVolumeStatus operations.
- SetVolumeStatus_busy_err: Number of volume busy conditions encountered during SetVolumeStatus operations.
- SetVolumeStatus_other_err: Number of miscellaneous other errors during SetVolumeStatus operations.
- GetRootVolume_srv_err: Number of server-down errors during GetRootVolume operations.
- GetRootVolume_net_err: Number of network errors during GetRootVolume operations.
- GetRootVolume_prot_err: Number of protection violations during GetRootVolume operations.
- GetRootVolume_vol_err: Number of volume related errors during GetRootVolume operations.
- GetRootVolume_busy_err: Number of volume busy conditions encountered during GetRootVolume operations.
- GetRootVolume_other_err: Number of miscellaneous other errors during GetRootVolume operations.
- CheckToken_srv_err: Number of server-down errors during CheckToken operations.
- CheckToken_net_err: Number of network errors during CheckToken operations.
- CheckToken_prot_err: Number of protection violations during CheckToken operations.
- CheckToken_vol_err: Number of volume related errors during CheckToken operations.
- CheckToken_busy_err: Number of volume busy conditions encountered during CheckToken operations.
- CheckToken_other_err: Number of miscellaneous other errors during CheckToken operations.
- GetTime_srv_err: Number of server-down errors during GetTime operations.
- GetTime_net_err: Number of network errors during GetTime operations.
- GetTime_prot_err: Number of protection violations during GetTime operations.
- GetTime_vol_err: Number of volume related errors during GetTime operations.
- GetTime_busy_err: Number of volume busy conditions encountered during GetTime operations.
- GetTime_other_err: Number of miscellaneous other errors during GetTime operations.
- NGetVolumeInfo_srv_err: Number of server-down errors during NGetVolumeInfo operations.
- NGetVolumeInfo_net_err: Number of network errors during NGetVolumeInfo operations.
- NGetVolumeInfo_prot_err: Number of protection violations during NGetVolumeInfo operations.

- NGetVolumeInfo_vol_err: Number of volume related errors during NGetVolumeInfo operations.
- NGetVolumeInfo_busy_err: Number of volume busy conditions encountered during NGetVolumeInfo operations.
- NGetVolumeInfo_other_err: Number of miscellaneous other errors during NGetVolumeInfo operations.
- BulkStatus_srv_err: Number of server-down errors during BulkStatus operations.
- BulkStatus_net_err: Number of network errors during BulkStatus operations.
- BulkStatus_prot_err: Number of protection violations during BulkStatus operations.
- BulkStatus_vol_err: Number of volume related errors during BulkStatus operations.
- BulkStatus_busy_err: Number of volume busy conditions encountered during BulkStatus operations.
- BulkStatus_other_err: Number of miscellaneous other errors during BulkStatus operations.
- XStatsVersion_srv_err: Number of server-down errors during XStatsVersion operations.
- XStatsVersion_net_err: Number of network errors during XStatsVersion operations.
- XStatsVersion_prot_err: Number of protection violations during XStatsVersion operations.
- XStatsVersion_vol_err: Number of volume related errors during XStatsVersion operations.
- XStatsVersion_busy_err: Number of volume busy conditions encountered during XStatsVersion operations.
- XStatsVersion_other_err: Number of miscellaneous other errors during XStatsVersion operations.
- GetXStats_srv_err: Number of server-down errors during GetXStats operations.
- GetXStats_net_err: Number of network errors during GetXStats operations.
- GetXStats_prot_err: Number of protection violations during GetXStats operations.
- GetXStats_vol_err: Number of volume related errors during GetXStats operations.
- GetXStats_busy_err: Number of volume busy conditions encountered during GetXStats operations.
- GetXStats_other_err: Number of miscellaneous other errors during GetXStats operations.

File Server RPC Transfer Timings Group (FS_RPCopBytes_group)

- FetchData_xfers: Number of FetchData operations.
- FetchData_xfers_ok: Number of successful FetchData operations.
- FetchData_xfers_sum: Sum of timing values for FetchData operations.
- FetchData_xfers_sqr: Sum of squares of sample timings for FetchData operations.
- FetchData_xfers_min: Minimum transfer time observed for FetchData operations.
- FetchData_xfers_max: Maximum transfer time observed for FetchData operations.
- FetchData_xfers_bytes_sum: Sum of bytes transferred for FetchData operations.
- FetchData_xfers_bytes_min: Minimum byte transfer observed for FetchData operations.
- FetchData_xfers_bytes_max: Maximum byte transfer observed for FetchData operations.

- FetchData_xfers_bucket0: Tally in bucket0 for FetchData operations.
- FetchData_xfers_bucket1: Tally in bucket1 for FetchData operations.
- FetchData_xfers_bucket2: Tally in bucket2 for FetchData operations.
- FetchData_xfers_bucket3: Tally in bucket3 for FetchData operations.
- FetchData_xfers_bucket4: Tally in bucket4 for FetchData operations.
- FetchData_xfers_bucket5: Tally in bucket5 for FetchData operations.
- FetchData_xfers_bucket6: Tally in bucket6 for FetchData operations.
- FetchData_xfers_bucket7: Tally in bucket7 for FetchData operations.
- FetchData_xfers_bucket8: Tally in bucket8 for FetchData operations.
- StoreData_xfers: Number of StoreData operations.
- StoreData_xfers_ok: Number of successful StoreData operations.
- StoreData_xfers_sum: Sum of timing values for StoreData operations.
- StoreData_xfers_sqr: Sum of squares of sample timings for StoreData operations.
- StoreData_xfers_min: Minimum transfer time observed for StoreData operations.
- StoreData_xfers_max: Maximum transfer time observed for StoreData operations.
- StoreData_xfers_bytes_sum: Sum of bytes transferred for StoreData operations.
- StoreData_xfers_bytes_min: Minimum byte transfer observed for StoreData operations.
- StoreData_xfers_bytes_max: Maximum byte transfer observed for StoreData operations.
- StoreData_xfers_bucket0: Tally in bucket0 for StoreData operations.
- StoreData_xfers_bucket1: Tally in bucket1 for StoreData operations.
- StoreData_xfers_bucket2: Tally in bucket2 for StoreData operations.
- StoreData_xfers_bucket3: Tally in bucket3 for StoreData operations.
- StoreData_xfers_bucket4: Tally in bucket4 for StoreData operations.
- StoreData_xfers_bucket5: Tally in bucket5 for StoreData operations.
- StoreData_xfers_bucket6: Tally in bucket6 for StoreData operations.
- StoreData_xfers_bucket7: Tally in bucket7 for StoreData operations.
- StoreData_xfers_bucket8: Tally in bucket8 for StoreData operations.

Cache Manager RPC Operation Timings Group (CM_RPCopTimes_group)

- CallBack_ops: Number of CallBack operations executed.
- CallBack_ops_ok: Number of successful CallBack operations.
- CallBack_ops_sum: Sum of timings for CallBack operations.
- CallBack_ops_min: Minimum execution time observed for CallBack operations.
- CallBack_ops_max: Maximum execution time observed for CallBack operations.

- CallBack_ops_sqr: Sum of the square of CallBack operations executed.
- InitCallBackState_ops: Number of InitCallBackState operations executed.
- InitCallBackState_ops_ok: Number of successful InitCallBackState operations.
- InitCallBackState_ops_sum: Sum of timings for InitCallBackState operations.
- InitCallBackState_ops_min: Minimum execution time observed for InitCallBackState operations.
- InitCallBackState_ops_max: Maximum execution time observed for InitCallBackState operations.
- InitCallBackState_ops_sqr: Sum of squares of timings for InitCallBackState operations.
- Probe_ops: Number of Probe operations executed.
- Probe_ops_ok: Number of successful Probe operations.
- Probe_ops_sum: Sum of timings for Probe operations.
- Probe_ops_min: Minimum execution time observed for Probe operations.
- Probe_ops_max: Maximum execution time observed for Probe operations.
- Probe_ops_sqr: Sum of squares of timings for Probe operations.
- GetLock_ops: Number of GetLock operations executed.
- GetLock_ops_ok: Number of successful GetLock operations.
- GetLock_ops_sum: Sum of timings for GetLock operations.
- GetLock_ops_min: Minimum execution time observed for GetLock operations.
- GetLock_ops_max: Maximum execution time observed for GetLock operations.
- GetLock_ops_sqr: Sum of squares of timings for GetLock operations.
- GetCE_ops: Number of GetCE operations executed.
- GetCE_ops_ok: Number of successful GetCE operations.
- GetCE_ops_sum: Sum of timings for GetCE operations.
- GetCE_ops_min: Minimum execution time observed for GetCE operations.
- GetCE_ops_max: Maximum execution time observed for GetCE operations.
- GetCE_ops_sqr: Sum of squares of timings for GetCE operations.
- XStatsVersion_CM_ops: Number of XStatsVersion operations executed.
- XStatsVersion_CM_ops_ok: Number of successful XStatsVersion operations.
- XStatsVersion_CM_ops_sum: Sum of timings for XStatsVersion operations.
- XStatsVersion_CM_ops_min: Minimum execution time observed for XStatsVersion operations.
- XStatsVersion_CM_ops_max: Maximum execution time observed for XStatsVersion operations.
- XStatsVersion_CM_ops_sqr: Sum of squares of timings for XStatsVersion operations.
- GetXStats_CM_ops: Number of GetXStats operations executed.
- GetXStats_CM_ops_ok: Number of successful GetXStats operations.
- GetXStats_CM_ops_sum: Sum of timings for GetXStats operations.
- GetXStats_CM_ops_min: Minimum execution time observed for GetXStats operations.

- GetXStats_CM_ops_max: Maximum execution time observed for GetXStats operations.
- GetXStats_CM_ops_sqr: Sum of squares of timings for GetXStats operations.

Authentication and Replicated File Access Section (Auth_Access_section)

Authentication Information for Cache Manager Group (Auth_Stats_group)

- curr_PAGs: Current number of PAGs.
- curr_Records: Current number of records in table.
- curr_AuthRecords: Current number of of authenticated records (with valid ticket).
- curr_UnauthRecords: Current number of of unauthenticated records (without any ticket at all).
- curr_MaxRecordsInPAG: Maximum records for a single PAG.
- curr_LongestChain: Length of longest current hash chain.
- PAGCreations: Number of PAG creations.
- TicketUpdates: Number of ticket additions/refreshes.
- HWM_PAGS: High water mark - number of PAGs.
- HWM_Records: High water mark - number of records.
- HWM_MaxRecordsInPAG: High water mark - maximum records for a single PAG.
- HWM_LongestChain: High water mark - longest hash chain.

Unreplicated File Access Group (Access_Stats_group)

- unreplicatedRefs: Number of references to unreplicated data.
- replicatedRefs: Number of references to replicated data.
- numReplicasAccessed: Number of replicas accessed.
- maxReplicasPerRef: Maximum number of replicas accessed per reference.
- refFirstReplicaOK: Number of references satisfied by 1st replica.

The File Server Statistics

File Server statistics are classified into the following sections and groups:

- PerfStats_section: Performance Statistics Section.
  - VnodeCache_group: Vnode Cache Group.
  - Directory_group: Directory Package Group.
  - Rx_group: Rx Group.
  - HostModule_group: Host Module Fields Group.
  - misc_group: Miscellaneous Variables Group.
- RPCop_section: RPC Operations Section.
  - RPCopTimes_group: Individual RPC Operation Timings.
  - RPCopBytes_group: Byte Information for Certain RPC Operations.

All File Server variables categorized under the above sections and groups names are listed below.

Performance Statistics Section (PerfStats_section)

Vnode Cache Group (VnodeCache_group)

- vcache_L_Entries: Number of entries in LARGE vnode cache.
- vcache_L_Allocs: Number of allocs (large).
- vcache_L_Gets: Number of gets (large).
- vcache_L_Reads: Number of reads (large).
- vcache_L_Writes: Number of writes (large).
- vcache_S_Entries: Number of entries in SMALL vnode cache.
- vcache_S_Allocs: Number of allocs (small).
- vcache_S_Gets: Number of gets (small).
- vcache_S_Reads: Number of reads (small).
- vcache_S_Writes: Number of writes (small).
- vcache_H_Entries: Number of entries in HEADER vnode cache.
- vcache_H_Gets: Number of gets (header)
- vcache_H_Replacements: Number of replacements (header)

Directory Package Group (Directory_group)

- dir_Buffers: Number of buffers in use.
- dir_Calls: Number of read calls made.
- dir_IOs: I/O operations performed.

Rx Group (Rx_group)

- rx_packetRequests: Packet allocation requests.
- rx_noPackets_RcvClass: Failed packet requests (receive class).
- rx_noPackets_SendClass: Failed packet requests (send class).
- rx_noPackets_SpecialClass: Failed packet requests (special class).
- rx_socketGreedy: Did SO_GREEDY succeed?
- rx_bogusPacketOnRead: Short packets received.
- rx_bogusHost: Host address from bogus packets.
- rx_noPacketOnRead: Read packets with no packet there.
- rx_noPacketBuffersOnRead: Packets dropped due to buffer shortage.
- rx_selects: Selects waiting on packet or timeout.
- rx_sendSelects: Selects forced upon sends.
- rx_packetsRead_RcvClass: Packets read (receive class).
- rx_packetsRead_SendClass: Packets read (send class).
- rx_packetsRead_SpecialClass: Packets read (special class).
- rx_dataPacketsRead: Unique data packets read off wire.
- rx_ackPacketsRead: ACK packets read.
- rx_dupPacketsRead: Duplicate data packets read.
- rx_spuriousPacketsRead: Inappropriate packets read.
- rx_packetsSent_RcvClass: Packets sent (receive class).
- rx_packetsSent_SendClass: Packets sent (send class).
- rx_packetsSent_SpecialClass: Packets sent (special class).
- rx_ackPacketsSent: ACK packets sent.
- rx_pingPacketsSent: Ping packets sent.
- rx_abortPacketsSent: Abort packets sent.
- rx_busyPacketsSent: Busy packets sent.
- rx_dataPacketsSent: Unique data packets sent.
- rx_dataPacketsReSent: Retransmissions sent.
- rx_dataPacketsPushed: Retransmissions pushed by NACK.
- rx_ignoreAckedPacket: Packets with ACKed flag on rxi_Start.

- rx_totalRtt_Sec and rx_totalRtt_Usec: Total round trip time (in seconds and milliseconds).
- rx_minRtt_Sec and rx_minRtt_Usec: Minimum round trip time (in seconds and milliseconds).
- rx_maxRtt_Sec and rx_maxRtt_Usec: Maximum round trip time (in seconds and milliseconds).
- rx_nRttSamples: Round trip samples.
- rx_nServerConns: Total server connections.
- rx_nClientConns: Total client connections.
- rx_nPeerStructs: Total peer structures.
- rx_nCallStructs: Total call structures.
- rx_nFreeCallStructs: Total free call structures.

Host Module Fields Group (HostModule_group)

- host_NumHostEntries: Number of host entries.
- host_HostBlocks: Blocks in use for hosts.
- host_NonDeletedHosts: Non-deleted hosts.
- host_HostsInSameNetOrSubnet: Hosts in same subnet as server.
- host_HostsInDiffSubnet: Hosts in different subnet than server.
- host_HostsInDiffNetwork: Hosts in different network than server.
- host_NumClients: Number of client entries.
- host_ClientBlocks: Blocks in use for clients.

Miscellaneous Variables Group (misc_group)

- numPerfCalls: Number of performance calls received.

RPC Operations Section (RPCop_section)

Individual RPC Operation Timings Group (RPCopTimes_group)

- epoch: Time when data collection began.
- FetchData_ops: Number of FetchData operations executed.
- FetchData_ops_ok: Number of successful FetchData operations.
- FetchData_sum: Sum of timings for FetchData operations.
- FetchData_sqr: Sum of squares of sample timings for FetchData operations.
- FetchData_min: Minimum execution time observed for FetchData operations.

- FetchData_max: Maximum execution time observed for FetchData operations.
- FetchACL_ops: Number of FetchACL operations executed.
- FetchACL_ops_ok: Number of successful FetchACL operations.
- FetchACL_sum: Sum of timings for FetchACL operations.
- FetchACL_sqr: Sum of squares of sample timings for FetchACL operations.
- FetchACL_min: Minimum execution time observed for FetchACL operations.
- FetchACL_max: Maximum execution time observed for FetchACL operations.
- FetchStatus_ops: Number of FetchStatus operations executed.
- FetchStatus_ops_ok: Number of successful FetchStatus operations.
- FetchStatus_sum: Sum of timings for FetchStatus operations.
- FetchStatus_sqr: Sum of squares of sample timings for FetchStatus operations.
- FetchStatus_min: Minimum execution time observed for FetchStatus operations.
- FetchStatus_max: Maximum execution time observed for FetchStatus operations.
- StoreData_ops: Number of StoreData operations executed.
- StoreData_ops_ok: Number of successful StoreData operations.
- StoreData_sum: Sum of timings for StoreData operations.
- StoreData_sqr: Sum of squares of sample timings for StoreData operations.
- StoreData_min: Minimum execution time observed for StoreData operations.
- StoreData_max: Maximum execution time observed for StoreData operations.
- StoreACL_ops: Number of StoreACL operations executed.
- StoreACL_ops_ok: Number of successful StoreACL operations.
- StoreACL_sum: Sum of timings for StoreACL operations.
- StoreACL_sqr: Sum of squares of sample timings for StoreACL operations.
- StoreACL_min: Minimum execution time observed for StoreACL operations.
- StoreACL_max: Maximum execution time observed for StoreACL operations.
- StoreStatus_ops: Number of StoreStatus operations executed.
- StoreStatus_ops_ok: Number of successful StoreStatus operations.
- StoreStatus_sum: Sum of timings for StoreStatus operations.
- StoreStatus_sqr: Sum of squares of sample timings for StoreStatus operations.
- StoreStatus_min: Minimum execution time observed for StoreStatus operations.
- StoreStatus_max: Maximum execution time observed for StoreStatus operations.
- RemoveFile_ops: Number of RemoveFile operations executed.
- RemoveFile_ops_ok: Number of successful RemoveFile operations.
- RemoveFile_sum: Sum of timings for RemoveFile operations.
- RemoveFile_sqr: Sum of squares of sample timings for RemoveFile operations.

- RemoveFile_min: Minimum execution time observed for RemoveFile operations.
- RemoveFile_max: Maximum execution time observed for RemoveFile operations.
- CreateFile_ops: Number of CreateFile operations executed.
- CreateFile_ops_ok: Number of successful CreateFile operations.
- CreateFile_sum: Sum of timings for CreateFile operations.
- CreateFile_sqr: Sum of squares of sample timings for CreateFile operations.
- CreateFile_min: Minimum execution time observed for CreateFile operations.
- CreateFile_max: Maximum execution time observed for CreateFile operations.
- Rename_ops: Number of Rename operations executed.
- Rename_ops_ok: Number of successful Rename operations.
- Rename_sum: Sum of timings for Rename operations.
- Rename_sqr: Sum of squares of sample timings for Rename operations.
- Rename_min: Minimum execution time observed for Rename operations.
- Rename_max: Maximum execution time observed for Rename operations.
- Symlink_ops: Number of Symlink operations executed.
- Symlink_ops_ok: Number of successful Symlink operations.
- Symlink_sum: Sum of timings for Symlink operations.
- Symlink_sqr: Sum of squares of sample timings for Symlink operations.
- Symlink_min: Minimum execution time observed for Symlink operations.
- Symlink_max: Maximum execution time observed for Symlink operations.
- Link_ops: Number of Link operations executed.
- Link_ops_ok: Number of successful Link operations.
- Link_sum: Sum of timings for Link operations.
- Link_sqr: Sum of squares of sample timings for Link operations.
- Link_min: Minimum execution time observed for Link operations.
- Link_max: Maximum execution time observed for Link operations.
- MakeDir_ops: Number of MakeDir operations executed.
- MakeDir_ops_ok: Number of successful MakeDir operations.
- MakeDir_sum: Sum of timings for MakeDir operations.
- MakeDir_sqr: Sum of squares of sample timings for MakeDir operations.
- MakeDir_min: Minimum execution time observed for MakeDir operations.
- MakeDir_max: Maximum execution time observed for MakeDir operations.
- RemoveDir_ops: Number of RemoveDir operations executed.
- RemoveDir_ops_ok: Number of successful RemoveDir operations.
- RemoveDir_sum: Sum of timings for RemoveDir operations.

- RemoveDir_sqr: Sum of squares of sample timings for RemoveDir operations.
- RemoveDir_min: Minimum execution time observed for RemoveDir operations.
- RemoveDir_max: Maximum execution time observed for RemoveDir operations.
- SetLock_ops: Number of SetLock operations executed.
- SetLock_ops_ok: Number of successful SetLock operations.
- SetLock_sum: Sum of timings for SetLock operations.
- SetLock_sqr: Sum of squares of sample timings for SetLock operations.
- SetLock_min: Minimum execution time observed for SetLock operations.
- SetLock_max: Maximum execution time observed for SetLock operations.
- ExtendLock_ops: Number of ExtendLock operations executed.
- ExtendLock_ops_ok: Number of successful ExtendLock operations.
- ExtendLock_sum: Sum of timings for ExtendLock operations.
- ExtendLock_sqr: Sum of squares of sample timings for ExtendLock operations.
- ExtendLock_min: Minimum execution time observed for ExtendLock operations.
- ExtendLock_max: Maximum execution time observed for ExtendLock operations.
- ReleaseLock_ops: Number of ReleaseLock operations executed.
- ReleaseLock_ops_ok: Number of successful ReleaseLock operations.
- ReleaseLock_sum: Sum of timings for ReleaseLock operations.
- ReleaseLock_sqr: Sum of squares of sample timings for ReleaseLock operations.
- ReleaseLock_min: Minimum execution time observed for ReleaseLock operations.
- ReleaseLock_max: Maximum execution time observed for ReleaseLock operations.
- GetStatistics_ops: Number of GetStatistics operations executed.
- GetStatistics_ops_ok: Number of successful GetStatistics operations.
- GetStatistics_sum: Sum of timings for GetStatistics operations.
- GetStatistics_sqr: Sum of squares of sample timings for GetStatistics operations.
- GetStatistics_min: Minimum execution time observed for GetStatistics operations.
- GetStatistics_max: Maximum execution time observed for GetStatistics operations.
- GiveUpCallbacks_ops: Number of GiveUpCallbacks operations executed.
- GiveUpCallbacks_ops_ok: Number of successful GiveUpCallbacks operations.
- GiveUpCallbacks_sum: Sum of timings for GiveUpCallbacks operations.
- GiveUpCallbacks_sqr: Sum of squares of sample timings for GiveUpCallbacks operations.
- GiveUpCallbacks_min: Minimum execution time observed for GiveUpCallbacks operations.
- GiveUpCallbacks_max: Maximum execution time observed for GiveUpCallbacks operations.
- GetVolumeInfo_ops: Number of GetVolumeInfo operations executed.
- GetVolumeInfo_ops_ok: Number of successful GetVolumeInfo operations.

- GetVolumeInfo_sum: Sum of timings for GetVolumeInfo operations.
- GetVolumeInfo_sqr: Sum of squares of sample timings for GetVolumeInfo operations.
- GetVolumeInfo_min: Minimum execution time observed for GetVolumeInfo operations.
- GetVolumeInfo_max: Maximum execution time observed for GetVolumeInfo operations.
- GetVolumeStatus_ops: Number of GetVolumeStatus operations executed.
- GetVolumeStatus_ops_ok: Number of successful GetVolumeStatus operations.
- GetVolumeStatus_sum: Sum of timings for GetVolumeStatus operations.
- GetVolumeStatus_sqr: Sum of squares of sample timings for GetVolumeStatus operations.
- GetVolumeStatus_min: Minimum execution time observed for GetVolumeStatus operations.
- GetVolumeStatus_max: Maximum execution time observed for GetVolumeStatus operations.
- SetVolumeStatus_ops: Number of SetVolumeStatus operations executed.
- SetVolumeStatus_ops_ok: Number of successful SetVolumeStatus operations.
- SetVolumeStatus_sum: Sum of timings for SetVolumeStatus operations.
- SetVolumeStatus_sqr: Sum of squares of sample timings for SetVolumeStatus operations.
- SetVolumeStatus_min: Minimum execution time observed for SetVolumeStatus operations.
- SetVolumeStatus_max: Maximum execution time observed for SetVolumeStatus operations.
- GetRootVolume_ops: Number of GetRootVolume operations executed.
- GetRootVolume_ops_ok: Number of successful GetRootVolume operations.
- GetRootVolume_sum: Sum of timings for GetRootVolume operations.
- GetRootVolume_sqr: Sum of squares of sample timings for GetRootVolume operations.
- GetRootVolume_min: Minimum execution time observed for GetRootVolume operations.
- GetRootVolume_max: Maximum execution time observed for GetRootVolume operations.
- CheckToken_ops: Number of CheckToken operations executed.
- CheckToken_ops_ok: Number of successful CheckToken operations.
- CheckToken_sum: Sum of timings for CheckToken operations.
- CheckToken_sqr: Sum of squares of sample timings for CheckToken operations.
- CheckToken_min: Minimum execution time observed for CheckToken operations.
- CheckToken_max: Maximum execution time observed for CheckToken operations.
- GetTime_ops: Number of GetTime operations executed.
- GetTime_ops_ok: Number of successful GetTime operations.
- GetTime_sum: Sum of timings for GetTime operations.
- GetTime_sqr: Sum of squares of sample timings for GetTime operations.
- GetTime_min: Minimum execution time observed for GetTime operations.
- GetTime_max: Maximum execution time observed for GetTime operations.
- NGetVolumeInfo_ops: Number of NGetVolumeInfo operations executed.

- NGetVolumeInfo_ops_ok: Number of successful NGetVolumeInfo operations.
- NGetVolumeInfo_sum: Sum of timings for NGetVolumeInfo operations.
- NGetVolumeInfo_sqr: Sum of squares of sample timings for NGetVolumeInfo operations.
- NGetVolumeInfo_min: Minimum execution time observed for NGetVolumeInfo operations.
- NGetVolumeInfo_max: Maximum execution time observed for NGetVolumeInfo operations.
- BulkStatus_ops: Number of BulkStatus operations executed.
- BulkStatus_ops_ok: Number of successful BulkStatus operations.
- BulkStatus_sum: Sum of timings for BulkStatus operations.
- BulkStatus_sqr: Sum of squares of sample timings for BulkStatus operations.
- BulkStatus_min: Minimum execution time observed for BulkStatus operations.
- BulkStatus_max: Maximum execution time observed for BulkStatus operations.
- XStatsVersion_ops: Number of XStatsVersion operations executed.
- XStatsVersion_ops_ok: Number of successful XStatsVersion operations.
- XStatsVersion_sum: Sum of timings for XStatsVersion operations.
- XStatsVersion_sqr: Sum of squares of sample timings for XStatsVersion operations.
- XStatsVersion_min: Minimum execution time observed for XStatsVersion operations.
- XStatsVersion_max: Maximum execution time observed for XStatsVersion operations.
- GetXStats_ops: Number of GetXStats operations executed.
- GetXStats_ops_ok: Number of successful GetXStats operations.
- GetXStats_sum: Sum of timings for GetXStats operations.
- GetXStats_sqr: Sum of squares of sample timings for GetXStats operations.
- GetXStats_min: Minimum execution time observed for GetXStats operations.
- GetXStats_max: Maximum execution time observed for GetXStats operations.

Byte Information for Certain RPC Operations Group (RPCopBytes_group)

- FetchData_xfers: Number of FetchData operations.
- FetchData_xfers_ok: Number of successful FetchData operations.
- FetchData_xfers_sum: Sum of timing values for FetchData operations.
- FetchData_xfers_sqr: Sum of squares of sample timings for FetchData operations.
- FetchData_xfers_min: Minimum transfer time observed for FetchData operations.
- FetchData_xfers_max: Maximum transfer time observed for FetchData operations.
- FetchData_xfers_bytes_sum: Sum of bytes transferred for FetchData operations.
- FetchData_xfers_bytes_min: Minimum byte transfer observed for FetchData operations.
- FetchData_xfers_bytes_max: Maximum byte transfer observed for FetchData operations.

- FetchData_xfers_bucket0: Tally in bucket0 for FetchData operations.
- FetchData_xfers_bucket1: Tally in bucket1 for FetchData operations.
- FetchData_xfers_bucket2: Tally in bucket2 for FetchData operations.
- FetchData_xfers_bucket3: Tally in bucket3 for FetchData operations.
- FetchData_xfers_bucket4: Tally in bucket4 for FetchData operations.
- FetchData_xfers_bucket5: Tally in bucket5 for FetchData operations.
- FetchData_xfers_bucket6: Tally in bucket6 for FetchData operations.
- FetchData_xfers_bucket7: Tally in bucket7 for FetchData operations.
- FetchData_xfers_bucket8: Tally in bucket8 for FetchData operations.
- StoreData_xfers: Number of StoreData operations.
- StoreData_xfers_ok: Number of successful StoreData operations.
- StoreData_xfers_sum: Sum of timing values for StoreData operations.
- StoreData_xfers_sqr: Sum of squares of sample timings for StoreData operations.
- StoreData_xfers_min: Minimum transfer time observed for StoreData operations.
- StoreData_xfers_max: Maximum transfer time observed for StoreData operations.
- StoreData_xfers_bytes_sum: Sum of bytes transferred for StoreData operations.
- StoreData_xfers_bytes_min: Minimum byte transfer observed for StoreData operations.
- StoreData_xfers_bytes_max: Maximum byte transfer observed for StoreData operations.
- StoreData_xfers_bucket0: Tally in bucket0 for StoreData operations.
- StoreData_xfers_bucket1: Tally in bucket1 for StoreData operations.
- StoreData_xfers_bucket2: Tally in bucket2 for StoreData operations.
- StoreData_xfers_bucket3: Tally in bucket3 for StoreData operations.
- StoreData_xfers_bucket4: Tally in bucket4 for StoreData operations.
- StoreData_xfers_bucket5: Tally in bucket5 for StoreData operations.
- StoreData_xfers_bucket6: Tally in bucket6 for StoreData operations.
- StoreData_xfers_bucket7: Tally in bucket7 for StoreData operations.
- StoreData_xfers_bucket8: Tally in bucket8 for StoreData operations.

Appendix D. AIX Audit Events

This Appendix provides a complete listing of the AFS events that can be audited on AIX file server machines. See Chapter "Monitoring and Auditing AFS Performance" on page 295 for instructions on auditing AFS events on AIX file server machines.

Introduction

Below is a list of the AFS events contained in the file **/afs/usr/local/audit/events.sample**. Each entry contains information on the event class, the name of the event, the parameters associated with the event, and a description of the event.

Most events have an associated error code that shows the outcome of the event (since each event is recorded after it occurs), an AFSName (the authentication identify of the requesting process), and a host ID (from which the request originated). Many events follow the RPC server entry calls defined in the *AFS Programmer's Reference Manual*.

Events are classed by functionality (this is AIX specific). Some events possibly fall into one of more of the following classes which are defined by the file **/usr/afs/local/config.sample**:

- A (afsauthent): Authentication and Identification Events
- S (afssecurity): Security Events
- P (afsprivilege): Privilege Required Events
- O (afsobjects): Object Creation and Deletion Events
- M (afsattributes): Attribute modification
- C (afsprocess): Process Control Events

Audit-Specific Events

Event	Class	Parameters	Description
AFS_Audit_WR	None	<string>	The file "/usr/afs/Audit" has been written to (AIX specific event).
AFS_Aud_On	S	ECode	Auditing is on for this server process (recorded on startup of a server).
AFS_Aud_Off	S	ECode	Auditing is off for this server process (recorded on startup of a server).
AFS_Aud_Unauth	S	ECode Event	Event triggered by an unauthorized user.

Note: The following audit-specific events indicate an error has occurred while recording the event. Most events have an AFSName associated with them and a host ID. If this information cannot be gathered out of the Rx structure, one of these events is raised.

Event	Class	Parameters	Description
AFS_Aud_NoCall	S	ECode Event	No rx call structure with this event. Cannot get security, AFS ID, or origin of call.
AFS_Aud_NoConn	S	ECode Event	No connection info associated with rx call. Cannot get security, AFS ID, or origin of call.
AFS_Aud_UnknSec	S	ECode Event	Security of call is unknown (must be authorized or unauthorized caller).
AFS_Aud_NoAFSId	S	ECode Event	No AFS ID/name associated with a secure event.
AFS_Aud_NoHost	S	ECode Event	No information about origin (machine) of caller.
AFS_Aud_EINVAL	None	Event	Error in audit event parameter (can't record the event parameter).

Volume Server Events

Event	Class	Parameters	Description
AFS_VS_Start	P C	ECode	The volume server has started.
AFS_VS_Finish	C	ECode	The volume server has finished. Finish events are rare since the server process is normally aborted.
AFS_VS_Exit	C	ECode	The volume server has exited. Exit events are rare since the server process is normally aborted.
AFS_VS_TransCr	None	ECode AFSName HostID Trans VolID	AFSVolTransCreate - Create transaction for a [volume, partition]
AFS_VS_EndTrn	None	ECode AFSName HostID Trans	AFSVolEndTrans - End a transaction.
AFS_VS_CrVol	P O	ECode AFSName HostID Trans VolID VolName Type ParentID	AFSVolCreateVolume - Create a volume (volumeId volumeName)
AFS_VS_DelVol	P O	ECode AFSName HostID Trans	AFSVolDeleteVolume - Delete a volume.
AFS_VS_NukVol	P O	ECode AFSName HostID VolID	AFSVolNukeVolume - Obliterate a volume completely (volume ID).
AFS_VS_Dump	None	ECode AFSName HostID Trans	AFSVolDump - Dump the contents of a volume.

Event	Class	Parameters	Description
AFS_VS_SigRst	P M	ECode AFSName HostID VolName	AFSVolSignalRestore - Show intention to call AFSVolRestore.
AFS_VS_Restore	P O	ECode AFSName HostID Trans	AFSVolRestore - Recreate a volume from a dump.
AFS_VS_Forward	P O	ECode AFSName HostID FromTrans Host DestTrans	AFSVolForward - Dump a volume, then restore to a given server and volume.
AFS_VS_Clone	P O	ECode AFSName HostID Trans Purge NewName NewType NewVolID	AFSVolClone - Clone (and optionally purge) a volume.
AFS_VS_ReClone	P O	ECode AFSName HostID Trans CloneVolID	AFSVolReClone - Reclone a volume.
AFS_VS_SetForw	P M	ECode AFSName HostID Trans NewHost	AFSVolSetForwarding - Set forwarding information for a moved volume.
AFS_VS_GetFlgs	None	ECode AFSName HostID Trans	AFSVolGetFlags - Get volume flags for a transaction.
AFS_VS_SetFlgs	P M	ECode AFSName HostID Trans Flags	AFSVolSetFlags - Set volume flags for a transaction.
AFS_VS_GetName	None	ECode AFSName HostID Trans	AFSVolGetName - Get the volume name associated with a transaction.
AFS_VS_GetStat	None	ECode AFSName HostID Trans	AFSVolGetStatus - Get status of a transaction/volume.
AFS_VS_SetIdTy	P M	ECode AFSName HostID Trans VolName Type ParentId CloneID BackupID	AFSVolSetIdsTypes - Set header information for a volume.
AFS_VS_SetDate	P M	ECode AFSName HostID Trans Date	AFSVolSetDate - Set creation date in a volume.
AFS_VS_ListPar	None	ECode AFSName HostID	AFSVolListPartitions - Return a list of AFS partitions on a server.
AFS_VS_ParInf	None	ECode AFSName HostID PartName	AFSVolPartitionInfo - Get partition information.
AFS_VS_ListVol	None	ECode AFSName HostID	AFSVolListVolumes - Return a list of volumes on a server.
AFS_VS_XLstVol	None	ECode AFSName HostID	AFSVolXListVolumes - Return a (detailed) list of volumes on a server.
AFS_VS_Lst1Vol	None	ECode AFSName HostID VolID	AFSVolListOneVolume - Return header information for a single volume.

Event	Class	Parameters	Description
AFS_VS_XLst1Vl	None	ECode AFSName HostID VolID	AFSVolXListOneVolume - Return (detailed) header information for a single volume.
AFS_VS_GetNVol	None	ECode AFSName HostID VolID	AFSVolGetNthVolume - Get volume header given its index.
AFS_VS_Monitor	None	ECode AFSName HostID	AFSVolMonitor - Collect server transaction state.
AFS_VS_SetInfo	P O M	ECode AFSName HostID Trans	AFSVolSetInfo - Set volume status.

Backup Server Events

Event	Class	Parameters	Description
AFS_BUDB_Start	P	ECode	The backup server has started.
AFS_BUDB_Finish	None	ECode	The backup server has finished. Finish events are rare since the server process is normally aborted.
AFS_BUDB_Exit	None	ECode	The backup server has exited. Exit events are rare since the server process is normally aborted.
AFS_BUDB_CrDmp	P O	ECode AFSName HostID dumpId	BUDB_CreateDump - Create a new dump.
AFS_BUDB_AppDmp	P	ECode AFSName HostID dumpId	BUDB_makeDumpAppended - Make the dump an appended dump.
AFS_BUDB_DelDmp	P O	ECode AFSName HostID dumpId	BUDB_DeleteDump - Delete a dump.
AFS_BUDB_FinDmp	P	ECode AFSName HostID dumpId	BUDB_FinishDump- Notify buserver that dump is finished.
AFS_BUDB_UseTpe	P M	ECode AFSName HostID dumpId	BUDB_UseTape - Create/add a tape entry to a dump.
AFS_BUDB_DelTpe	P M	ECode AFSName HostID dumpId	BUDB_DeleteTape - Remove a tape from the database.
AFS_BUDB_FinTpe	P	ECode AFSName HostID dumpId	BUDB_FinishTape - Writing to a tape is completed.
AFS_BUDB_AddVol	P M	ECode AFSName HostID volId	BUDB_AddVolume - Add a volume to a particular dump and tape.
AFS_BUDB_GetTxV	None	ECode AFSName HostID Type	BUDB_GetTextVersion - Get the version number for hosts/volume-sets/dump-hierarchy.

Event	Class	Parameters	Description
AFS_BUDB_GetTxt	P	ECode AFSName HostID Type	BUDB_GetText - Get the information about hosts/volume-sets/dump-hierarchy.
AFS_BUDB_SavTxt	M	ECode AFSName HostID Type	BUDB_SaveText - Overwrite the information about hosts/volume-sets/dump-hierarchy.
AFS_BUDB_GetLck	None	ECode AFSName HostID	BUDB_GetLock - Take a lock for reading/writing text information.
AFS_BUDB_FrALck	None	ECode AFSName HostID	BUDB_FreeLock - Free a lock.
AFS_BUDB_FreLck	None	ECode AFSName HostID	BUDB_FreeAllLocks - Free all locks.
AFS_BUDB_GetIId	None	ECode AFSName HostID	BUDB_GetInstanceId - Get lock instance id.
AFS_BUDB_DmpDB	None	ECode AFSName HostID	BUDB_DumpDB - Start dumping the database.
AFS_BUDB_RstDBH	None	ECode AFSName HostID	BUDB_RestoreDbHeader - Restore the database header.
AFS_BUDB_DBVfy	None	ECode AFSName HostID	BUDB_DbVerify - Verify the database.
AFS_BUDB_FndDmp	P	ECode AFSName HostID volName	BUDB_FindDump - Find the dump a volume belongs to.
AFS_BUDB_GetDmp	P	ECode AFSName HostID	BUDB_GetDumps - Get a list of dumps in the database.
AFS_BUDB_FnLTpe	P	ECode AFSName HostID dumpId	BUDB_FindLastTape - Find last tape, and last volume on tape of a dump.
AFS_BUDB_GetTpe	P	ECode AFSName HostID	BUDB_GetTapes - Find a list of tapes based on name or dump ID.
AFS_BUDB_GetVol	P	ECode AFSName HostID	BUDB_GetVolumes - Find a list of volumes based on dump or tape name.
AFS_BUDB_DelVDP	P M	ECode AFSName HostID dumpSetName	BUDB_DeleteVDP - Delete dumps with given name and dump path.
AFS_BUDB_FndCln	P M	ECode AFSName HostID volName	BUDB_FindClone - Find clone time of volume.
AFS_BUDB_FndLaD	P	ECode AFSName HostID volName	BUDB_FindLatestDump - Find the latest dump a volume belongs to.
AFS_BUDB_TGetVr	None	ECode AFSName HostID	BUDB_T_GetVersion - Test Get version.
AFS_BUDB_TDmpHa	P	ECode AFSName HostID file	BUDB_T_DumpHashTable - Test dump of hash table.
AFS_BUDB_TDmpDB	P	ECode AFSName HostID file	BUDB_T_DumpDatabase - Test dump of database.

Protection Server Events

Event	Class	Parameters	Description
AFS_PTS_Start	P	ECode	The protection server has started.
AFS_PTS_Finish	C	ECode	The protection server has finished. Finish events are rare since the server process is normally aborted.
AFS_PTS_Exit	C	ECode	The protection server has exited. Exit events are rare since the server process is normally aborted.
AFS_PTS_NmToId	None	ECode AFSName HostID	PR_NameToID - Perform one or more name-to-ID translations.
AFS_PTS_IdToNm	None	ECode AFSName HostID GroupId	PR_IDToName - Perform one or more ID-to-name translations.
AFS_PTS_NewEnt	None	ECode AFSName HostID GroupId Name OwnerId	PR_NewEntry - Create a PDB (Protection DataBase) entry for the given name.
AFS_PTS_INewEnt	None	ECode AFSName HostID GroupId Name OwnerId	PR_INewEntry - Create a PDB entry for the given name and ID.
AFS_PTS_LstEnt	None	ECode AFSName HostID GroupId	PR_ListEntry - Get the contents of a PDB entry based on its ID.
AFS_PTS_DmpEnt	None	ECode AFSName HostID Position	PR_DumpEntry - Get the contents of a PDB entry based on its offset.
AFS_PTS_ChgEnt	None	ECode AFSName HostID GroupId NewName NewOwnerId NewId	PR_ChangeEntry - Change an existing PDB entry’s ID, name, owner, or a combination.
AFS_PTS_SetFEnt	None	ECode AFSName HostID GroupId	PR_SetFieldsEntry - Change miscellaneous fields in an existing PDB entry.
AFS_PTS_Del	None	ECode AFSName HostID GroupId	PR_Delete - Delete an existing PDB entry.
FS_PTS_WheIsIt	None	ECode AFSName HostID GroupId Position	PR_WhereIsIt - Get the PDB byte offset of the entry for a given ID.
AFS_PTS_AdToGrp	None	ECode AFSName HostID GroupId UserId	PR_AddToGroup - Add a user to a group.
AFS_PTS_RmFmGrp	None	ECode AFSName HostID GroupId UserId	PR_RemoveFromGroup - Remove a user from a chosen group.
AFS_PTS_LstMax	None	ECode AFSName HostID	PR_ListMax - Get the largest allocated user and group ID.
AFS_PTS_SetMax	None	ECode AFSName HostID GroupId flag	PR_SetMax - Set the largest allocated user and group ID.

Event	Class	Parameters	Description
AFS_PTS_LstEle	None	ECode AFSName HostID GroupId	PR_ListElements - List all IDs associated with a user or group.
AFS_PTS_GetCPS	None	ECode AFSName HostID GroupId	PR_GetCPS - Get the CPS (Current Protection Subdomain) for the given ID.
AFS_PTS_GetCPS2	None	ECode AFSName HostID GroupId Host	PR_GetCPS2 - Get the CPS for the given id and host.
AFS_PTS_GetHCPS	None	ECode AFSName HostID Host	PR_GetHostCPS - Get the CPS for the given host.
AFS_PTS_LstOwn	None	ECode AFSName HostID GroupId	PR_ListOwned - Get all IDs owned by the given ID.
AFS_PTS_IsMemOf	None	ECode AFSName HostID UserId GroupId	PR_IsAMemberOf - Is a given user ID a member of a specified group?

Authentication Events

Event	Class	Parameters	Description
AFS_KAA_ChPswd	S	ECode AFSName HostID name instance	KAA_ChangePassword - Change password.
AFS_KAA_Auth	A S	ECode AFSName HostID name instance	KAA_Authenticate - Authenticate to the cell.
AFS_KAA_AuthO	S	ECode AFSName HostID name instance	KAA_Authenticate_old - Old style authentication.
AFS_KAT_GetTkt	A S	ECode AFSName HostID name instance	KAT_GetTicket - An attempt was made to get an AFS ticket for some principal listed in the Authentication Database.
AFS_KAT_GetTktO	S	ECode AFSName HostID name instance	KAT_GetTicket_old - An attempt was made to get an AFS ticket for some principal listed in the Authentication Database.
AFS_KAM_CrUser	S P	ECode AFSName HostID name instance	KAM_CreateUser - Create a user.
AFS_KAM_DelUser	S P	ECode AFSName HostID name instance	KAM_DeleteUser - Delete a user.
AFS_KAM_SetPswd	S	ECode AFSName HostID name instance	KAM_SetPassword - Set the password for a user.
AFS_KAM_GetPswd	S	ECode AFSName HostID name	KAM_GetPassword - Get the password of a user.

Event	Class	Parameters	Description
AFS_KAM_GetEnt	S	ECode AFSName HostID name instance	KAM_GetEntry - The RPC made by the **kas examine** command to get one entry from the Authentication Database (by index entry).
AFS_KAM_LstEnt	S	ECode AFSName HostID index	KAM_ListEntry - The RPC made to list one or more entries in the Authentication Database.
AFS_KAM_Dbg	S	ECode AFSName HostID	KAM_Debug - The RPC that produces a debugging trace for the Authentication Server.
AFS_KAM_SetFld	S P	ECode AFSName HostID name instance flags date lifetime maxAssoc	KAM_SetFields - The RPC used by the **kas setfields** command to manipulate the Authentication Database.
AFS_KAM_GetStat	S	ECode AFSName HostID	KAM_GetStatus - An RPC used to get statistics on the Authentication Server.
AFS_KAM_GRnKey	S	ECode AFSName HostID	KAM_GetRandomKey - An RPC used to generate a random encryption key.
AFS_UnlockUser	S	ECode AFSName HostID name instance	KAM_Unlock - The RPC used to initiate the **kas unlock** command.
AFS_LockStatus	None	ECode AFSName HostID name instance	KAM_LockStatus - The RPC used to determine whether a user's Authentication Database entry is locked.
AFS_UseOfPriv	P	ECode AFSName HostID name instance cell	An authorized command was issued and allowed because the user had privilege.
AFS_UnAth	S	ECode AFSName HostID name instance cell	An authorized command was issued and allowed because the system was running in noauth mode.
AFS_UDPAuth	A S	ECode name instance	An authentication attempt was made with a Kerberos client.
AFS_UDPGetTckt	A S	ECode name instance cell name instance	An attempt was made to get a Kerberos ticket.
AFS_RunNoAuth	S	ECode	Check was made and some random server is running noauth.
AFS_NoAuthDsbl	S P	ECode	Server is set to run in authenticated mode.
AFS_NoAuthEnbl	S P	ECode	Server is set to run in unauthenticated mode.

File Server and Cache Manager Interface Events

Event	Class	Parameters	Description
AFS_SRX_FchACL	None	ECode AFSName HostID (FID)	RXAFS_FetchACL - Fetch the ACL associated with the given AFS file identifier.
AFS_SRX_FchStat	None	ECode AFSName HostID (FID)	RXAFS_FetchStatus - Fetch the status information for a file system object.
AFS_SRX_StACL	M	ECode AFSName HostID (FID)	RXAFS_StoreACL - Associate an ACL with the names directory.
AFS_SRX_StStat	M	ECode AFSName HostID (FID)	RXAFS_StoreStatus - Store status information for the specified file.
AFS_SRX_RmFile	O	ECode AFSName HostID (FID) name	RXAFS_RemoveFile - Delete the given file.
AFS_SRX_CrFile	O	ECode AFSName HostID (FID) name	RXAFS_CreateFile - Create the given file.
AFS_SRX_RNmFile	O M	ECode AFSName HostID (oldFID) oldName (newFID) newName	RXAFS_Rename - Rename the specified file in the given directory.
AFS_SRX_SymLink	O	ECode AFSName HostID (FID) name	RXAFS_Symlink - Create a symbolic link.
AFS_SRX_Link	O	ECode AFSName HostID (FID) name (FID)	RXAFS_Link - Create a hard link.
AFS_SRX_MakeDir	O	ECode AFSName HostID (FID) name	RXAFS_MakeDir - Create a directory.
AFS_SRX_RmDir	O	ECode AFSName HostID (FID) name	RXAFS_RemoveDir - Remove a directory.
AFS_SRX_SetLock	None	ECode AFSName HostID (FID) type	RXAFS_SetLock - Set an advisory lock on the given file identifier.
AFS_SRX_ExtLock	None	ECode AFSName HostID (FID)	RXAFS_ExtendLock - Extend an advisory lock on a file.
AFS_SRX_RelLock	None	ECode AFSName HostID (FID)	RXAFS_ReleaseLock - Release the advisory lock on a file.
AFS_SRX_FchData	None	ECode AFSName HostID (FID)	StartRXAFS_FetchData - Begin a request to fetch file data.
AFS_SRX_StData	O	ECode AFSName HostID (FID)	StartRXAFS_StoreData - Begin a request to store file data.
AFS_SRX_BFchSta	None	ECode AFSName HostID (FID)	RXAFS_BulkStatus - Fetch status information regarding a set of file system objects.

Event	Class	Parameters	Description
AFS_SRX_SetVolS	M	ECode AFSName HostID volId volName	RXAFS_SetVolumeStatus - Set the basic status information for the named volume.
AFS_Priv	P	ECode viceId callRoutine	Checking Permission Rights of user - user has permissions.
AFS_PrivSet	P	ECode viceId callRoutine	Set the privileges of a user.

BOS Server Events

Event	Class	Parameters	Description
AFS_BOS_CreBnod	P C	ECode AFSName HostID	BOZO_CreateBnode - Create a process instance.
AFS_BOS_DelBnod	P C	ECode AFSName HostID instance	BOZO_DeleteBnode - Delete a process instance.
AFS_BOS_SetReSt	P M C	ECode AFSName HostID	BOZO_Restart - Restart a given process instance.
AFS_BOS_GetLog	P	ECode AFSName HostID	StartBOZO_GetLog - Pass the IN params when fetching a BOS Server log file.
AFS_BOS_SetStat	P M C	ECode AFSName HostID instance	BOZO_SetStatus - Set process instance status and goal.
AFS_BOS_SetTSta	P M C	ECode AFSName HostID instance	BOZO_SetTStatus - Temporarily set process instance status and goal.
AFS_BOS_StartAl	P C	ECode AFSName HostID	BOZO_StartupAll - Start all existing process instances.
AFS_BOS_ShtdAll	P C	ECode AFSName HostID	BOZO_ShutdownAll - Shut down all process instances.
AFS_BOS_ReStAll	P C	ECode AFSName HostID	BOZO_RestartAll - Shut down, then restart all process instances.
AFS_BOS_ReBos	P C	ECode AFSName HostID	BOZO_ReBozo - Shut down, then restart all process instances and the BOS Server itself.
AFS_BOS_ReBosIn	P C	ECode	BOZO_ReBozo - Same as AFS_BOS_ReBos but done internally (server restarts).
AFS_BOS_ReStart	P C	ECode AFSName HostID instance	BOZO_Restart - Restart a given process instance.
AFS_BOS_WaitAll	P C	ECode AFSName HostID	BOZO_WaitAll - Wait until all process instances have reached their goals.

Event	**Class**	**Parameters**	**Description**
AFS_BOS_AddSUsr	S P	ECode AFSName HostID	BOZO_AddSUser - Add a user to the UserList.
AFS_BOS_DelSUsr	S P	ECode AFSName HostID	BOZO_DeleteSUser - Delete a user from the UserList.
AFS_BOS_LstSUsr	None	ECode AFSName HostID	BOZO_ListSUsers - Get the name of the user in the given position in the UserList file.
AFS_BOS_LstKey	P	ECode AFSName HostID	BOZO_ListKeys - List information about the key at a given index in the key file.
AFS_BOS_LstKeyU	P	ECode AFSName HostID	BOZO_ListKeys - Same as AFS_BOS_LstKey, but unauthorized.
AFS_BOS_AddKey	S P	ECode AFSName HostID	BOZO_AddKey - Add a key to the key file.
AFS_BOS_DelKey	S P	ECode AFSName HostID	BOZO_DeleteKey - Delete the entry for an AFS key.
AFS_BOS_SetNoAu	S P	ECode AFSName HostID flag	BOZO_SetNoAuthFlag - Enable or disable authenticated call requirements.
AFS_BOS_SetCell	S P	ECode AFSName HostID name	BOZO_SetCellName - Set the name of the cell to which the BOS Server belongs.
AFS_BOS_AddHst	S P	ECode AFSName HostID name	BOZO_AddCellHost - Add an entry to the list of database server hosts.
AFS_BOS_DelHst	S P	ECode AFSName HostID name	BOZO_DeleteCellHost - Delete an entry from the list of database server hosts.
AFS_BOS_Inst	P O M	ECode AFSName HostID name	StartBOZO_Install - Pass the IN parameters when installing a server binary. EndBOZO_Install - Get the OUT parameters when installing a server binary.
AFS_BOS_UnInst	P O M	ECode AFSName HostID name	BOZO_UnInstall - Roll back from a server binary installation.
AFS_BOS_PrnLog	P O	ECode AFSName HostID	BOZO_Prune - Throw away old versions of server binaries and core file.
AFS_BOS_Exec	P C	ECode AFSName HostID cmd	BOZO_Exec - Execute a shell command at the server.
AFS_BOS_DoExec	P C	ECode exec	The **bosserver** process was restarted.

Event	Class	Parameters	Description
AFS_BOS_StpProc	P C	ECode cmd	An RPC to stop any process controlled by the BOS Server.

Volume Location Server Events

Event	Class	Parameters	Description
AFS_VL_CreEnt	P M	ECode AFSName HostID name	VL_CreateEntry - Create a VLDB entry.
AFS_VL_DelEnt	P M	ECode AFSName HostID volID	VL_DeleteEntry - Delete a VLDB entry.
AFS_VL_GetNVlID	None	ECode AFSName HostID	VL_GetNewVolumeId - Generate a new volume ID.
AFS_VL_RepEnt	P M	ECode AFSName HostID volID	VL_ReplaceEntry - Replace entire contents of VLDB entry.
AFS_VL_UpdEnt	P M	ECode AFSName HostID volID	VL_UpdateEntry - Update contents of VLDB entry.
AFS_VL_SetLck	P	ECode AFSName HostID volID	VL_SetLock - Lock VLDB entry.
AFS_VL_RelLck	P	ECode AFSName HostID volID	VL_ReleaseLock - Unlock VLDB entry.

Index