About This Guide

This section describes the purpose, organization, and conventions of this document. It contains the following sections:

Audience and Purpose xiv
Organization of the Document xiv
How to Use This Document xv
Related Documents. xvi
Document Conventions xvii
Command Syntax xviii
Rules for Typing AFS Commands xx
Rules for Using Abbreviations and Aliases xxii
Getting Online Help on AFS Commands. xxv
Ordering, Support, and Training Information xxviii

Audience and Purpose

This guide describes the concepts and procedures that an AFS® system administrator needs to know. The reader should be familiar with UNIX® and administering a UNIX system, but no previous knowledge of AFS is required.

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

Organization of the Document

This document groups AFS administrative tasks into the following sections, further dividing each section into chapters. Tabbed dividers separate the sections, which are:

- Concepts and Issues. It is critical that you be familiar with the basic concepts and guidelines in this section before attempting to use or manage an AFS system.
- Managing File Server Machines
- Managing Client Machines
- Managing Users and Groups

The individual chapters in each section contain the following:

- A chapter overview.
- A table of contents for the chapter.
- 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. You should understand these concepts prior to performing the tasks 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 step-by-step instructions are as specific as has been judged practical. Thus if two related procedures differ from one another in important details, then there are two separate sets of step-by-step instructions. Also, the same steps may be repeated in different sets of instructions, rather than referring the reader to a central place ‘for an explanation of those steps.

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 *AFS Command Reference Manual* for more detailed information about the commands.

Related Documents

You should also be familiar with the following documents included in the AFS Documentation Kit:

AFS Installation Guide

This guide provides instructions for installing an AFS system. The procedures described in this document should be performed by an experienced UNIX administrator when setting up the initial system.

Prior to using any other AFS document, the administrator should verify that AFS has been set up in the cell in accordance with this guide. Some commands only work as described if that is the case.

AFS Command Reference Manual

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

Each chapter in the *AFS Command Reference Manual* describes one suite of AFS commands, presenting them in alphabetical order. The entry for each command specifies its syntax, including the acceptable aliases and abbreviations. It then describes the command's effects, arguments, and output if any. Examples and a list of related topics/commands are provided, as are warnings where appropriate.

This manual complements the *AFS System Administrator's Guide*: while it does not include procedural or conceptual information beyond a very general introduction to the suite, it does describe commands in more detail than the *AFS System Administrator's Guide*.

AFS User's 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 logging in, changing a password, listing information, protecting files, creating groups, and troubleshooting. It provides illustrative examples for each function and describes some of the differences between the UNIX file system and AFS.

The information in the *AFS User's Guide* is a subset of the information in the *AFS System Administrator's Guide*, so the latter is probably a better source for administrators. However, administrators should be aware of the topics addressed in the *AFS User's Guide* to be better able to help users.

Document Conventions

This document uses the following conventions:

1. Command names, switches, and flags appear in **bold type** in syntax definitions, examples, and running text.
2. 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.
3. Names of directories, files, file server machines, and partitions appear in *italic type*.
4. New terms appear in ***bold italic type***.
5. Volume names and examples of screen output and file contents appear in `typewriter type`.
6. The following symbols appear in command syntax definitions, both in the manuals and online. When issuing a command, do not type any of these symbols.
   - Square brackets [ ] surround optional items.
   - Angle brackets < > surround instances (user-supplied information).
   - A plus sign + follows an argument that accepts a list.
   - A percent sign `%` represents the command shell prompt. Your prompt may differ.

Command Syntax

AFS commands have the following structure:

> **command_suite** **<***operation_code***>** **-switch** **<***instance***>[+]** **-flag**

The Command Name

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. The name indicates which command interpreter and/or server process executes 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 **command_suite** is omitted.

The *operation_code* tells the command interpreter/server process what action to perform. Most command suites include multiple operation codes. The *AFS Command Reference Manual* describes each operation code in detail, and the *AFS System Administrator's Guide* describes how to use them in the context of performing administrative tasks.

Note: There is also a miscellaneous group of commands that are not associated with any **command_suite**.

The Argument

Following the command name come one or more ***arguments***, each of which has two parts: a **-switch** and an *instance*. Arguments tell the command interpreter/server process which entities to act on while executing the command (for example, which file server machine, server process, or file). In general, the issuer should provide a command's arguments in the order prescribed in its *AFS Command Reference Manual* command description and the online help.

The two parts of an argument appear in this 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 file server machine. Switches may usually be omitted, subject to the *Rules for Omitting Switches* presented on page xx.
- The *instance* specifies a particular entity of the type specified by the preceding switch. For example, the proper instance for a **-server** switch is a file server machine name like *fs1.transarc.com*. Unlike switches (which have a required form), instances vary depending on what the issuer wants to accomplish. Instances appear surrounded by angle brackets (< >) in command descriptions and the online help to show that they are user-supplied variable information.

Some arguments accept lists of instances, rather than just one instance. A plus sign (+) follows a ***list-taking argument*** in the command descriptions and online help. The number and prescribed order of list-taking arguments in a command affects when it is legal to omit the switches. See the **Rules for Omitting Switches** section on page xx.

Some commands have optional as well as required arguments; the command descriptions and online help show optional arguments in square brackets ([ ]).

The Flag

Some commands have one or more **-flag**s, which specify the manner in which the command interpreter/server process should execute the command or what kind of output it should produce. Flags are preceded by hyphens like switches, but they take no instances. Although the command descriptions and online help generally list all flags after all instances, there is no prescribed order for flags. They may appear anywhere on the command line following the operation code, except in between a switch and its instances. Flags are always optional.

An Example Command

In the example command

% **bos getdate -server fs1.transarc.com -file ptserver kaserver -noauth**

- **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 file server machine (in this case *fs1.transarc.com*) to report the modification dates of binary files in the */usr/afs/bin* directory on its local disk.
- **-server fs1.transarc.com** is one argument, with **-server** as the switch and **fs1.transarc.com** as the instance. This argument specifies the file server machine whose BOS Server should report binary dates.
- **-file ptserver kaserver** is another argument—a list-taking one—with **-file** as the switch and **ptserver** and **kaserver** as two instances. This argument tells the BOS Server to report the modification dates on the files */usr/afs/bin/kaserver* and */usr/afs/bin/ptserver*.
- **-noauth** is a flag. It tells the BOS Server not to attempt to authenticate the issuer of the command.

Rules for Typing AFS Commands

General Rules

Type each AFS command on a single line, followed by a carriage return only at the end of the command. Some commands in this document appear broken across multiple lines, but that is for legibility reasons only (fitting them on a single line would require using very small type).

Use a space to separate each element on a command line (command suite, operation code, switch, instance, flag) from its neighbors. Spaces (not commas) also separate multiple instances of an argument.

In many cases, it is possible to reduce the typing necessary on a command line by

- Omitting the switch part of arguments.
- Using accepted abbreviations for operation codes, switches (if they are included at all), and some types of instances.

The following sections explain the conditions necessary for omitting or shortening parts of the command line. It is always legal to type a command in full, with all of its switches and no abbreviations.

Rules for Omitting Switches

It is always legal to type the switch part of an argument, but in many cases it is not necessary. Specifically, a command's switches may be omitted if the following conditions are met.

Condition 1 All of the command's required arguments appear in the order prescribed by the command description (the order is the same in the online help).

Condition 2 No switches are used on any arguments, even if they are in the correct order.

Condition 3 There is only one instance for each argument.

Note: The important exception to this condition is if the final required argument is list-taking; in this case, it is legal to provide multiple instances for the argument.

Why Omitting Switches is Possible

Omitting switches is possible only because there is a prescribed order for the arguments in each AFS command. If no switches are provided to distinguish arguments from one another, then 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, that the next element is the command's second argument, and so on. This is why arguments must appear in the prescribed order and only one instance be given per argument when there are no switches (but note the exception discussed in the next paragraph).

The Important Exception

The important exception to the one-instance-per-argument condition (Condition 3) is when a command's final required argument is a list-taking one. In this case, the command interpreter assumes that the issuer has correctly provided one instance for each argument up through the list-taking one, so that any "extra" arguments at the end are additional instances for the list-taking argument.

An Example of Omitting Switches

Consider again the example command from page xix:

> % **bos getdate -server fs1.transarc.com -file ptserver kaserver -noauth**

This command has two required arguments—file server machine name (identified by **-server**) and binary file name (identified by **-file**). The second argument (**-file**) is list-taking. By obeying all three conditions, the issuer can omit the switches:

> % **bos getdate fs1.transarc.com ptserver kaserver -noauth**

Since there are no switches (**-noauth** is a flag, not a switch), the **bos** command interpreter relies on the order of arguments. It assumes that the first element following the operation code (i.e., **fs1.transarc.com**) is the file server machine name, and that the next argument (**ptserver**) is a binary file name. Then, because the command's second (and last) argument is list-taking, the command interpreter correctly interprets **kaserver** as an additional instance for the list-taking argument.

On the other hand, the following is not acceptable because it violates conditions 1 and 2, even though it meets condition 3: even though there is only one instance 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 safe1.fs -noauth**

When Omitting Switches is Illegal

The following list summarizes the rules for omitting switches from the opposite perspective, by defining when it is necessary to provide the switch for an argument. An argument's switch must be provided when

- The command's arguments do not appear in the prescribed order.
- An argument (even an optional one) that precedes this argument in the command definition is missing.
- A switch is provided for a preceding argument.
- More than one instance is supplied for a preceding argument (which must be list-taking, of course). Without a switch on the current argument, the command interpreter thinks that the current argument is simply another item in the list for the preceding argument.

Rules for Using Abbreviations and Aliases

Shortening Operation Codes (Abbreviating and Aliasing)

It is legal to abbreviate an operation code to the shortest form that still distinguishes it from the other operation codes in its suite.

Abbreviations

For example, it is legal 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 have to 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**

Aliases

In addition to the acceptable, unambiguous 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 may be familiar with the former name. For example, **bos listhosts** has the alias **bos getcell**, its former name. Note that it is legal to abbreviate aliases of this type to their shortest unambiguous form—for instance, **bos getcell** to **bos getc**.

Even if an operation code has an alias, it is still legal to use the shortest unambiguous form. Thus, the **fs setacl** command has three acceptable forms:

fs setacl	full form
fs seta	shortest unambiguous form derived from full form
fs sa	alias

The ACCEPTABLE ABBREVIATIONS/ALIASES section of each command description in the *AFS Command Reference Manual* shows the shortest unambiguous form of the operation codes, along with alias(es) if any.

Abbreviating Switches

It is legal to shorten a switch to the shortest form that distinguishes it from the other switches for its operation code. The ACCEPTABLE ABBREVIATIONS/ALIASES section of each command description in the *AFS Command Reference Manual* shows the shortest unambiguous form for each switch.

Keep in mind that switches can often be omitted entirely, subject to the conditions listed in the **Rules for Omitting Switches** section (see page xx).

Abbreviating File Server Machine Names

All file server machines should be assigned proper Internet-style host names (for example, *fs1.transarc.com*). However, typing full names like this on every command line can become tedious. AFS commands accept unambiguous shortened forms, but which abbreviations are acceptable depends on the domain naming service available in the local cell at the time the command is issued.

It is also legal to identify a machine by its four-part decimal Internet address rather than its name.

Abbreviating Partition Names

Partitions that house AFS volumes must have names of the form */vicep**x***, where ***x*** is a letter of the alphabet. The first such partition on a file server machine is called */vicepa*, the second */vicepb* and so on. The *AFS Installation Guide* 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

and so on through

/vicepz = vicepz = z = 25

Abbreviating Cell Names

A cell's full name is its proper Internet domain name (such as *andrew.cmu.edu* for the Andrew cell at Carnegie Mellon University or *transarc.com* for Transarc® Corporation). Like file server machine names, cell names are often long and can become tedious to type. Unambiguous shortened forms are allowed in AFS commands, but which abbreviations are acceptable depends on the domain naming service available in the local cell at the time the command is issued.

Abbreviating Other Instances

It is not possible to abbreviate instances other than file server machine names, partition names, and cell names.

Getting Online Help on AFS Commands

You can access online help for AFS commands using the **help** and **apropos** operation codes available in most suites, and the **-help** flag available on every command. In addition, UNIX-style manual pages are available for the command suites.

The online help entry for each 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 will appear on the next line.
- The final line, which begins with `Usage:`, lists the command's arguments, in the prescribed order, and flags. Online help entries use the same symbols (brackets, etc.) as described in the **Document Conventions** section on page xvii.

The **help** operation code displays either the first line (short description) for every operation code in the suite, if no operation code is specified or the complete online entry (short description, alias if any, and syntax) for each specified operation code. Use it to see a complete list of a suite's operation codes or when you need to review the syntax for a certain command.

The **-help** flag displays the syntax for the command (it does not display the short description or alias).

The **apropos** operation code lists a short description of any command in a suite whose operation code or short online description includes the specified keyword. Use the **apropos** operation code when you cannot remember the exact name of the command that achieves the effect you are interested in.

Using the Help Operation Code or Flag

To list the brief descriptions of all commands in a suite, type

`%` **<***command_suite***>** **help**

without an operation code.

To display the complete help entry for one or more operation codes, type

`%` <*command_name*> **help** *operation_code*+

or

`%` <*command_name*> <*operation_code*> **-help**

Example

To see the complete online help entry for the **fs setacl** command, type the following:

```
% fs help setacl
fs setacl: set access control list
aliases: sa
Usage: fs setacl -dir <directory>+
-acl <access list entries>+ [-clear] [-negative]
```

To see only the syntax, type the following:

```
% fs setacl -help
Usage: fs setacl -dir <directory>+
-acl <access list entries>+ [-clear] [-negative]
```

Using the Apropos Operation Code

The **apropos** operation code lists the short description of any command in a suite whose name or short description contains the specified keyword string. If the keyword string includes more than one word, surround it with double quotes or other delimiters.

To display all commands that concern a certain keyword *help string*, type the following:

% *<command_name>* **apropos** *"<help string>"*

For example, suppose that a user wants to list the quota on her directory. She knows that the relevant command belongs to the **fs** suite, but cannot remember the operation code. She could use "list" as the keyword:

```
% fs apropos list
cleanacl: clean up access control list
copyacl: Copy access control list
listacl: list access control list
listcells: list configured cells
listquota: list volume quota
lsmount: list mount point
primarycell: obsolete (listed primary cell)
setacl: set access control list
whereis: list file’s location
whichcell: list file’s cell
wscell: list workstation’s cell
```

However, using "quota" would probably be more appropriate:

```
% fs apropos quota
listquota: list volume quota
quota: show volume quota usage
setquota: set volume quota
```

Note that if she used "list quota" as a keyword, she would see a message indicating that no command name or short description contained that string:

```
% fs apropos "list quota"
Sorry, no commands found
```

Using the man Command to View Manual Pages

UNIX-style manual pages are available for the AFS command suites (**bos**, **fs**, and so on). The manual page lists all of the operation codes in the suite, describes which AFS server or client process the suite provides an interface to, and details the syntax of each operation code.

To view a man page for a command suite, type the following:

% **man** <*command_suite*>

Ordering, Support, and Training Information

Ordering

To order additional software, documentation, or training, contact Transarc File System Marketing at (412) 338-4400.

Product Support

Product support is covered under basic software maintenance and includes telephone and electronic mail support for installation problems, high priority bugs, and questions. All support calls to Transarc should originate from the Site System Administrator/Site Contact.

If you have trouble with AFS or the procedures described in this document, contact your Site System Administrator/Site Contact. If necessary, the Site System Administrator/Site Contact will contact Transarc through the usual AFS Product Support channels.

Training

Transarc offers two AFS training courses: User's Training and System Administrator's Training. Training classes are usually held at customer sites; however, a limited number of Transarc-based classes are offered. A list of the topics covered in training classes is available on request.

Concepts and Issues

1. An Overview of AFS for the System Administrator

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. It has the following sections:

1.1 A Broad Overview of AFS 1-2
1.2 More Detailed Discussions of Some Basic Concepts 1-5
1.3 AFS Server Processes and the Cache Manager 1-11
1.4 The Four "Roles" for File Server Machines 1-17
1.5 AFS Security: A More Detailed Look at Mutual Authentication 1-22

1.1. A Broad Overview of AFS

This section introduces most of the key terms and concepts necessary for a basic understanding of AFS. The ideas are simplified here for easy understanding, but the corresponding subsections in section 1.2 discusses them in more detail.

AFS: A Distributed File System

AFS is a ***distributed file system*** that allows 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 may 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 "file server machines" and "server processes," not simply to "servers." Section 1.3 below discusses the server processes in detail, and section 1.4 discusses the different types of file server machines.

Cells

A ***cell*** is an administratively independent site running AFS. As a cell's system administrator, you will make many decisions about setting up and maintaining your cell in the way that best serves its users, without having to consult the administrators in other cells. For example, you will 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 Name Space

Although your AFS cell is administratively independent, you will probably want to organize your local collection of files (your ***file space*** or ***tree***) so that users from other cells can also access the information in it. AFS allows cells to combine their local file spaces into a "global file space," 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 name space*** to its users.

Volumes

AFS groups files into ***volumes***, making it possible to distribute files across many machines and yet maintain a uniform name space. 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 may 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 file space—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 allows administrators to ***replicate*** commonly-used volumes, such as those containing binaries for popular programs. Replication means putting an identical ReadOnly copy (sometimes called a ***clone***) of a volume on more than one file server machine. The crash 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 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 may build an ***access control list*** 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 rights and up to twenty different people or groups of people may appear on an access control list.

Section 1.5 describes AFS's mutual authentication procedure in somewhat more detail. Further description of access control lists appears in chapter 19 of this manual.

1.2. More Detailed Discussions of Some Basic Concepts

The previous section offered a brief overview of the many concepts that an AFS system administrator should be familiar with. The following sections examine some important concepts in more detail. Although not all concepts may be new to an experienced administrator, reading this section will help ensure a common understanding of term and concepts.

1.2.1. 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. See section 1.2.3 for more about the classes of machines used in an AFS environment.

1.2.2. Distributed File Systems

A ***file system*** is a collection of files and the facilities (programs and commands) that allow 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 file space.

1.2.3. 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.

1.2.3.1. 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 allows each server process to specialize in one area, and thus perform more efficiently. Section 1.3 describes the function of each AFS server process.

Not all AFS file server machines 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. File server machines are classified according to the server processes they run, as described in section 1.4.

1.2.3.2. 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 section 1.3.11. There are usually many more client machines in a cell than file server machines.

1.2.3.3. 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 (see section 1.2.9 for an explanation of caching). Diskless machines can access AFS if they are running NFS^{TM} and the NFS/AFS Translator (available as an add-on to AFS).

1.2.4. 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 file space 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 and/or **cd**s 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 may 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 file space visible to other AFS cells, so that users in different cells may share files if they choose. If your cell is to participate in the "global" AFS name space, it must comply with a few basic conventions governing how the local file space is configured and how the addresses of certain file server machines are advertised to the outside world.

1.2.5. The Uniform Name Space 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 file space. 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 name space***— 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 name space features, the system administrator must follow a few simple conventions in configuring client machines and file trees. See section 2.3.3 for details.

1.2.6. Volumes

A ***volume*** is a conceptual "container" for a set of related files, keeping them all together on one file server machine partition. Volumes may 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 would be impossible if a partition were 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 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. See section 1.2.7 for more about the directory/volume correspondence.

- 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; see section 1.2.8 for details. Entire sets of volumes may be backed up to tape and restored to the file system; see chapter 6. 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 case files in it are accidentally deleted or changed. See section 5.6.

- Volumes are the unit of resource management. A space quota associated with each volume limits how large it may grow. See section 5.12.

1.2.7. 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 that begins with ’#’ or ’%’ because AFS will believe that the link is actually a mount point to a regular or ReadWrite 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 section 1.3.11) 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 should appear 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.

1.2.8. Replication

Replication refers to making a copy, or ***clone***, of a "source" 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 crashes, 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 (replicated volumes should contain popular files that do not change very often). See section 2.5.7 for more details.

1.2.9. 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, "first-time 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 ReadWrite 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 readonly copy of a file (from a ReadOnly volume) to the Cache Manager, the File Server sends along a callback associated with the volume (if the file is the first file delivered from that volume). Only a single callback is required per accessed ReadOnly volume because files in a ReadOnly volume can only be changed on a volume-basis as part of volume replication. If a new version of the ReadOnly volume is released, all callbacks associated with the old version are broken.

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 will necessarily notice 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.

1.3. AFS Server Processes and the Cache Manager

Section 1.2.3 mentioned briefly that 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 communication on the network is 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 crashes; it repairs any inconsistencies caused by the crash. 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 section 2.13.2.1 for more information. 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 responsibility is to retrieve files for application programs running on the client and to maintain the files in the cache.

1.3.1. 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 flocks) on request.

1.3.2. 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 statuses 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. See chapter 3 for more details about these configuration files.

1.3.3. 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. See section 1.5.1.2 for more on tokens.
- 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 a "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. See section 1.5 to learn more about the procedures AFS uses for verification of user identity and for mutual authentication.

1.3.4. 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 rights 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 rights for the directory and all the files it contains. See chapter 19 for a definition of AFS’s seven access rights and how users can set them on access control lists.
- Allowing users to grant rights to numerous individual users—a different combination to each individual if desired. UNIX protection distinguishes only between three "users": the owner of the file, members of a single group, and everyone on the system.
- Allowing 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 rights to many users simultaneously.

- Allowing 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 ***user names*** (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 user names (to comply with Kerberos standards). The AFS server processes identify users by AFS UID, not by user name. Before they can understand whom the token represents, they need the Protection Server to translate the user name into an AFS UID. See section 1.5 for more on tokens.

1.3.5. The Volume Server

Section 1.2.6 explained the advantages gained by storing files in "volumes." 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). Creating and deleting volumes are necessary when adding and removing users from the system; volume moves are done for load balancing; and replication allows volume placement on multiple file server machines (see section 1.2.8 for more on replication).

1.3.6. 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 section 1.3.11) 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. Section 5.2.3 lists the information stored in the VLDB about each volume.

1.3.7. The Update Server

The ***Update Server*** helps guarantee that all file server machines are running the same version of a server process. System performance would be inconsistent if some machines were 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, AFS has you install new software on a single file server machine of each CPU/operating 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. See section 1.4.3 for more on binary distribution machines.

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 (section 3.3 explains the contents and purpose of these files). 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 (see section 1.4.2 for more on the system control machine). Cells running the international edition of AFS do not use a system control machine to distribute configuration files; for details, see section 1.4.2.

1.3.8. The Backup Server

The ***Backup Server*** maintains the information in the ***Backup Database***. The Backup Server and the Backup Database allow 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 should it ever become corrupted. Chapter 6 provides more detailed information on configuring and using the Backup System.

1.3.9. 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 and/or Volume Server crash. The Salvager attempts to repair disk corruption that can result from a crash.

As a system administrator, you may also invoke the Salvager as necessary, even if the File Server or Volume Server has not crashed. See section 5.11.

1.3.10. 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, Protection, Volume Location, and Backup Databases; see section 2.13. 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. Section 3.1.8 provides more technical detail about the NTPD.

1.3.11. The Cache Manager

As already mentioned in section 1.2.9, 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 several processes some of which run in the client machine's kernel and communicate with the server processes running on file server machines. Its main duty is to translate file requests (made by application programs on client machines) into remote procedure calls 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 section 1.3.6). 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.

1.4. The Four "Roles" for File Server Machines

In cells using more than one file server machine, not all of the file server machines perform exactly the same duties. A machine can assume four possible "roles," depending on the processes it is running. A machine can assume more than one role by running the processes that define the roles. **All** file server machines run the processes necessary for storing and delivering files. The four different roles are

- ***Simple file server machine***. A simple file server machine runs only the basic set of processes necessary for storing and delivering files. All file server machines run these processes, but a simple file server machine runs none of the others that define the other roles. There may be several simple file server machines in a cell. See section 1.4.1.
- ***System control machine***. A single machine acts as the system control machine for a cell running the United States edition of AFS, guaranteeing that all of the other machines always have identical versions of common configuration files. This machine usually synchronizes its clock with a time source outside the cell, is typically the binary distribution machine for its type, and is generally a database server machine. Normally, your cell's first file server machine is installed as the system control machine; it is possible (though not recommended) to change the identity of the system control machine as more machines are installed. Cells running the international edition of AFS do not use the system control machine to distribute configuration files. See section 1.4.2.
- ***Binary distribution machine***. One binary distribution machine exists for each AFS system type that the cell uses as a file server machine. The binary distribution machine ensures that other file server machines of its type have identical binaries for the AFS server processes. See section 1.4.3.
- ***Database server machine***. Each cell needs to run some number of database server machines. Each one runs the four "database server processes" that maintain replicated databases—the Authentication, Protection, Volume Location, and Backup Servers. See section 1.4.4.

The software for all AFS server processes is installed on every file server machine, even though a given machine may not actively run all server processes. This makes it easy to change the role(s) a machine takes by simply starting or stopping the role-defining process(es).

In cells running only one file server machine, the single machine is a database server machine. During installation, it is set up as the system control machine and binary distribution machine for its AFS system type, but those functions remain "dormant" until an additional file server machine is installed.

1.4.1. Simple File Server Machines

The term ***simple file server machine*** is reserved for machines that run only the AFS processes necessary for storing and delivering files, monitoring process status, and picking up new binaries and configuration files.

In general, only cells with more than three file server machines need to run any 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 section 1.4.4.

1.4.1.1. Processes on a Simple Server Machine

The processes run by a simple file server machine include

- A BOS Server process.
- An "*fs*" collection of processes combining the File Server, Volume Server, and Salvager. Being combined into a single process allows the three to coordinate their operations on the data in volumes, preventing the inconsistencies that could result from multiple simultaneous operations. See section 3.1.1 for more information on this combined 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).
- A client portion of the Update Server that picks up binary files from the binary distribution machine of its AFS system type.
- An NTP Daemon that keeps the machine's clock synchronized with the clocks on the other file server machines in the cell.

1.4.2. The System Control Machine

In cells that run the United States edition of AFS, the ***system control machine*** stores and distributes system configuration information shared by all the file server machines in the cell. Having the same configuration information makes all the file server machines behave the same way. The machines store this information in several files in their */usr/afs/etc* directories. To ensure that the files are identical, you as a system administrator should make any necessary changes only on the cell's system control machine. The United States edition of AFS is delivered to sites in the United States and Canada.

Cells running 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 cell security 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.

Section 3.3 discusses in detail the format and function of the configuration files in */usr/afs/etc*. Briefly, the files and their contents are

CellServDB	lists the cell's database server machines (see also section 1.4.4).
KeyFile	lists all of the server encryption keys that server processes may need to decrypt tickets presented by clients (see section 1.5 for information on tickets).
ThisCell	specifies the cell to which the machine belongs.
UserList	lists all of the users authorized to issue privileged commands to the BOS, Volume, Volume Location, and Backup Servers.

The *AFS Installation Guide* instructs you to assign the role of system control machine to the first machine you install in your cell, even if you will only have one machine for a while. You can later reassign the role to a different machine if you wish, although that requires changing the client portion of the Update Server running on all the other file server machines to refer to the new system control machine.

1.4.2.1. Processes on a System Control Machine

The processes that define the system control machine are

- A server portion of the Update Server, in cells using the United States edition of AFS. This distributes the common configuration files in */usr/afs/etc* to all the other file server machines. (A client portion of the Update Server runs on the other file server machines that requests distribution of the configuration files to ensure that they are the same across all file server machines.)
- An instance of *runntp* that points to a time source outside the cell, if the cell runs *runntp* at all. Instances of *runntp* on other machines point to the system control machine as their main time source.

1.4.2.2. Other Roles for the System Control Machine

The system control machine also runs the same processes as simple file server machines; section 1.4.1.1 lists them. The system control machine can also take the role of binary distribution machine for its AFS system type, and/or the role of database server machine (see sections 1.4.3 and 1.4.4). This is usually recommended for simplicity's sake. Making the system control machine a binary distribution machine does not require running an additional server portion of the Update Server (instead, the definition of the client portion on the machines to which binaries are distributed is changed).

1.4.3. Binary Distribution Machines

A ***binary distribution machine*** stores and distributes binary files for the AFS processes and command suites to all the other file server machines of its AFS system type. Each file server machine keeps its own copy of all AFS server process binaries in the directory */usr/afs/bin*. However, all file server machines must be running the same version of a process for the system to perform consistently. To ensure this, AFS binary files are installed on a single binary distribution machine. One distribution machine exists for each AFS system type used as a file server machine in the cell.

1.4.3.1. Processes on a Binary Distribution Machine

The process that defines a binary distribution machine is a server portion of the Update Server. This process distributes binaries from */usr/afs/bin* to the file server machines of the binary distribution machine's CPU/operating system type. A corresponding client portion of the Update Server runs on the other machines of that CPU/OS type and installs the binaries locally.

1.4.3.2. Other Roles for a Binary Distribution Machine

Binary distribution machines also run the same processes as simple file server machines; section 1.4.1.1 lists them. A binary distribution machine may also take the role of system control machine and/or database server machine. To learn which processes a binary distribution machine must run to assume these roles, see sections 1.4.2 and 1.4.4.

1.4.4. Database Server Machines

A ***database server machine*** is one that runs the four AFS processes that maintain databases: the Protection Server, Authentication Server, Volume Location (VL) Server, and Backup Server. The databases maintained by the processes are the Authentication Database, Protection Database, Volume Location Database (VLDB), and Backup Database, respectively. To review the functions of these server processes and their databases, see the relevant subsections of section 1.3.

1.4.4.1. Benefits of Multiple Database Server Machines

If a cell has more than one file server machine, it is recommended that more than one database server machine be run; more than three are rarely necessary. It is important to run more than one database server machine in order to replicate the databases. Replicated databases yield the same benefits as replicated 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; also, no one machine becomes overburdened by requests for information from a database.

Unlike replicated volumes, however, replicated databases do change frequently. Consistent system performance demands that all copies of the database be identical at all times, so it is not possible to distribute changes to only one or some of them. To guarantee that all copies of the database are identical, each database server process calls on AFS’s distributed database technology to handle updates. See section 2.13 to learn more about how the distributed database technology works and how to manage it.

1.4.4.2. Making Database Server Machines Known

It is critical that the AFS server processes on every file server machine in a cell recognize which machines are database server machines. The database server processes in particular must know where their peers are so that they can coordinate all of the copies of the database. The non-database server processes often need information from the databases, so they need to know where they can contact the maintaining processes. Each file server machine keeps a list of its cell’s database server machines in the file */usr/afs/etc/CellServDB*. This is one of the files that should be updated only on the cell’s system control machine in cells using the United States edition of AFS (see section 1.4.2).

1.4.4.3. Processes on a Database Server Machine

The processes that define a database server machine are

- An Authentication Server.
- A Protection Server.
- A VL Server.
- A Backup Server.

1.4.4.4. Other Roles for a Database Server Machine

Database server machines can also run the same processes as simple file server machines; section 1.4.1.1 lists the processes. One database server machine can also be the cell’s system control machine, and any database server machine can serve as the binary distribution machine of its CPU/operating system type; see sections 1.4.2 and 1.4.3.

1.5. AFS Security: 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 should not make 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 to allow unscrupulous users to monitor others' transactions, or worse, break into them and "fake" the identity of one of the participants.

The most effective precaution against eavesdropping and information theft or fakery is to have servers and clients ***mutually authenticate***, or prove their identities to one another, before trusting that the other party in the transaction is really what it claims to be. In other words, the nature of the network forces all parties on the network to be "paranoid" or "mutually suspicious," assuming that their partner 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 to one another by demonstrating that they possess a ***shared secret***. A shared secret is a piece of information known only to the parties who are mutually authenticating (they may learn it in the first place from a trusted third party or some other source). In establishing a connection, one of the processes "presents" the shared secret for the other party to identify, 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 in a transaction use their shared key to encrypt the "packets" of information they send and to decrypt the ones they receive. Keys actually serve two related purposes in mutual authentication. First, they protect messages as they cross the network by preventing anyone who does not know the key from eavesdropping on the transaction, and using the knowledge to masquerade as client or server. Second, knowledge of the key serves as the main proof of identity (the shared secret). The client and server's ability to encrypt and decrypt messages with it indicates that they share the same key. If they were using two different keys, messages would remain scrambled and unintelligible after decryption.

1.5.1. Simple and Complex Mutual Authentication

The paragraphs above describe the most important concepts underlying mutual authentication. The remainder of this section explains how AFS employs the basic concepts to protect data, and includes some descriptions of implementation. You should feel free to skip this section if you are not interested in the mutual authentication process.

Mutual authentication procedures are considered "simple" or "complex" depending on how many encryption keys are used (and how many parties are involved before mutual authentication is finished).

1.5.1.1. 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 should be 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 could 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. If the client had used a different key than the server, the challenge message would have remained scrambled and unintelligible even after decryption.

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

1.5.1.2. Complex Mutual Authentication

"Complex" mutual authentication involves three encryption keys and three parties. All secure AFS transactions (except the first part of login) 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 server encryption key is a shared secret between the ticket-granter and the server; the client does not know the secret.

The ticket-granter sends several other items to the client with the ticket. These items inform the client about important things about the ticket, and are necessary because the client itself cannot read the ticket. Together with the ticket, all of these items make up 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 may be several transactions during a single connection).
- An indication of which server the ticket is used for (which server encryption key the ticket itself is encrypted with).
- An indication of how long the ticket is valid. The default lifetime of AFS server tickets is 100 hours. If the client wants to contact the server again after the ticket expires, it will have to 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, so no one but a server who knows the key can read the ticket. This protects the ticket as it crosses the network.
- 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 decrypts the ticket (using the server encryption key) 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 had used a different session key than the one the server found inside the ticket, then the request message would have remained unintelligible even after decryption. The two copies of the session key (the one inside the ticket and the one the client used) could 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.

2. Issues in Cell Set-Up and Administration

This chapter summarizes many of the issues you should be aware of when 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 presented in chapter 1.

You should read this chapter before installing your cell's first file server machine or performing any other administrative task. The *AFS Installation Guide* refers you to this chapter in some cases.

This chapter has the following sections:

2.1 Differences between AFS and UNIX: A Summary 2-2
2.2 Choosing a Cell Name 2-6
2.3 Participating in the AFS Global Name Space 2-9
2.4 Setting Up Your File Tree 2-13
2.5 Setting Up Your Volumes 2-16
2.6 Supported System Types 2-22
2.7 Setting Up File Server Machines 2-23
2.8 Setting Up Client Machines 2-27
2.9 Setting Up AFS User Accounts 2-30
2.10 Using AFS Protection Groups 2-36
2.11 Login and Authentication in AFS 2-38
2.12 Security and Authorization in AFS 2-47
2.13 AFS's Distributed Database Technology 2-51
2.14 Backing Up the System 2-57
2.15 Using Standard Remote Services in the AFS Environment 2-58
2.16 Accessing AFS through NFS 2-59

2.1. Differences between AFS and UNIX: A Summary

AFS is intended to behave like a standard UNIX file system in most respects, while also making file sharing easy within and between cells. This section lists the important respects in which AFS and UNIX do differ, and (in most cases) also refers you to a more detailed discussion.

2.1.1. 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 allows users to define a large number of their own groups, which can appear simultaneously on an ACL.

2.1.1.1. 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 defines seven access rights rather than the three possible with mode bits. Section 19.2.3 defines the rights.
- AFS uses only the "owner" mode bits, whose meaning interacts with rights on the directory's ACL. See section 19.11.
- A directory's ACL protects all of the files in a directory in the same manner. To apply a more restrictive set of AFS rights to certain file, place it in directory with a different ACL.
- Moving a file to a different directory changes file protection. See section 19.2.2.
- An ACL can include about 20 entries granting different combinations of rights to different users or groups, rather than only the three UNIX users/groups (owner, group, and world). See section 19.2.1.3.
- You cannot make an AFS directory "write only" as is possible by turning on only the `w` mode bit on a UNIX directory or file. To exercise the ACL WRITE right, the possessor must also have the LOOKUP and READ rights. However, it is possible to create write only files by manipulating the UNIX mode bits.

2.1.1.2. Groups

Another difference between UNIX and AFS protection is that AFS allows users to define the groups of other users. Placing these groups on ACLs extends the same rights 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 chapter 18.

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 sections 18.2.5 and 19.2.5.

2.1.2. Differences in Authentication

Just as the AFS file system is separate from each machine's local UNIX file system, the AFS "authentication space" is separate from the UNIX space. This has two practical implications.

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

  AFS provides a modified version of the **login** program that accomplishes both local login and AFS authentication in one step, based on a single password. If you choose not to use the AFS version of **login**, your users must use a multistep login process, as detailed in the *AFS User's Guide*. See section 2.11.2 of this Guide for a further discussion of the AFS **login**.

- 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 may differ if desired, though the effect depends if and how the cell is using AFS **login**. See section 2.11.2.

2.1.3. Differences in the Semantics of Standard UNIX Commands

This section summarizes differences in the functionality of some commonly issued UNIX commands.

chown Only members of *system:administrators* may issue this command on files stored in AFS. An exception: under Ultrix, only users logged into the UNIX file system as "root" may issue this command on AFS files; AFS authentication status is irrelevant.

chgrp Only members of *system:administrators* may issue this command on files in AFS. An exception: under Ultrix, only users logged into the UNIX file system as "root" may issue this command on AFS files; AFS authentication status is irrelevant.

ftpd The AFS version of this daemon attempts to authenticate remote issuers of **ftp** with the local AFS Authentication Server. See section 2.15.

groups If the user has issued AFS **login** or **pagsh**, the output of this command will include two large numbers, necessary for the implementation of PAGs. See section 2.11.2 for more on PAGs.

inetd The AFS version of this daemon authenticates remote issuers of AFS-modified **rcp** and **rsh** with the local AFS Authentication Server. See section 2.15.

login the AFS version of this daemon both logs the issuer into the local UNIX file system and authenticates the user with the AFS Authentication Server. See section 2.11.2.

ln under AFS, this command cannot create hard links between files in different directories. See section 2.1.5.

rcp the AFS version of this command allows the issuer to access files on the remote machine as an authenticated AFS user. See section 2.15.

rlogind the AFS version of this daemon authenticates remote issuers of **rlogin** with the local AFS Authentication Server. See section 2.15.

rsh the AFS version of this command allows the issuer to execute commands on the remote machine as an authenticated AFS user. See section 2.15.

2.1.4. The AFS version of the fsck Command

⇒ Never use the standard UNIX *fsck* command on an AFS file server machine.

Standard **fsck** will remove all of the V files (volume headers) that it finds in the */vicep***x** directories, and place them in *lost+found*. **This essentially erases all of your volumes**.

Instead, use the AFS version of the **fsck** program provided by Transarc. It functions like **fsck** on both UNIX and AFS files, and does not disturb AFS files like standard **fsck** does.

You can tell you are running the AFS version of **fsck** because it identifies itself with a banner as it initializes. The banner may look something like the following:

```
---Transarc AFS (R) 3.3 fsck---
```

The *AFS Installation Guide* explains how to replace **fsck** with the AFS version.

2.1.5. Creating Hard Links

AFS does not allow hard links (created with the UNIX **ln** command) between files that reside in different directories. Allowing this would make it unclear which ACL to associate with the link, since the ACLs on the two directories could differ.

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. However, you must not create a symbolic link to a file that begins with either a '#' or '%' because AFS will believe that the link is actually a mount point to a regular or ReadWrite volume, respectively.

2.1.6. AFS Implements "Save on Close"

Upon issue of the UNIX "*close*" or "*fsync*" system call, the AFS Cache Manager sends file modifications to the File Server for permanent storage in the central copy of the file kept there and in the file server's non-volatile storage.

Upon issue of the UNIX "*write*" system call, modifications are stored in the Cache Manager's local cached copy only. If the local machine crashes or an application program exits without issuing the "*close*" system call, changes will be saved in the locally cached copy only.

Most application programs issue "*close*" upon completion or automatically upon exit.

2.1.7. Setuid Programs

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

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

The **fs setcell** command determines whether setuid programs that originate in a foreign cell may run on a given client machine. See section 13.5.

2.2. Choosing a Cell Name

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

2.2.1. How to Choose a Cell Name

Your cell name should distinguish your cell from all others in the AFS global name space. 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 trees. For instance, both the Transarc and Andrew cells can have a "*usr/joe*" home directory because they are distinguished as */afs/transarc.com/usr/joe* and */afs/andrew.cmu.edu/usr/joe*.

By convention, cell names follow the ARPA Internet Domain System conventions for site names. If you are already an Internet site, then choosing a cell name is very simple—it is your Internet domain name.

If you are not an Internet site, you should choose a unique Internet-style name, particularly if you plan to join the Internet community in the near future. AFS Product Support is available for help in selecting an appropriate name. There are few constraints on the form of an Internet domain name that can be used as the name of an AFS cell:

- It can contain as many as 64 characters, but shorter names are usually better because you and others will be typing your cell name frequently as part of machine and file names. If your cell name is long, one way to save typing is to create a symbolic link at the second level in your file tree between an abbreviated name and the full name; see section 2.4.2.
- For portability between different operating system types, it should include only lowercase characters, numbers, underscores, dashes, and periods; it cannot include shell metacharacters.
- It can include any number of fields, which are conventionally separated by periods (see the examples below).
- It should end in a suffix that indicates the type of institution you are. Some of the standard suffixes are

  .com for businesses and other commercial organizations

  Example: *transarc.com* for Transarc Corporation

  .edu for educational institutions such as universities

  Examples: *andrew.cmu.edu* for the Andrew Cell at Carnegie Mellon University, *testers.athena.mit.edu* for one of the Project Athena cells at the Massachusetts Institute of Technology

  .gov for government institutions

  Example: *alw.nih.gov* for a cell at the National Institute of Health

  .mil for military installations

  Example: *cmf.nrl.navy.mil* for a cell at the Naval Research Laboratory

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 at (800) 235-3155. The NIC can also provide you with the forms necessary for registering your cell name as an Internet domain name. The advantage of registering your name is that it prevents another Internet site from adopting the name later.

There is no charge for registering, nor do you actually have to be connected to the Internet. However, you do have to provide them with the names of two machines actually connected to the Internet that will act as "domain name servers" for your site (a domain name server stores and makes available a record of how your site can be reached). The domain name servers do not actually have to be at your site, if you can find another site willing to perform that service for you.

2.2.2. 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 section 2.2.3. 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 */usr/afs/etc/ThisCell* and */usr/afs/etc/CellServDB*. As described more explicitly in the *AFS Installation Guide*, you set the cell name in both by issuing the **bos setcellname** command on the first file server machine you install in your cell. You should never need to issue **bos setcellname** again. As you install additional file server machines, the Update Server distributes its copy of *ThisCell* and *CellServDB* to each machine.

For ***client machines***, the two files that record the cell name are */usr/vice/etc/ThisCell* and */usr/vice/etc/CellServDB*. 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 (of course, only the first option is available on the very first client you install). See chapter 15 for details.

You will 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, you should follow the complete set of instructions in the *AFS Installation Guide* for setting up 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.

You can set the default cell name used by most AFS commands **without** changing the local */usr/vice/etc/ThisCell* file by setting the AFSCELL environment variable to the cell name that you want to work in. When this variable is set, it overrides the value found in *ThisCell* for most AFS commands. It is worth setting this variable if you intend to do a significant amount of work in a foreign cell for a temporary period of time.

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 *ThisCell*, 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.

2.2.3. Why Choosing the Appropriate Cell Name is Important

Take care to select a cell name that will be satisfactory for a long time to come. 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 throughout the AFS global name space, no matter how similar cells' file trees are at lower levels. For instance, it means that every cell could have a home directory called */afs/**cell**/usr/smith* 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**) by default execute in the cell of the machine on which they are issued. The command interpreters check *ThisCell* on the local disk and then contact the database server machines listed in *CellServDB* for the indicated cell (the **bos** commands work differently because the issuer always has to specify which machine's BOS Server will execute the command).

ThisCell also determines which cell a user is authenticated in (receives a token for) by default when he or she logs in on that machine. The cell name also plays a role in security. AFS converts passwords into encryption keys before storing them in the Authentication Database (section 1.5 explains how such keys are used for security). Before converting the password into an encryption key, the Authentication Server combines it with the cell name found in *ThisCell*. The AFS **login** program uses the same algorithm to convert the user's password into an encryption key at login (it learns the cell name by looking in *ThisCell* on the machine where it is executing).

This method of converting passwords into encryption keys means that the same password will result in different keys in different cells. Thus, if the cell name ever changes, a system administrator must change every entry in the Authentication Database to force the Authentication Server to generate a new encryption key using the new cell name. The *ThisCell* and *CellServDB* files on every file server and client machine must also change.

If the administrator does not make the change everywhere, users cannot log in—the encryption keys produced by the login program would not match the keys stored in the Authentication Database—and many commands from the AFS suites will not work as expected.

2.3. Participating in the AFS Global Name Space

Participating in the AFS global name space 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 name space.

There are several general points to note. Further details appear below.

- Participation in the global name space 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 section 2.3.4.
- 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. Most cells choose to participate in both ways.
- You make your cell visible to others by advertising your database server machines. See section 2.3.2.
- 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, if you so choose. See section 2.3.3.

2.3.1. What the Global Name Space Looks Like

The AFS global name space 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 */afs* to indicate that they belong to the AFS global name space.

The second convention is a cell name appears as the second element in every pathname; it indicates where the file resides (that is, the cell in which a file server machine houses the file). The presence of a cell name in pathnames is what makes the global name space possible: because the cell name is guaranteed to be unique, its presence in the pathname guarantees that all AFS pathnames are unique. This further implies that the pathname uniquely identifies the file no matter which cell the file is viewed from. This remains true even if cells use the same directory names at lower levels in their local trees.

What appears at the third and lower levels in an AFS pathname depends on how a cell has chosen to arrange its file space. There are some suggested conventional directories at the third level; see section 2.4.3.

2.3.2. Making Your Cell Visible to Others

You make your cell visible to others by advertising your cell name and the names and (Internet) addresses of your database server machines. There are two places you can make this information available:

- In the file */afs/transarc.com/service/etc/CellServDB.export* maintained by Transarc Corporation. 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, your site contact should contact Transarc Corporation. It is up to individual cells to monitor this file for changes and make a new copy of the file when it does change.

- The file */afs/**cell**/service/etc/CellServDB.local* in your file tree (***cell*** is your cell name). This file lists the database server machines for your cell only, rather than for every cell that you know about.

Whenever you change the identity of your cell's database server machines, you should make sure these two files are updated. You will also need to update the copies of *CellServDB* files used by your file server machines (*/usr/afs/etc/CellServDB*) and client machines (*/usr/vice/etc/CellServDB*). See chapters 9 and 15, respectively, for more information about maintaining the file server and client versions of *CellServDB*.

Advertising your database server machines makes your cell visible by telling foreign Cache Managers how to reach your cell's VL Servers. These tell the Cache Managers which file server machine is currently housing a file. The Cache Manager contacts the File Server running on that machine to access it (the same sequence of steps used in accessing local files). Similarly, knowing how to reach a cell's Authentication Servers allows **login** programs running in foreign cells to obtain a token for the cell, provided the user has an Authentication Database entry in the cell.

Once you have advertised your database server machines, it is difficult to make your cell invisible again. You can remove your */afs/**cell**/service/etc/CellServDB.local* file and ask Transarc Corporation to remove your cell from */afs/transarc.com/service/etc/CellServDB.export*, but other cells probably have the cell name in their local *CellServDB* files already. The only sure way to make your cell invisible is to change the names and/or addresses of the database server machines, which of course requires changing them in all of your local file server and client machines' *CellServDB*s too.

However, making your cell invisible is not required in order to make it inaccessible. You can make your cell completely inaccessible to foreign users by removing *system:anyuser* from all ACLs at the top three levels of your file tree; see section 2.3.4.

2.3.3. Making Other Cells Visible in Your Cell

Three steps are necessary in making a foreign cell's file tree visible from a client machine in your cell. The first concerns your cell's file tree, the others the configuration of the client machine.

Requirement 1

You must mount the cell's `root.cell` volume at the second level in your cell's file tree, just below */afs*, using the **fs mkmount** command with the **-cell** argument. (Note that if `root.afs` is replicated, you must make the mount point in the ReadWrite version of that volume and release it; see the *AFS Installation Guide* for instructions.)

Requirement 2

Each client machine must mount AFS at the */afs* directory. The **afsd** program, which initializes the Cache Manager at each reboot, performs this mount automatically. It is recommended that you invoke **afsd** in */etc/rc* or its equivalent. (It is possible to mount AFS at a different directory on a given client machine by using the **-mountdir** argument on **afsd**. See the description of **afsd** in the *AFS Command Reference Manual*).

Requirement 3

The client machine must have a kernel-resident list of database server machines for each cell you wish to make accessible. The omission of a cell from this list prevents contact with it, even if Requirements 1 and 2 are met for it. This means that you can make different cells visible and accessible from different client machines, if you choose. It is, however, less confusing to configure all clients identically in this regard, particularly if your users work on multiple machines.

The file */usr/vice/etc/CellServDB* on every client machine's local disk lists the database server machines for the local and foreign cells. When the machine reboots, **afsd** copies the contents of *CellServDB* into kernel memory, which the Cache Manager consults. It is possible to add or alter entries in the kernel list directly, without rebooting, by using the **fs newcell** command.

Transarc Corporation maintains a global *CellServDB*, which lists the database server machines for all cells that have agreed to make this information available to others. Its pathname is

/afs/transarc.com/service/etc/CellServDB.export

Transarc will inform your Site Administrator by electronic mail of changes to this file. You may also want to check the file periodically.

See chapter 15 for more information on configuring a cell's knowledge of database server machines.

2.3.4. Granting and Denying Foreign Users Access to Your Cell

As mentioned in section 2.3.2, making your cell visible in the AFS global name space 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 *anonymous*, which means they have only the rights you grant to *system:anyuser* on each directory's ACL. Normally these rights are limited to LOOKUP and READ.

There are two ways to grant wider access to foreign users; you can:

- Grant additional rights to *system:anyuser* 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 account for specific foreign users, or at least entries in the Protection and Authentication Databases and local password file (which is the minimum necessary to allow users to login locally). You can then grant rights to these user names on ACLs, and the foreign users can authenticate with your local Authentication Server in order to obtain access under a local identity. It is not currently possible to place foreign user names on ACLs, nor to authenticate in a foreign cell without having an account in it.

2.4. Setting Up Your File Tree

This section summarizes the issues to consider when setting up your file tree. Section 2.5 discusses, among other things, how to set up volumes that correspond to a file tree's directories most efficiently.

As mentioned in section 2.3.1, AFS pathnames must follow a few conventions so the AFS global name space looks the same from any AFS client machine. There are corresponding conventions you should 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 pathnames to be configured in a certain way.

2.4.1. The Top */afs* Level

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

2.4.2. The Second "Cell" Level

The second convention is that just below */afs* you must 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. Actually, each such "directory" is a mount point to the indicated cell's `root.cell` volume. For example, in the Transarc Corporation cell, */afs/transarc.com* is a mount point for the Transarc cell's own `root.cell` volume and *andrew.cmu.edu* is a mount point for the Andrew (at Carnegie Mellon University) cell's `root.cell` volume. This is confirmed by output from the **fs lsmount** command.

```
% cd /afs/transarc.com
% fs lsmount /afs/transarc.com
'/afs/transarc.com' is a mount point for volume '#root.cell'
% fs lsmount /afs/andrew.cmu.edu
'/afs/andrew.cmu.edu' is a mount point for volume
     '#andrew.cmu.edu:root.cell'
```

To reduce the amount of typing necessary in pathnames, you may wish to create symbolic links with abbreviated names to the mount points of those cells your users will contact often (particularly the home cell). In the Transarc cell, for instance, */afs/tr* is a symbolic link to */afs/transarc.com*, as **fs lsmount** reveals.

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

2.4.3. 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 commonly appear at this level:

common 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 *ThisCell* and *CellServDB.*

public this directory is useful if you do not want to grant *system:anyuser* any rights to the *usr* directory also found at this level (in which user home directories reside), but you still want to allow your users to make selected information available to everyone.

This directory's ACL grants the LOOKUP and READ rights to *system:anyuser*, whereas *usr*'s ACL grants those rights only to *system:authuser* (users who have authenticated in your cell). This directory contains a subdirectory for each user in your cell.

When users want to make a file available to everyone, including foreign and unauthenticated users, they put it in their subdirectory in this directory. When they want to make a file available only to locally authenticated users, they can put it in their home directory or a subdirectory of it (since only members of *system:authuser* can pass through *usr* to get at the home directory).

service this directory contains files and subdirectories that help cells coordinate resource sharing. Transarc has proposed a standard for files and subdirectories in this directory; if you wish to comply, contact AFS Product Support at Transarc for a copy of the standard.

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

- *CellServDB.export*, a list of database server machines this cell knows about
- *CellServDB.local*, a list of the cell's own database server machines
- *passwd*, a copy of the */etc/passwd* file commonly found on the local disk of the cell's client machines
- *group*, a copy of the */etc/group* file commonly found on the local disk of the cell's client machines

sys_type a separate directory to store the binaries for each system type you use as a file server or client machine. You should name these directories using the standard system type names developed by Transarc Corporation (*rs_aix32*, *sun4c_411*, and so on). You must do so if you plan to use the *@sys* variable in pathnames (see section 2.8.5). Section 2.6 lists the system type names.

Each such directory has *bin*, *etc*, *usr* and other subdirectories to store what is normally found in */bin*, */etc* and */usr* directories on a local disk. Symbolic links can then be used to link those local directories on clients into AFS, which saves local disk space; see section 2.8.2. 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 contains the home directory of each user in your cell and any foreign users granted a local account. As discussed in the previous entry for *public*, you may wish to protect this directory so that only locally authorized users can access it. This keeps the contents of your user's home directories as secure as possible.

If your cell is quite large, you may find that directory lookup is slowed if you put all home directories in a single *usr* directory. Section 2.9.6 discusses some ways to divide user home directories in multiple grouping directories.

wsadmin contains prototype, configuration and library files for use with the **package** program.

2.5. Setting Up Your Volumes

This section discusses how to create volumes in ways that make administering your system easier. In many places, it refers you to chapter 5 for more detailed information.

2.5.1. Creating Volumes that Correspond to Your File Tree

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 */afs/**cell**/common* and */afs/**cell**/usr*.

It is not required to create a separate volume for every directory level in a tree. However, if you do, each volume tends to be smaller and thus 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 may be mounted at several locations, though this is not recommended because it distorts the strict hierarchy of the file tree, potentially causing confusion.

2.5.2. Restrictions on Volume Names

The AFS implementation of volumes imposes an absolute length limit of 31 characters on volume names. Because there must be room for the addition of a `.readonly` extension if you replicate a volume, the Volume Server does not allow you to create ReadWrite volumes with names over 22 characters in length.

Do not add the extensions `.readonly` and `.backup` to volume names yourself, even if they are appropriate. The extensions are reserved for ReadOnly and Backup versions of a volume, respectively. The Volume Server adds them automatically when necessary.

2.5.3. Two Required Volume Names

There are two volumes that every cell must include in its file system, and which must be named as follows:

- `root.afs` for the volume corresponding to the top (*/afs*) level in the file tree.
- `root.cell` for the volume mounted just below */afs*, at the cell’s name (for example, at */afs/transarc.com* in the Transarc Corporation cell).

Deviating from these required names only creates confusion and extra work. Changing `root.afs`, for instance, means that the Cache Manager will not be able to find the volume which by default it mounts at */afs*. You would have to use the **-rootvol** argument when starting **afsd**, to make sure that the Cache Manager found the correct volume. If you forgot to invoke **afsd** with this argument, either in a machine's reboot configuration file (*/etc/rc* or equivalent) or on the command line, then the Cache Manager would not be able to access AFS.

Similarly, changing `root.cell` makes it difficult for other cells to make your file tree visible to their users. As section 2.3.3 discusses, the way to make a foreign cell's file tree visible in the local cell is to mount its `root.cell` volume at the second level in the local AFS file tree. If you change the name from `root.cell`, then other cell's attempts to see your file tree by mounting your "`root.cell`" volume in their tree will fail—the mount point will be created, but it will refer to a non-existent volume. Of course, if you want to make your cell invisible, this is one possible way to do it. You must also mount the volume with the changed name in your own cell's file tree, rather than mounting `root.cell`.

2.5.4. Naming Volumes to Make Administration Easier

You can name your volumes anything you choose, subject to the two simple restrictions mentioned in section 2.5.2. Adopting the right naming scheme for volumes can greatly ease administration, however. Two important qualities in a volume name are that it reflect the contents of the volume and that it be similar to the names of 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. The advantage of these qualities is that once someone understands the general pattern, they can accurately guess what a volume contains and where it is mounted, without having to issue a number of commands to figure those things out.

2.5.5. Using Prefixes on Related Volumes

Many cells find that the most effective volume naming scheme puts a common prefix on the names of all related volumes. You may wish to adopt the scheme summarized in the following chart.

Volume Prefix	Volume Type	Example Name	Example Mount Point
`common.`	common volumes	`common.etc`	*/afs/**cell**/common/etc*
`src.`	source volumes	`src.afs`	*/afs/**cell**/src/afs*
`proj.`	project volumes	`proj.portafs`	*/afs/**cell**/proj/portafs*
`test.`	test volumes	`test.smith`	*/afs/**cell**/usr/smith/test*
`user.`	user volumes	`user.terry`	*/afs/**cell**/usr/terry*
sys_type.	system volumes	`sun4c_411.bin`	*/afs/**cell**/sun4c_411/bin*

The following is a more specific example for a cell's *vax_ul43* system volumes and directories:

Example Name	Example Mount Point
`vax_ul43.bin`	*/afs/**cell**/vax_ul43/bin*
`vax_ul43.etc`	*/afs/**cell**/vax_ul43/etc*
`vax_ul43.usr`	*/afs/**cell**/vax_ul43/usr*
`vax_ul43.usr.afsws`	*/afs/**cell**/vax_ul43/usr/afsws*
`vax_ul43.usr.lib`	*/afs/**cell**/vax_ul43/usr/lib*
`vax_ul43.usr.bin`	*/afs/**cell**/vax_ul43/usr/bin*
`vax_ul43.usr.etc`	*/afs/**cell**/vax_ul43/usr/etc*
`vax_ul43.usr.inc`	*/afs/**cell**/vax_ul43/usr/inc*
`vax_ul43.usr.man`	*/afs/**cell**/vax_ul43/usr/man*
`vax_ul43.usr.sys`	*/afs/**cell**/vax_ul43/usr/sys*
`vax_ul43.usr.loc`	*/afs/**cell**/vax_ul43/usr/local*

There are several advantages to this scheme:

- The volume name is similar to the mount point name in the file tree (assuming that the prefix matches the name of the corresponding level in the file tree). In all of the entries in the chart, 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 reflects the contents, or at least suggests the directory on which to issue **ls** 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 allows 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. See section 5.6 for instructions for creating Backup versions and chapter 6 for more information on the AFS Backup System.
- It makes it easy to group related volumes together on a partition. Grouping related volumes together has several advantages of its own, discussed in section 2.5.6.

2.5.6. Grouping Related Volumes on a Partition

If your cell is large enough to make it practical, you may want to consider grouping related volumes together on the same partition of a file server machine. 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 goes down:

- If you keep a hardcopy record of the volumes on a partition, then when a file server machine goes down you know which volumes are unavailable. You could 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 may prevent you from accessing an online copy or from issuing the **vos listvol** command, which gives you the same information.
- It means that 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 could 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 would be 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 would affect everyone.

Admittedly, the need to move volumes for load balancing purposes can limit the practicality of grouping related volumes. The system administrator has to weigh the opposing advantages case by case.

2.5.7. When to Replicate Volumes

As discussed in section 1.2.8, replication refers to making a copy, or clone, of a ReadWrite "source" 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 crashes, users will still be able to access the volume 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 might be unduly "expensive," because each clone not on the same partition as the ReadWrite 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 whose contents change frequently. Clones are ReadOnly versions of the volume; they cannot change once they are copied from a source volume, even if the ReadWrite source volume does change. To keep a ReadOnly clone in sync with its changing ReadWrite source, you have to issue commands to make a new ReadOnly clone and distribute it to the different file server machines that house a clone of that volume.

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 "high-level" volumes. User volumes usually exist only in a ReadWrite version since they change so often.

2.5.7.1. Specific Volumes to Replicate

If possible, you should replicate your cell's `root.afs` and `root.cell` volumes at two or three sites, even if your cell only has two or three file server machines. The unavailability of these volumes (perhaps due to a server crash) makes all other volumes unavailable too, even if the file server machines storing the other volumes are still functioning. The Cache Manager needs to pass through the directories corresponding to `root.afs` and `root.cell` as it interprets any pathname.

Another reason to replicate `root.afs` is that it can lessen the load on the File Server machine. The Cache Manager has a built-in bias to access a ReadOnly version of `root.afs` if it is available. By accessing the ReadOnly version of `root.afs`, the Cache Manager gets on a "ReadOnly path." While on such a ReadOnly path, the Cache Manager requests files from the ReadOnly version(s) of a volume. And when distributing files from ReadOnly volumes, the fileserver process needs to maintain only one callback per volume, rather than one callback per file for ReadWrite volumes. Fewer callbacks translate into a smaller load on the File Server machine.

If a replicated `root.afs` is not available in your cell, local Cache Managers will have to follow the "ReadWrite path." While on the ReadWrite path, the Cache Manager can only request files from the ReadWrite version of a volume. When a File Server machine must distribute files from a ReadWrite volume, a large number of callbacks must be maintained (one for each copy of a file distributed). This puts a greater load on the File Server machine containing the ReadWrite volume.

For more on ReadWrite and ReadOnly paths, see section 5.7.1.

Other volumes to consider replicating are system binary volumes, the volume corresponding to */afs/**cell**/usr*, and the volume(s) corresponding to */afs/**cell**/common* and its subdirectories.

2.5.7.2. Where to Place ReadOnly Volumes

It is a good idea to release a ReadOnly clone to the same partition as the ReadWrite source because only the ReadWrite volumes takes up the full amount of disk space. The ReadOnly clone on the same partition as its source is actually like a Backup volume — it is a copy of the source volume's "vnode index."

This setup increases the availability of the volume content without requiring the storage costs of a non-clone ReadOnly volume (one that does not share the same partition as its ReadWrite source). Only if the ReadWrite volume moves to another partition or it changes substantially will the ReadOnly clone take up the full amount of disk space.

If your cell is sufficiently large, you may wish to dedicate a small set of file server machines to storing only ReadOnly volumes. Only one callback is required per ReadOnly volume, while a callback is required for each distributed file in a ReadWrite volume. Thus, by keeping very popular programs on the ReadOnly file server machines, you lessen the load on file server machines containing many ReadWrite volumes.

2.5.8. Smaller Volumes are More Efficient for Load Balancing

In general, smaller volumes are easier to administer and manipulate than larger ones. Once a volume is more than about 80 megabytes, it can be difficult to move it for load balancing purposes— many partitions may not have that much empty space available.

2.5.9. Space Quotas on Volumes

Every AFS volume has associated with it a quota that limits the amount of disk space the volume is allowed to use. As a system administrator, you can set and change volume quota using the commands described in section 5.12.

2.5.10. The Default Quota and ACL on a New Volume

By default, every new volume is assigned a space quota of 5000 kilobyte blocks. When you mount the volume in the file tree, the ACL on the volume's corresponding directory grants all seven access rights to *system:administrators*.

Section 17.5.2 explains how to change these values when creating an account with individual commands. When using **uss**, you can specify alternate values for ACL and quota on the template "V" line that creates a volume. See section 16.5.7.

2.6. Supported System Types

Transarc Corporation supports the following system types as both file server and client machines, and identifies each type with the indicated system type name. You are encouraged to adopt these names. Indeed, if you wish to use the *@sys* variable in pathnames, you must use the standard names to name the AFS directories where you store binaries for different system types. See section 2.8.5 to learn about the use of *@sys*.

System name	CPU and operating system type
hp300_ux90	Hewlett-Packard 9000 Series 300/400 running HP-UX 9.0
hp700_ux90	Hewlett-Packard 9000 Series 700 running HP-UX 9.0
hp800_ux90	Hewlett-Packard 9000 Series 800 running HP-UX 9.0
next_mach30	NeXT (68030 or 68040 systems) running NeXT 0S Release 3.0
pmax_ul43	DECstation 2100, 3100, or 5000 (single processor only) running Ultrix 4.3
pmax_ul43a	DECstation 2100, 3100, or 5000 (single processor only) running Ultrix 4.3a
rs_aix32	IBM RS/6000 running AIX 3.2
rt_aix221	IBM-RT/PC running AIX 2.2.1
rt_aos4	IBM-RT/PC running AOS Release 4
sgi_ir50	Silicon Graphics IRIS Indigo running IRIX 5.0.1 (client only)
sgi_ir51	Silicon Graphics IRIS Indigo running IRIX 5.1
sun3_411	Sun 3 (68020 systems) running SunOS 4.1.1, 4.1.2, or 4.1.3
sun3x_411	Sun 3 (68030 systems) running SunOS 4.1.1, 4.1.2, or 4.1.3
sun4_411	Sun 4 (*except* SPARCstations) running SunOS 4.1.1, 4.1.2, or 4.1.3
sun4_52	Sun 4 (*except* SPARCstations) running Solaris 2.2
sun4c_411	Sun SPARCstations running SunOS 4.1.1, 4.1.2, or 4.1.3
sun4c_52	Sun SPARCstations running Solaris 2.2
sun4m_412	Sun 600 Series MP SPARCstations running SunOS 4.1.2 or 4.1.3
sun4m_52	Sun 600 Series MP SPARCstations running Solaris 2.2
vax_ul43	Vax systems running Ultrix 4.3

Transarc also recommends names for system types that are not supported. An updated list appears in

/afs/transarc.com/service/systypes

Note that the presence of a suggested name in the file mentioned above does not imply that Transarc plans to support AFS on the associated system type.

2.7. Setting Up File Server Machines

Section 1.2.3 discusses how AFS divides machines into two classes, file servers and clients. This section discusses some issues to consider when setting up file server machines, which store your cell's AFS data and transfer it to client machines on request. File server machines also run the programs that maintain administrative databases.

To learn about client machines, see section 2.8.

2.7.1. The Supported System Types

AFS is available on a number of popular platforms. They are listed in section 2.6.

2.7.2. The Four Roles for a File Server Machine

As section 1.4 explains in detail, there are four roles a file server machine can fill:

- ***Simple file server machines*** only run the basic AFS server programs necessary for storing and delivering files. All file server machines take this role.
- ***System control machine***. A single machine that distributes its copy of common configuration files (in a cell running the United States edition of AFS) to keep them synchronized on all file server machines. It is generally also the one machine that synchronizes its clock with a time source outside the cell.
- ***Binary distribution machine***. There is one binary distribution machine of each CPU/operating system (CPU/OS) type which the cell uses as a file server machine. Each distributes its copy of the AFS binaries to the other machines of its type.
- ***Database server machine***. A small number of machines (generally no more than three) run the four "database server processes" that maintain databases—the Authentication, Protection, Volume Location, and Backup Servers.

See section 1.4 and the *AFS Installation Guide* for more information on how to configure each type of machine.

2.7.3. Your Cell's First File Server Machine

The *AFS Installation Guide* directs you to configure your cell's first file server machine to assume all four roles:

- Simple file server machine.
- System control machine.
- Binary distribution machine for its CPU/operating system type.

- Database server machine.

It is simplest if the first machine you install has the lowest Internet address of any machine you plan to use as a database server machine; because if later decide to use a machine with a lower Internet address as a database server machine, you must update the *CellServDB* file on all clients before introducing the new machine. See section 2.13 and the *AFS Installation Guide* for more information.

2.7.4. Installing Additional File Server Machines

When installing additional file server machines of an existing CPU/operating system type, you may find it convenient to make the machine a client first. You can then load the AFS server binaries onto the machine via AFS rather than from tape. The *AFS Installation Guide* provides complete instructions for installing additional file server machines.

2.7.5. Running Three Database Server Machines

A cell with three or more file server machines should run three database server machines to benefit from replicating the information in AFS's administrative databases. You will probably need no more than three database server machines. Cells with many file server machines find that three database server machines can handle all of the requests for information from the databases.

Obviously, a cell with one file server machine should run it as a database server machine.

A cell with two file server machines may or may not benefit from running both as database server machines. Having two machines yields the benefits of replication as long as both machines are running. However, if the machine acting as the Ubik coordinator (usually the one with the lower Internet address) goes down, the other one cannot distribute database information either—it is no longer in touch with the coordinator and cannot elect itself as the new coordinator, because it does not have the tie-breaking vote of the machine with the lowest address.

For more information on AFS's distributed database technology, see section 2.13.

2.7.6. Protecting Files on A File Server Machine's Local Disk

For security reasons, you should ensure that the directories and files under */usr/afs* on a file server machine give access only to their owner, "root." The */usr/afs/etc/KeyFile* file lists the AFS server encryption keys, so you do not want anyone attempting to read it except through the **bos listkeys** command. Similarly, */usr/afs/etc/UserList* controls privilege for **vos**, **bos**, and **backup** commands, so you do not want anyone attempting to alter it except through the proper **bos** commands.

The suggested protections are

Directory/File	Protections	Equivalent
/usr/afs	`drwxr-xr-x`	755
/usr/afs/backup	`drwx------`	700
/usr/afs/bin	`drwxr-xr-x`	755
/usr/afs/db	`drwx------`	700
/usr/afs/etc	`drwxr-xr-x`	755
/usr/afs/etc/KeyFile	`-rw-------`	600
/usr/afs/etc/UserList	`-rw-------`	600
/usr/afs/local	`drwx------`	700
/usr/afs/logs	`drwxr-xr-x`	755

2.7.7. Keeping All Binaries Available

It is recommended that you store the binaries for all AFS server processes in the */usr/afs/bin* directory on every file server machine, even if some processes run on a limited number of machines. This makes it easier to reconfigure a machine to fill a new role.

See chapter 3 for a description of the contents of all AFS directories on a file server machine.

2.7.8. Configuring Partitions to Store AFS Data

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

*/vicep**x***

where ***x*** is a letter of the alphabet. By convention, the first AFS partition created is mounted at */vicepa*, the second at */vicepb*, and so on.

Note: Each */vicep**x*** directory must correspond to an entire partition listed in */etc/fstab* or equivalent, and must be a subdirectory of the root directory (/). It is not acceptable to store AFS files in part of (for example) the */usr* partition and create a directory called something like */usr/vicepa*.

You may store non-AFS files on the partitions corresponding to */vicep**x*** directories, but it is not recommended. This creates competition between AFS and the local UNIX file system for access to the partition, particularly if the UNIX files are frequently used.

2.7.9. Automatic Process Restarts

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 */usr/afs/bin*. It checks for files with time stamps later than the time at which the corresponding process last restarted. If it finds any new binaries, it restarts the corresponding process to start using them.

Restarting processes causes a service outage, so the default times are in the early morning hours when an outage is likely to disturb the fewest number of people. You may list the restart times on a per-machine basis with **bos getrestart**, and set them with **bos setrestart**. The latter command allows you to disable automatic restarts entirely, by setting the time to "*never*".

For more information on automatic restarting, see section 4.9.

2.7.10. Limiting Reboots

Rebooting a file server machine requires shutting down the AFS processes first and so inevitably causes a service outage. You should reboot file server machines as infrequently as possible. This means that it is unwise to run a file server machine as a client that you will have to reboot frequently.

For instructions on rebooting file server machines, see section 4.11.

2.7.11. Monitoring File Server and Cache Manager Processes

AFS comes with two main monitoring tools that run on client machines:

- ***Scout*** – monitors the status of file server processes and gathers data on those processes.
- ***AFSMonitor*** – monitors the status of both file server and cache manager processes and gathers data on those processes.

These tools will also alert you when configured process performance limits are exceeded.

Neither the **scout** nor the **afsmonitor** programs impose much load on the File Server processes, so there is little reason to restrict their use even by regular users. Only if hundreds of users are polling the same File Server process would its performance suffer. Nevertheless, you can restrict use of **scout** and/or **afsmonitor** by placing the appropriate binary files in a directory unavailable to regular users.

See chapter 7 for more information about **Scout** and **AFSMonitor**.

2.8. Setting Up Client Machines

This section summarizes issues to consider as you set up client machines in your cell.

2.8.1. The Supported System Types

AFS is available on a number of popular platforms. They are listed in section 2.6.

2.8.2. Configuring the Local Disk

You can save disk space on the machine's local disk by storing in AFS those files that are normally kept on the machine's local disk. You then create a symbolic link on the local disk that refers to the AFS file. You may find the *@sys* pathname variable useful in links to system-specific files; see section 2.8.5.

Two basic classes of files actually must reside on the local disk: boot sequence files needed before AFS is accessible during reboot (that is, before **afsd** is invoked), and files that can be helpful during file server machine outages.

2.8.2.1. Files Needed During Reboot

During a reboot, AFS is inaccessible until **afsd** executes and reinitializes the Cache Manager (normally **afsd** is invoked in */etc/rc* or equivalent). Any files needed during reboot prior to that point must reside on the local disk. **The following list should not be considered inclusive.**

- standard UNIX files needed during reboot, which include the following or their equivalents:
  - */etc/rc*
  - */etc/fstab*
  - */etc/mount*
  - */etc/umount*
- all subdirectories and files found under */usr/vice,* including
  - */usr/vice/cache*
  - */usr/vice/etc/afsd*
  - */usr/vice/etc/cacheinfo*
  - */usr/vice/etc/CellServDB*
  - */usr/vice/etc/ThisCell*

  More information on these files appears in chapter 12.
- files needed by **dkload**, including

– */bin/as*

– */bin/ld*

– */bin/nm*

2.8.2.2. Diagnostic and Recovery Files

Certain commands in the **fs** and **bos** command suites can help users diagnose and recover from problems caused by file server outage. It is useful to have local disk copies of the binaries for those suites, since the file server outage that requires their use can also make them inaccessible.

You may wish to store the **bos** and **fs** binary files in */usr/vice/etc* as well as */usr/afsws*, which is normally a link into AFS. Then set PATH variables so that */usr/afsws* appears before */usr/vice/etc*. That way, the user will access the */usr/afsws* copy when the file servers are accessible; being in AFS, it is more likely to be up-to-date than a local copy.

You may also wish to keep the binaries for a text editor (such as **ed** or **vi**) on the local disk for use during outages.

2.8.3. Using the package Program to Configure Local Disks

The **package** program automates the configuration of the local disk on client machines, which can save a tremendous amount of time. It works by updating the contents of the local disk to match definitions found in a "**package** configuration file." See chapter 14.

2.8.4. Client Access to Foreign Cells Can Vary

As detailed in section 2.3.3, you enable the Cache Manager to access a cell’s file tree by providing it a list of the cell’s database server machines. The list is kept in the file */usr/vice/etc/CellServDB*, and loads into kernel memory at reboot for easy retrieval by the Cache Manager. You can change the kernel list of a cell’s database server machine in between reboots with the **fs newcell** command.

The fact that access to cells relies on information on the local disk and in kernel memory means that the "view" of the AFS global name space can differ from client to client in your cell. By including a cell in one machine’s kernel list but not another’s, you make that cell’s file tree accessible to the former machine but not to the latter machine.

For the sake of consistency, you should make the same cells accessible from every client. This is particularly recommended in cells where users work on different machines each day. You may wish to store a source version of *CellServDB* in AFS and use **package** periodically to update each client’s version with the source copy. See section 2.3.3 and chapter 15.

2.8.5. Using the *@sys* Variable in Pathnames

When constructing links into AFS on the local disk, you may want to use the "*@sys*" variable in some pathnames. When the Cache Manager encounters *@sys* in a pathname, it substitutes the local machine's CPU/operating system type for *@sys*. For example, the Cache Manager on a DECStation running Ultrix 4.3 performs the following translation

/afs/transarc.com/@sys —> */afs/transarc.com/pmax_ul43*

whereas a Sun SPARCstation running Sun OS 4.1.2 would interpret the same pathname as

/afs/transarc.com/@sys —> */afs/transarc.com/sun4c_411*

The Cache Manager learns the local machine's system type from a location in kernel memory. The **fs sysname** command displays the current value, and allows authorized users to change it.

Using *@sys* **requires that you use Transarc's recommended system type names when naming the system binary directories in your file tree.** Consult the list in section 2.6.

The advantage of *@sys* is that you can place the same links on machines of different system types and still have each machine access the files appropriate to its system type. However, you should use *@sys* carefully and sparingly, as it can make the traversing of pathnames unpredictable. Its use is most appropriate when you do not know what the current system type might be. Specifying an actual system type in most pathnames is recommended. If you use *@sys*, restrict its use to only one level in the file tree; the third level is a common choice, because that is where most cells store the binaries for different machine types.

Multiple instances of *@sys* in a pathname are especially dangerous to people who must explicitly **cd** into directories storing binaries for system types different from the local machine (such as administrators or developers who maintain those directories). After changing directories, such people should verify they are in the desired directory.

2.8.6. Setting a Cache Manager's Preferences for File Server Machines

The Cache Manager stores a table of preferences for file server machines in the kernel of the client machine. A preference is a file server machine's IP address and an associated "rank" (an integer in the range from 1 to 65,534).

A file server machine's rank determines the Cache Manager's preference for selecting ReadOnly replicas that reside on that machine. The Cache Manager's preferences allow you to bias it to access replicas from file server machines situated "close" to the client machine instead of those replicas on "distant" machines.

The **fs getserverprefs** command can be used to display a Cache Manager's preferences. The **fs setserverprefs** command can be used to set a Cache Manager's preferences for one or more file server machines. See section 13.11 for more information about server preferences and the associated commands.

2.9. Setting Up AFS User Accounts

This section discusses some of the issues to consider when setting up AFS user accounts. Because AFS is separate from the UNIX file system, an AFS account is separate from any UNIX account the user may already have. An AFS user account has several components, which are listed in section 2.9.2. You can create them in one of two ways.

2.9.1. The Two Methods for Creating AFS User Accounts

The preferred method for creating a user account is with the **uss** suite. 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 chapter 16.

The other method is to issue the individual commands to create each component of the account. Chapter 17 provides instructions, along with instructions for removing user accounts and changing user passwords, storage quotas and user names.

2.9.2. The Components of an AFS User Account

Below is a list of the components of an AFS user account. Sections 16.2 and 17.2 also list these components and describe them in greater detail. The components are

- 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 rights on its ACL.
- An entry in the local password file (*/etc/passwd* or equivalent) of each machine the user will log into.
- Optionally, standard files and subdirectories that you as an administrator have provided to make the account more useful.

2.9.3. Levels of Account Functionality

You can create accounts at different levels of "completeness" or functionality. Three common levels are

- The user only has entries in the Authentication and Protection Databases. He or she is able to authenticate in AFS, but has no home directory/volume. Such an account gives the user the ability to access files as a locally authenticated user (instead of only as *system:anyuser*), but not the ability to store anything in a personal home directory. This level can be useful for "dummy" administrative accounts and guest accounts for users from foreign cells.

- In addition to the necessary Database entries, the user has a home directory/volume which he or she owns and has full access rights to. The account is otherwise "bare bones," without any standard directories or configuration files that may help with printing, logging on and off, or mail delivery.
- The account includes standard directories and configuration files in the home directory, to make it more user-friendly. Most users, especially novices, will find an account like this much more usable than the "bare bones" type.

Sections 16.5.2 and 17.5.1 further discuss the implications of these levels and explain how achieve them (with **uss** and manual creation, respectively).

2.9.4. Choosing User Names and Naming Other Account Components

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

2.9.4.1. User Names

You may choose any naming scheme you wish for user names. By convention, many components of the account share this name, including the entries in the Protection and Authentication Databases, the volume, and the mount point. Preferably, the name gives some indication of the user's identity (which often implies that you cannot allow users their own free choice of user names). Depending on your electronic mail delivery system, the user name may become part of the user's mailing address. The user name is also the string that the user types at the **login** program's prompt.

Some common choices for user names are last names, first names, initials, first name with initial of last name, first initial with last name, or initials combined with sequential or randomly generated numbers. First names will probably work well only in very small cells. The last choice is often appropriate in a very large cell where several users share the same two initials or even the same last name.

To comply with UNIX restrictions, user names cannot be longer than 8 characters, nor can they contain any of the following characters:

- The period (**.**). AFS reserves its use for special administrative "instance" names used internally.
- The comma (**,**).
- The colon (**:**). AFS reserves the colon as a field separator in protection group names. See section 2.10.3.
- The semicolon (**;**).
- The "at-sign" (**@**). This character is reserved for Internet mailing addresses.
- Spaces.
- The new-line character.

2.9.4.2. AFS UIDs and UNIX UIDs

Each AFS user name in a cell is associated with an AFS UID, which is a number that has the same function in AFS as the UNIX UID has in the UNIX file system. It uniquely identifies the user to AFS server processes and the Cache Manager; they use it internally when interacting with the user, rather than the user name.

AFS users also must have a UNIX UID, recorded in the local password file (*/etc/passwd* or equivalent) of each client machine they will log into. For ease of administration, **a user's AFS UID and UNIX UID should be the same.** One important reason is that it allows for the correct calculation of file ownership (in the listing from **ls -l**, the listed owner matches the AFS user name).

In most cases, you should allow the Protection Server to allocate an AFS UID as it creates the Protection Database entry. However, both **uss** and the manual creation method allow you to assign AFS UIDs explicitly. The two most common reasons to assign AFS UIDs are:

- 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. See section 2.9.5.

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. You can, however, change this value by resetting the `UserMax` counter before creating any user accounts. The Protection Server uses `UserMax` to track the next available AFS UID, and you can reset it with **pts setmax** at any time (not just soon after initialization). You can examine the current value of `UserMax` with **pts listmax**. See section 18.13.

2.9.4.3. A Reserved AFS UID: *anonymous*

One user AFS UID (32766) is reserved, the one for *anonymous*. The AFS server processes assign this identity and AFS UID to any user who does not possess a token. You should not assign this AFS UID to any other user, nor should you hardcode its current value into any programs or fields (such as the UNIX owner field on a file), as this number may change in future releases.

2.9.4.4. Volume Names

As with user names, you have a free choice in naming user volumes. Like other volume names, user volume names are subject to the 22 character length restriction and should not have a `.readonly` or `.backup` extension on their ReadWrite version. See section 2.5.2 for further discussion.

You should include the user name in the volume name to help you identify it easily. The name suggested for user volumes in section 2.5.5 is "`user.`*user_name*".

2.9.4.5. Mount Point Names

By convention, the mount point for a user's volume is the user name. Many cells follow the convention discussed in section 2.4.3 and mount user volumes in the */afs/**cell**/usr* directory. Very large cells may find that mounting all user volumes in the same directory slows directory lookup, however. Section 2.9.6 suggests alternatives to a single *usr* directory.

2.9.5. Converting Unix Accounts into AFS Accounts

Many cells have UNIX user accounts that predate the introduction of AFS in the cell, and wish to convert these account into AFS accounts. There are three main issues to address when performing such conversions:

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

Because **uss** allows you to assign explicit AFS UIDs, it is possible to use that program when converting accounts; see section 16.4. Section 17.4 explains how to convert accounts manually.

2.9.6. Grouping Home Directories

The suggested directory for mounting user volumes is */afs/**<cellname>**/usr*, an AFS-appropriate variation on the standard UNIX practice of putting user home directories under */usr*. However, cells with a large number of users (that is, more than several hundred) may find that mounting all user volumes in a single directory slows 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 real-world divisions, such as academic or corporate departments. A company might establish group directories called *usr/marketing*, *usr/research*, *usr/finance*, for instance. 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 the directory 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 real-world divisions.
- Distribute home directories into alphabetic subdirectories of *usr* (*usr/a*, *usr/b* and so on), based on the first letter of the user name. This scheme has the same advantages and disadvantages of a department-based scheme. Anyone who knows the user's user name can find the user's home directory, but users with names that begin with popular letters may experience slower lookup.
- Distribute home directories randomly but evenly into more than one grouping directory, not along real-world divisions. One cell that uses this scheme has over twenty such directories called *usr1*, *usr2* 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 */afs/bigcell/usr17/smith*, then */afs/bigcell/usr/smith* would be a symbolic link to *../usr17/smith*. This way, if someone does not know which directory *smith* is in, he or she can access it as *usr/smith*; people who do know the appropriate directory save lookup time by specifying it.

Section 16.5.7.5 describes how to implement these types of groupings using **uss**. The program has a built-in feature for random distribution (the "G" line).

2.9.7. Making a Backup Version of User Volumes Available

AFS provides a simple mechanism that allows users themselves to restore data they have accidentally removed or deleted: the Backup version of a volume. You create a Backup version of the user's volume and mount it at a subdirectory in their home directory (called perhaps *OldFiles*). At the end of every day you create a new Backup version to capture the changes made that day, and overwrite the previous day's Backup version with the new one. Users can always retrieve the previous day's copy of a file without bothering you, leaving you more time to deal with more pressing tasks.

Note that the data in a Backup Volume does not count in the user volume's total quota, because it is a separate volume. The only space the Backup volume uses in the user volume is the amount needed for the mount point.

For further discussion of Backup volumes, see sections 2.14 and 5.6.

2.9.8. 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 *.login* and *.cshrc*) to make an account easier to use.

You may wish to add some AFS-specific directories to the definition of the user's PATH environment variable, including

- The path to a *bin* subdirectory in the user's home directory (that is, */afs/**cell**/usr/**user_name**/bin*)
- */usr/afsws/bin*, which conventionally includes programs like **fs**, **klog**, **kpasswd**, **login**, **pts**, **tokens**, and **unlog**
- */usr/afsws/etc*, if the user is an administrator; it usually houses the AFS command suites that require privilege (**backup**, **butc**, **kas**, **uss**, **vos**), the **package** program, and others

If you are not using the AFS version of **login**, you may wish to include commands in *.login* to create a PAG (see section 2.11.1) and authenticate the user with AFS, as in

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

This sequence creates a PAG, then echoes the string "`klog`" to the screen before executing the **klog** command (which produces a "`Password:`" prompt at which the user provides his or her AFS password.

If you use AFS **login**, this sequence is not necessary, as that command performs all the equivalent steps automatically. See section 2.11.2.

2.9.9. Removing User Accounts

When users leave your system, you can remove their accounts to free up space in your file tree. There is no **uss** command for removing user accounts, so you must remove account components individually. Instructions appear in section 17.10.

2.10. Using AFS Protection Groups

AFS allows users to define their own ***groups*** of other users. The groups are placed on ACLs to grant the same rights to many users without listing each user individually. Chapter 18 discusses the creation and use of groups. The section summarizes some of the issues relevant to groups.

2.10.1. Group AFS UIDs

Groups have AFS UIDs, just as users do; but a group AFS UID is a negative integer whereas a user AFS UID is a positive integer. By default, the Protection Server allocates the AFS UID for a new group, but the **pts creategroup** command allows members of *system:administrators* to assign AFS UIDs if desired. As with user UIDs, allowing the Protection Server to allocated group UIDs itself is preferred. Before explicitly assigning a group UID, the issuer may wish to verify that it is not already in use.

2.10.2. The Three System Groups

As the Protection Server initializes for the first time on a cell's first database server machine, it automatically creates and assigns AFS UIDs to three group entries: *system:anyuser*, *system:authuser*, and *system:administrators*. It then sets the `GroupMax` counter it uses in allocating group AFS UIDs to be larger (more negative) than these AFS UIDs, avoiding conflict.

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

- Everyone who access a cell's file tree is considered a member of *system:anyuser*. They may be logged into a machine inside or outside the cell, and may be authenticated or not. Placing *system:anyuser* on an ACL grants access to the widest possible range of users.
- Everyone who is currently authenticated in the local cell is considered a member of *system:authuser*. Thus only users with a valid token gain the access granted to *system:authuser* on an ACL.

An implication of the lack of stable membership is that it is not possible to list *system:anyuser* and *system:authuser*'s current members. Similarly, these groups do not appear when **pts membership** is used to list the 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 may issue any **pts** command, and are the only ones who can issue several other restricted commands (such as **chown** on AFS files). They also have the implicit ADMINISTER right on every ACL in the file tree even though *system:administrators* might not appear on an ACL. See sections 18.2.5 and 19.2.5 for more information.

2.10.3. The Two Types of User-Defined Groups

All users may create "regular" groups. The name of these groups has two fields separated by a colon, the first of which must accurately reflect the group's ownership. The Protection Server refuses to create or change the name of a group if that action would result in an inaccurate first field in the name. The syntax is

owner-name:group-name

Members of *system:administrators* may create "prefix-less" groups whose names do not have the first, owner field. Such groups can be administratively convenient.

See section 18.2.2 for suggestions on using the two types of groups effectively.

2.10.4. Rules Concerning Groups

The practical limit on group size is 5000 members, because the **pts membership** command that lists group membership cannot display any more than that.

Groups cannot be members of other groups, but groups can own other groups. A group must already have at least one member in order to own another group.

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

Each Protection Database entry (group or user) is protected by a set of five "privacy flags" that limit who may administer the entry and what they may do. The default privacy flags are fairly restrictive, especially for user entries. See section 18.14.

The current owner of a group may transfer ownership of the group to another user without the new owner's permission. At that point the former owner loses administrative control over the group. No matter how many times ownership is transferred, a group's existence is always "counted against" the group-creation quota of its creator. When the group is removed from the Protection Database, its creator's quota increases by one.

2.11. Login and Authentication in AFS

Section 2.1.2 explained that AFS authentication is separate from UNIX authentication because the two file systems are separate. As mentioned in that section, the separation has two practical implications:

- To access AFS files, users must both log into the local machine's UNIX file system and authenticate with the AFS Authentication Server. (Logging into the local UNIX file system is necessary because the only way to access AFS 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 (*/etc/passwd* or equivalent) for the UNIX file system.

Transarc provides a modified version of the **login** program that authenticates a user in AFS and logs the user into the local UNIX file system in one step. If you do not use AFS **login**, your users must issue additional AFS authentication commands after logging into the UNIX file system. Using AFS **login** therefore makes login simpler for your users.

Note: AFS **login** is not compatible with the DecWindows environment provided by Digital Equipment Corporation (DEC), because it does not include the special features that DEC has incorporated into their version of **login**. It also may not include other features incorporated into proprietary versions of **login**. If you use DecWindows and would like to use AFS **login**, contact Transarc AFS Product Support for a possible solution.

This section details further how login and passwords work in cells that do and do not use AFS **login**. First, it explains the importance of the " process authentication group" (or PAG) in either type of cell.

2.11.1. Using PAGs to Identify AFS Tokens

Several users can log into an AFS client machine at the same time and obtain separate AFS token(s). To make sure that each user accesses AFS with the proper authorization, the Cache Manager needs a way to track which token belongs to which user. (Recall that in AFS a ***token*** is a small collection of data that the AFS Authentication Server grants to users who have proved their identity by providing the correct AFS password. AFS server processes require their clients to present a token when requesting service, and use it to establish that the user is genuine. See section 1.5 to review this "mutual authentication" procedure.)

The Cache Manager tracks tokens by storing them in a separate "credential structure" for each user who is currently logged into the machine. There are two alternate ways the Cache Manager can associate a credential structure with a user: either by the user's UNIX UID or by a ***PAG***, which stands for "process authentication group". The PAG is preferred because it solves several potential security problems that can arise when the credential structure is based on a UID.

A PAG's superiority over a UID is due to its uniqueness. Like a UNIX UID, a PAG is simply a number, but unlike a UNIX UID, a PAG always uniquely identifies the user. It is never possible for more than one user to have the same PAG, because the Cache Manager creates PAGs based on a counter that increments with each use. In contrast, there are two situations in which two users of a machine can have the same UNIX UID. Both situations create a security risk.

- When two users working on different NFS client machines have the same UNIX UID (e.g., "10") in their respective local UNIX file systems. They both access the same NFS/AFS Translator machine. If the Cache Manager on the translator machine identifies them by UNIX UID, they become indistinguishable—they are both "10". The Cache Manager then has no way to tell which user is allowed to use which tokens. To solve 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.
- The second situation arises because UNIX allows anyone who is already "root" in the local UNIX file system to assume another identity (that is, UNIX UID) by issuing **su**, without providing the password associated with that identity. If the credential structure storing a token is identified by UNIX UID, then assuming the same UID means being able to use the token, too. Identifying the credential structure by PAG eliminates this possibility.

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 "root") that the AFS server processes recognize only as *anonymous*. Unless PAGs are used, such daemons cannot access files protected against *system:anyuser*.

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 **klog** before the PAG expires, the new token is associated with the existing PAG, and the PAG is "recycled."

AFS **login** automatically generates a PAG; see section 2.11.2. If you do not use AFS **login**, your users should issue the **pagsh** command that generates a PAG, before issuing **klog**. See section 2.11.3.

2.11.2. Using the AFS login Program

As previously mentioned, AFS **login** simultaneously authenticates a user in AFS and logs him or her into the local UNIX file system. This section describes how it works, and how its use affects what you should put in the password field in the local password file.

2.11.2.1. How AFS login Works

More precisely, the **login** program performs the following sequence of steps:

1. It checks the user's entry in the local */etc/passwd* file.

   If no entry exists, or if an asterisk (`*`) appears in the password field, the login attempt fails.

   If the entry exists, the attempt proceeds to step 2.

2. It obtains a PAG.

3. It 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 an AFS Authentication Server. The Authentication Server decrypts the packet and, depending on the success of the decryption, judges the password to be correct or incorrect. (See section 1.5 for more details.)

   If the Authentication Server judges the password incorrect, the user does not receive an AFS token. The PAG still exists, ready to be associated with any tokens obtained later. The attempt proceeds to step 4.

   If the Authentication Server judges the password correct, it issues a token to the user as proof of AFS authentication. The **login** program also logs the user into the local UNIX file system. It echoes the following banner to the screen to alert the user to authentication with AFS. Step 4 is skipped.

   ```
   AFS 3.3 Login
   ```

4. If no AFS token was granted in step 3, the **login** program attempts to log the user into the local UNIX file system, by comparing the password provided to the local password file (*/etc/passwd*, for instance).

   - If the password is incorrect or an uppercase `X` appears in the password field, the login attempt fails.
   - If the password is correct, the user is logged into the local UNIX file system only.

2.11.2.2. What to Put in the Local Password File When Using AFS login

As section 2.11.2.1 demonstrates, the password field in a machine's password file is no longer the primary "gate" for access to your system if you use AFS **login**. If the user provides the correct AFS password, then AFS **login** never consults the local password file. However, you can still use the password field to control access, in the following way:

- Putting an asterisk (`*`) in the field prevents the user from either authenticating with AFS or logging into the machine UNIX file system. This is useful in emergencies, when you want to prevent certain users from logging into the machine.

- Putting a string of any length other than the standard thirteen characters in the field means that if the user does not provide the correct AFS password, they will also not be allowed to log into the local machine's UNIX file system. This is appropriate if you want to make sure that only people with AFS accounts in your cell can log into your machines. Putting a single "X" or other character is the simplest choice.
- Putting an actual (scrambled) password in the field with **/bin/passwd** means that if the user does not provide the correct AFS password, he or she can still login to the local UNIX file system by providing the correct UNIX password. The AFS and UNIX passwords may differ, but it is less confusing if they do not.

2.11.3. Not Using the AFS login Program

If your cell does not use the AFS version of **login**, then users must execute a series of steps to login and authenticate:

1. They must login to the local UNIX file system with standard **login**, providing the password listed in the local password file.
2. They must issue the **pagsh** command to generate a new PAG.
3. They must issue **klog** to authenticate with the AFS Authentication Server.

The *AFS User's Guide* provides complete instructions.

Section 2.9.8 explains how you could put **pagsh** and **klog** in *.login* or a similar file to save users from typing the two commands at each login. They will still have to type a password twice, once for **login** and once for **klog**. This implies that the two passwords could differ, though it is less confusing if they do not.

Another effect of not using AFS **login** is that the standard **login** program runs as *anonymous*. If any files in AFS must be accessible to **login** (such as a *.login* file in the user's home directory), then the files must be accessible to unauthenticated users—*system:anyuser* must have the READ and LOOKUP rights on the directory where the file resides.

2.11.3.1. What to Put in the Local Password File When Not Using AFS login

An actual (scrambled) password must appear in the local password file for each user when you do not use AFS **login**. Use **/bin/passwd** 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.

2.11.4. Limiting the Number of Failed Authentication Attempts

Unauthorized users may attempt to gain access to a user account in your AFS cell by trying to guess the corresponding user's password. To protect against this type of unauthorized access, you can limit the number of times that a user can consecutively fail to authenticate (by incorrectly typing in his or her password) whether the user is using AFS **login** or **klog**. Should a user (whether the legal user or an unauthorized user) exceed this limit, the user account will lock up, preventing anyone from authenticating with that user account for a specified period of time.

The limit on the number of allowable failed authentication attempts and the resulting lockout time are defined using the **kas setfields** command. A system administrator (with an `ADMIN` flag in his or her Authentication Database entry) can unlock a user account that has been locked in this manner by using the **kas unlock** command.

See section 17.6 for instructions on setting this limit. Consult the *AFS Command Reference Manual* for detailed information on the **kas setfields** and **kas unlock** commands.

2.11.5. Changing Passwords

A regular AFS user can change his or her password using either the **kpasswd** or **kas setpassword** command. To change passwords, the user must prove his or her identity by typing in his or her current password. The user is then prompted to enter the new password twice (to screen out typing errors).

A system administrator can change any user password. To do this, the system administrator can use

- **kpasswd**, if he or she knows the user's current password.
- **kas setpassword**, if the `ADMIN` flag appears in the system administrator's Authentication Database entry.

Note: Neither **kpasswd** nor **kas setpassword** affect the local password file. Use **/bin/passwd** instead.

See section 17.7 for more instructions on changing passwords. Consult the *AFS Command Reference Manual* for detailed information on the **kpasswd** and **kas setpassword** commands.

2.11.5.1. Password Expiration

By default, passwords never expire: a user can go on using the same password without ever having to change it. For security reasons, however, you may want to require users to change their passwords after a given period of time.

The **kas setfields** command allows you to specify a lifetime for a user's password beyond which the password becomes invalid. The password lifetime can range between 1 and 254 days.

Once a user's password expires, he or she will be unable to authenticate with his or her account. However, for up to thirty days after the expiration of a password, a user can still use that password to change passwords. Beyond that, a system administrator will need to change the user's password.

See section 17.6 for instructions on setting password lifetimes. Consult the *AFS Command Reference Manual* for detailed information on the **kas setfields** and **kas unlock** commands.

2.11.5.2. Restricting Password Reuse

People are creatures of habit, and some people would like to go on using the same password forever. However, reusing the same password for long periods of time makes a user's account progressively more vulnerable to unauthorized entry. You can prevent users from reusing recently used passwords with the **kas setfields** command.

See section 17.6 for instructions on restricting password reuse. Consult the *AFS Command Reference Manual* for detailed information on the **kas setfields** command.

2.11.5.3. Checking Password Quality

From a security point of view, some passwords are better than others. You can automatically check the quality of every new user password created with either the **kpasswd** or the **kas setpassword** command. The quality check is made by a program called *kpwvalid* which must reside in the same directory as the **kpasswd** and **kas setpassword** programs.

You can create your own *kpwvalid* program or use the version provided by Transarc Corporation. See the entry on **kpasswd** or **kas setpassword** in the *AFS Command Reference Manual* for information on the general specifications of this program and the example program provided by Transarc Corporation.

If you create a custom *kpwvalid* program or shell script, the types of quality checks made are left to your discretion. Below are some suggested checks:

- Check to make sure the password is at least *n* characters long.
- Check to make sure that the password is not a word.
- Check to make sure that the password contains both numbers and letters.

See section 17.6 for instructions on checking password quality. Consult the entries on **kpasswd** and **kas setpassword** in the *AFS Command Reference Manual* for detailed information on *kpwvalid*.

2.11.6. Using pagsh to Obtain a New PAG

The **pagsh** command obtains a new PAG with which the issuer's credential structure is then identified. If the issuer already has a PAG, the new PAG is still created and the old one remains in place. Any jobs currently running started using the old PAG continue to run using the tokens associated with the old PAG; any new jobs use the new PAG and its associated tokens. When the new PAG is exited, the user will return to the old PAG, assuming that it has not expired.

See also section 2.11.1 and the entry for **pagsh** in the *AFS Command Reference Manual*.

2.11.7. The One-Token-Per-Cell-Per-Machine-Per-PAG Rule

A user can have only one token at a time per machine per PAG for a given cell. To obtain a second token for the same cell, the user must either log into a different machine or create a new PAG. It is, however, possible to possess tokens on one machine and PAG for many different cells (one token per cell). 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.

2.11.8. Using klog to Obtain a New Token

Once logged in, a user can obtain a token at any time with **klog**. 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, **klog** authenticates the user using his or her login name. The **-principal** argument allows the issuer to adopt a different identity (if that identity's password is known). The **-cell** argument allows authentication in other cells and can be combined with **-principal**.

For further information, see the *AFS User's Guide* and the entry for **klog** in the *AFS Command Reference Manual*.

2.11.9. Using tokens to Examine Tokens

The **tokens** command lists all of the tokens currently held by the Cache Manager on the local machine. The following examples illustrate its output in various situations.

The following shows the output when the issuer is not authenticated in any cell.

```
% tokens
Tokens held by the Cache Manager:

    [  1]   --End of list--
```

The following shows the output when the issuer is authenticated in Transarc Corporation cell, where he or she has AFS UID 1000.

```
% tokens
Tokens held by the Cache Manager:

    [  1]User's (AFS ID 1000) tokens for afs@transarc.com
                                       [Expires Jan 2 10:00]
    [  2]    --End of list--
```

The following shows the output when the issuer is authenticated in Transarc Corporation cell, Andrew cell at Carnegie Mellon University and Athena cell at MIT. The user has different AFS UIDs in the three cells. Tokens for the last cell are expired:

```
% tokens
Tokens held by the Cache Manager:

    [  1]User's (AFS ID 1000) tokens for afs@transarc.com
                       [Expires Jan 3 10:00]
    [  2]User's (AFS ID 4286) tokens for afs@andrew.cmu.edu
                       [Expires Jan 3 1:34]
    [  3]User's (AFS ID 22) tokens for afs@athena.mit.edu
                       [>>Expired<<]
    [  4]    --End of list--
```

Note: If your cell is using the Kerberos version of **tokens** (**tokens.krb**) the output from this command will also include information on the ticket-granting ticket used by Kerberos including: the owner of the ticket, the name of the ticket-granting service that issued the ticket, and the expiration date of the ticket. Below is an example of output from the Kerberos version of the **tokens** command:

```
% tokens.krb
Tokens held by the Cache Manager:

User's (AFS ID 1000) tokens for afs@transarc.com [Expires Jan 31 17:36]
User smith's tokens for krbtgt.TRANSARC.COM@transarc.com
[Expires Jan 31 17:36]
   --End of list--
```

For further information, see the *AFS User's Guide* and the entry for **tokens** in the *AFS Command Reference Manual*. See section 2.11.12 for more information on support for Kerberos authentication.

2.11.10. Using unlog to Discard Tokens

The **unlog** command discards tokens currently held for the issuer by the local machine's Cache Manager. You can discard all tokens or tokens for specified cells. For further information, see the *AFS User's Guide* and the entry for **unlog** in the *AFS Command Reference Manual*.

Note: Since tokens are granted on a per-machine basis, destroying your tokens on one machine has no effect on tokens on another machine.

2.11.11. Setting Default Token Lifetimes for Users

The maximum lifetime of a user token is determined by taking the smallest of the following values:

- The "maximum ticket lifetime" recorded in the "*afs*" entry in the Authentication Database. The default is 100 hours. Administrators can inspect this value using **kas examine**, and change it using **kas setfields**.
- The "maximum ticket lifetime" recorded in the issuer's Authentication Database entry. The default is 25 hours for user entries created by the AFS 3.1 or later versions of the Authentication Server, and 100 hours for user entries created by the AFS 3.0 version of the Authentication Server. Administrators and the user himself/herself can inspect this value using **kas examine**, and administrators can change it using **kas setfields**.

- The "maximum ticket lifetime" recorded in the "*krbtgt.**CELLNAME***" entry in the Authentication Database; this entry corresponds to the ticket-granting ticket used internally in generating the token. The default is 720 hours (30 days).

Note: Users who log into AFS using the AFS version of login will always receive a token with the default lifetime calculated above. Users who request tokens with the **klog** can specify a lifetime shorter than the default by using the **-lifetime** argument. For further information, see the *AFS User's Guide* and the entry for **klog** in the *AFS Command Reference Manual*.

As a system administrator, you can reset the any of the default lifetime values listed above using the **kas setfields** command. For further information, see the entry for the **kas setfields** command in the *AFS Command Reference Manual*.

2.11.12. Support for Kerberos Authentication

If your site is using standard Kerberos authentication for your clients, you should use the modified versions of the **klog**, **login**, **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:

- **klog.krb**
- **login.krb**
- **pagsh.krb**
- **tokens.krb**

A cell must use either the Kerberos or the non-Kerberos versions of these commands; you should not mix the two in the same cell. Call Transarc Product Support for instructions on installing the Kerberos version of these four commands. For information on the differences between the two versions of these commands, see the *AFS Command Reference Manual*.

2.12. 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.

2.12.1. Some of AFS's 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 may access the data in the directory, and in what way. See chapter 19 to learn more.

Mutual Authentication Between Client and Server

When an AFS client and server process communicate, each requires the other to prove its identity through a complex process called "mutual authentication". Mutual authentication involves the exchange of encrypted information which only valid parties can decrypt and respond to. Section 1.5 discusses mutual authentication in detail.

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 Server by providing the correct AFS password. If the password is correct, the Authentication Server sends the user a ***token*** as evidence of authenticated status. For most functions, the AFS server processes require that their clients present a token with a request for service. The token contains encryption keys used in the mutual authentication process.

Servers assign the identity *anonymous* to users and processes that do not have a valid token. *Anonymous* has only the access granted to *system:anyuser* 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 section 2.12.2.

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 will not be able to decipher encrypted data without the proper key.

AFS commands that involve server encryption keys and passwords use data encryption. These include

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

In addition, the United States edition of the Update Server encrypts sensitive information (such as the contents of *KeyFile*) when distributing it.

The remaining **bos** commands and the commands in the **fs**, **pts** and **vos** suites do not encrypt data before transmitting it.

2.12.2. Three Types of Privilege

AFS uses three separate types of privilege:

- Membership in the *system:administrators* group. Members are entitled to issue any **pts** command and those **fs** commands that set volume quota. They implicitly have the ADMINISTER right on every ACL in the cell's file system, which means that they can grant themselves any other rights as necessary.
- The `ADMIN` flag set in the Authentication Database. An administrator with this flag can issue any **kas** command.
- Inclusion in */usr/afs/etc/UserList*. An administrator whose user name appears in this file can issue any **bos**, **vos**, or **backup** command (although some **backup** commands require additional privilege; see Chapter 6).

The reasons for using three types of privilege are explained in section 20.2.4. The separation does have the advantage that if an unauthorized user adopts an identity that does not have all three types of administrative privilege, he or she will not automatically be able to perform every privileged action.

2.12.3. 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 file server machines. If authorization checking is enabled on a file server machine, then all of the server processes running on it will fill service requests only for authorized users. If authorization checking is disabled on a file server machine, then all of the server process will perform any action for anyone. Obviously, disabling authorization checking is very dangerous.

See chapter 8 for more information.

2.12.4. General Suggestions for Improving Security in Your Cell

There are several ways you can improve the level of security in your cell.

2.12.4.1. Users

Use the AFS **login** program, or **pagsh** along with **klog**, so that tokens are associated with a PAG rather than a UNIX UID. This prevents users from using someone else's tokens by assuming their UNIX identity. See section 2.11.1.

Encourage users to issue **unlog** to destroy their tokens before logging out. This prevents anyone from attempting to "steal" the tokens and access AFS as that user. One way to make sure users unlog is to include the **unlog** command in their *.logout* files.

2.12.4.2. File Server Machines

Disable authorization checking only in emergencies or for very brief periods of time (for example, when checking a key in the Authentication Database; see section 10.3). 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 phrase including non-alphabetic characters, for instance). Unlike user passwords, the password from which the AFS key is derived can be longer than eight characters, because no one ever types it at a login prompt. The **kas setpassword** command with which you set the key allows you to specify a password hundreds of characters long, if you wish. See chapter 10 for instructions.

As much as possible, limit the number of people who can log in or telnet to a file server machine. Imposing this limit is an extra security precaution rather than a necessity. Your ability to do so depends on your cell's hardware resources, as it implies not running the file server machine as a client also.

Particularly limit those who can become "root" on a file server machine, by keeping its root password secret. As discussed in section 2.7.6, "root" has free access to important administrative directories under */usr/afs*.

As in any computing environment, file server machines must be located in a secured area. Any other security measures are effectively worthless if unauthorized people can access the computer hardware.

2.12.4.3. 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 it is possible that unscrupulous users will install programs that try to steal tokens or passwords. If administrators must use publicly accessible workstations at times, they should be sure to unlog before leaving the machine.

Create "dummy" administrative accounts separate from system administrators' own accounts, and assign AFS privileges only to the dummy accounts. If you still want to be able to trace responsibility, create a different dummy account for each administrator. The administrators should authenticate to the dummy accounts only when performing duties that require privilege, and should unlog as soon as they are finished with administrative tasks.

Administrators should not leave a machine unattended while they have valid tokens. It is simple enough to issue **unlog** before leaving and **klog** upon return.

Set the ticket (token) lifetime for administrative accounts to a fairly short amount of time, using the **-lifetime** argument on **kas setfields**. You must issue this command for each account individually. The default lifetime for the AFS tokens is twenty-five hours. Twenty or thirty minutes might be a reasonable lifetime for administrative tokens. It means that administrators have to issue **klog** more frequently, but if they forget to unlog before leaving their machine, there is less of a chance that an unauthorized user will use the token. The ticket for administrators using the AFS Backup System should last somewhat longer, because it can take almost six hours to fill a 2 gigabyte tape (this is only a rough estimate); the administrator must remain authenticated during that time so that the Backup System running under his or her identity can access the volumes being dumped.

Limit administrators' use of the **telnet** program. It sends unencrypted passwords across the network. Similarly, limit administrators' use of the AFS **rsh** program, which sends tokens across the network without encrypting them.

2.13. AFS's Distributed Database Technology

Section 1.4.4 discussed the benefits of replicating the AFS administrative databases (the Authentication, Protection, Volume Location, and Backup Databases). It also mentioned that having more than one copy of a database creates a need for mechanisms that keep the copies synchronized, given that consistent system performance demands they be identical.

AFS's synchronization mechanism is ***Ubik***, a library of utilities that includes a server portion used by the database server processes and a client portion used by clients who wish to read and change the information in the databases. Ubik incorporates several features that enable it to work without human intervention. Section 2.13.1 describes Ubik's basic structure and the features that enable it to keep database information consistent and available even in the face of outages at the process, machine and network level.

Because Ubik is a subroutine library, it does not appear in listings of the processes running on a file server machine. Ubik's proper operation requires that certain information be available. Section 2.13.2 discusses the need for this information in more detail, and points to the chapters in this manual that provide explicit instructions on how to maintain it.

2.13.1. Ubik's Structure and Features

Several aspects of Ubik's design make it as maintenance-free as possible. They are summarized in the following list and detailed in the subsections below:

1. Ubik's server and client portions both operate automatically.
2. Ubik distributes changes to all copies of a database automatically and almost instantaneously.
3. Ubik has a mechanism for guaranteeing that database copies are always the same.
4. Ubik has a mechanism for keeping the databases available almost all the time.

2.13.1.1. Ubik's Server and Client Portions

As mentioned above, the Ubik library has both a client and a server portion. Each database server process runs a "lightweight" process that knows how to call on the server portion of the Ubik library ("lightweight" is a technical term indicating a process "forked off" from a parent process and incapable of running independently). It is common to refer to this lightweight process itself as Ubik. As a lightweight process, Ubik is hidden within its parent database server process, and does not appear in listings of the processes running on a database server machine.

The clients that need to read and change the databases do not run a lightweight process, but directly call the subroutines in the Ubik library's client portion. The standard clients are the command interpreters invoked by the commands in the AFS command suites. For example, there is a suite of **pts** commands available on client machines that users can utilize to access the Protection Database. The **pts** command interpreter uses Ubik client subroutines to handle the actual communication with the Protection Server.

2.13.1.2. Automatic Updates Distributed from a Synchronization Site

Ubik ensures that every copy of an AFS administrative database reflects changes almost immediately after they are made. Clients can modify only one of the copies of a database, referred to as the ***synchronization site***; the lightweight process in charge of that copy of the database is the ***Ubik coordinator***. The other copies of the database and the lightweight Ubiks in charge of them are called ***secondary***.

There is a separate coordinator for each database server process. The method Ubik uses to elect the coordinator means that the coordinator may end up on a different file server machine for each process. See section 2.13.1.4.

When a client wants to change the information in a database, the client calls a Ubik client subroutine to send the changes to the Ubik coordinator (after determining which machine the coordinator is on). The coordinator registers the change in its own copy of the database (the synchronization site), then distributes the changes to the secondary copies. During the brief distribution period, none of the copies are available to clients, even for reading.

If the coordinator cannot reach a majority of the secondary copies, it aborts the distribution and informs the client that the attempted change failed. (A majority is defined as more than 50% when used for an odd number of sites; when there are an even number of sites, the site with the lowest Internet address has a "weightier" vote that can break ties if necessary.) The requirement that a majority of sites have the newest version of the database helps system recovery after a failure has forced election of a new coordinator. See section 2.13.1.4 for more on coordinator election.

Each change to a database results in an increment to the database version number. The version number makes it easy for the coordinator to determine if a site has the most recent version or not. It is especially useful when communication between coordinator and secondary site is restored after an interruption. The coordinator can determine whether the secondary site still has the latest version of the database, and can send the database with the highest version number if not.

Ubik's update and synchronization scheme is similar to the process the Update Server uses for binaries and common configuration files (see section 1.3.7 for a review). The distribution from synchronization site to secondary copies is both automatic and almost instantaneous. The issuer of a modifying command does not have to know which file server machines have a database copy or which is the synchronization site, nor does the issuer have to wait a long time before the changes are visible in every copy of the database.

2.13.1.3. Guaranteed Consistency Among Database Copies

Ubik's server portion includes a mechanism for guaranteeing that all copies of a database are identical. Replication is useless without this guarantee: there is no point in making information available from more than one source (in this case, a copy of a database) if doing so introduces the possibility that users receive different information depending on which source they contact.

With Ubik, keeping all copies of a database consistent translates into maintaining constant contact between the synchronization site and the secondary sites. The Ubik coordinator (at the synchronization site) frequently sends a time-stamped "guarantee" message to each of the secondary copies. The message effectively says "because you have received this message, you are in contact with the coordinator and your copy of the database is guaranteed to be current until time T" (commonly, T is 60 seconds from the time the coordinator composes the message).

The coordinator sends guarantee messages more frequently than every T seconds, so that the expiration periods overlap. There is no danger of expiration unless network or other problems create a genuine communication failure. If at any time a secondary site's guarantee does expire, it will refuse to provide information to clients until it is again in contact with the coordinator.

For example, if a network outage cuts off one Protection Server process from the Protection Server synchronization site (an outage commonly referred to as a "network partition"), the solitary Protection Server no longer receives updates from the coordinator, so its database copy falls out of date. If the process did not notice the network partition, it would wrongly assume its copy of the database was still up-to-date, and continue to give out information. The information that clients get from this copy of the Protection Database could differ from the information in other copies.

2.13.1.4. A Flexible Coordinator Boosts Availability

The server portion of Ubik also includes a mechanism that makes a database available as much as possible: Ubik can elect a new coordinator if the current one cannot continue in that role. (Using a fixed coordinator might seem simpler, but makes the database less available. Whenever the coordinator fails, none of the secondary copies could give out information either, because they would no longer receive guarantees.)

The coordinator election depends on the exchange of time-stamped messages, like the synchronization of all copies described in the previous section. In response to the coordinator's "guarantee" message, the Ubik at each secondary site returns a "vote" message that tells the coordinator it may continue acting as coordinator until time X. As long as the coordinator continues to get votes from a majority of the secondary sites, the changes to the database are still reaching most of the copies. The coordinator is justified in continuing in that role. (Section 2.13.1.2 explained how Ubik defines a majority. The coordinator is included in calculation of the majority, and votes for itself.)

When the coordinator stops receiving votes from a majority of the secondary sites, it retires and forces election of a new coordinator. (This will 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), no information is available from the database.

Once the new coordinator is elected, it polls all the sites to determine which of them have the database with the highest version number. It adopts that version as the "official" one and distributes it to all the sites that do not have it yet. This scheme works because the update procedure described in section 2.13.1.2 guarantees that a majority of sites always have the latest version of the database.

Ubik's flexible election procedure makes it possible that the Ubik coordinators for the different database server processes might end up on different machines. For example, if the Ubik coordinators for all four processes start out on machine A and the Protection Server on machine A crashes for some reason, then a different site (say machine B) must be elected as the new Protection Database Ubik coordinator. Machine B will remain 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, VL, and Backup Servers, so their coordinators remain on machine A.

2.13.2. What Ubik Needs from the System Administrator

Ubik carries out its duties without any human intervention, but its correct operation relies on certain information. The following sections discuss Ubik's needs in some detail. In summary:

1. Ubik needs synchronized clocks on the file server machines so that all sites agree what time it is (to within 20 seconds) when they send time-stamped messages.
2. Ubik needs a complete list of all database server machines in order to know how many sites constitute a majority.
3. Ubik expects all database server processes to run at every site.

2.13.2.1. Ubik Needs Synchronized Clocks

Ubik needs synchronized clocks so that all sites agree what time it is when they send time-stamped messages. You can prevent needless outages by keeping the clocks at Ubik sites in sync.

The Reason

Time-stamped messages are the means through which Ubik sites keep in contact. Clock differences among Ubik sites can cause them to believe they are no longer in contact, even though they are all in fact operating and their network connections are open. The reactions to falling out of touch—trying to elect a new coordinator and/or refusing to give out information—cause service outages noticeable to users, as one or more of the copies of the database become inaccessible for a time. Such reactions are appropriate if the communication failure really is due to failure (process, machine or network), but Ubik cannot distinguish between many types of failures and the symptoms caused by clock differences.

There are a several ways that skewed clocks can cause service outages, depending on which clock is ahead of the other(s).

For example, the problem may be that the Ubik coordinator for the Protection Server believes that it is later than the secondary sites believe. The coordinator might think it is 9:35:30 while the secondary

Protection Server sites think it is 9:31:30. The secondary sites send the coordinator votes that say in effect, "I acknowledge you as coordinator until 9:33:30." This is 120 seconds in the future from the perspective of the secondary copies, 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 and makes the database unavailable at every site during that time.

The opposite possibility is that a secondary site believes it is later (say 14:50:00) than the coordinator does (say 14:46:30). When the coordinator sends a guarantee (good until 14:47:30) to the secondary site, it appears to the secondary site to be expired. Believing that it is out of contact with the coordinator, the secondary site reacts in two ways:

1. It refuses to give out any data to clients, which would be appropriate if a network partition really had separated it from the coordinator. The problem is that it actually is in contact with the coordinator, so its data actually is current.
2. It stops sending votes to the coordinator and tries to collect enough votes to get itself elected as coordinator. This would be appropriate if the coordinator had actually failed, since none of the sites can give out data until a new coordinator is in charge of synchronizing. However, it is inappropriate in this case because the current coordinator is still operative.

Both of these reactions cause the secondary site to be inaccessible. A user who contacts this site (and clients such as the command interpreters generally choose a site at random for reading) experiences a long delay before getting a response, because the command interpreter waits until a request at one Ubik site "times out" before forwarding the request to another.

The attempt of a single secondary site to get elected as the new coordinator usually does not affect the performance of the other sites. They ignore its request for votes and follow their built-in predisposition to continue voting for the current coordinator. However, if enough of the secondary sites' clocks get ahead of the coordinator's, they can force election of a new coordinator even though the current one is actually working fine. An election causes a service interruption of about three minutes.

The service interruptions discussed above are desirable if due to actual process failures or network partitions. If a secondary site really is cut off from the coordinator, it should not give out information, just as a new coordinator should be elected if the current one really is not in contact with a majority of the sites. The behaviors are only undesirable in the cases where clock difference is their only cause.

What you need to do

To keep the internal clocks of all your AFS file server machines synchronized, you run the ***Network Time Protocol Daemon (NTPD)*** on each one. As with many things in AFS (including Ubik), the NTPD on one machine serves as the "time synchronization site," broadcasting the correct time to the NTPDs on the other machines (it gets the correct time itself from some reliable source outside the cell). The instructions in the *AFS Installation Guide* tell you how to set up the NTPD on each machine you install.

2.13.2.2. Ubik Needs To Know About All the Database Server Machines

The Ubik sites all need to know about their peers so that for election purposes they know what number constitute a majority. This information is available in a file on each database server machine's local disk (indeed, on the disk of every AFS machine).

The Reason

Each Ubik site must be aware of the other sites. This need is especially crucial during election of a new coordinator, when all the sites need to know how many sites constitute a majority, and where they should send their votes. During normal operation, the Ubik coordinator needs the information to determine if it is receiving votes from a majority of sites so it may remain coordinator. It also needs to know the secondary sites' addresses so it knows where to send updates and its guarantee messages.

Actually, all file server machines need this information. Non-database server machines need to know where to get information from the databases. For instance, the File Server frequently needs to contact the Protection Server to determine if a user's identity and group memberships entitle him or her to a requested operation.

What you need to do

You need to ensure that the names and addresses of your cell's database server machines are always correct in the file */usr/afs/etc/CellServDB* on the local disk of every file server machine. All of them update their copies from the one on the system control machine so all have the same information. The *AFS Installation Guide* includes directions on how to create this file on the system control machine and distribute it to all the other machines as you add them.

Alter the file only when you add or decommission a database server machine; you must alter it at that time. Make changes only to the *CellServDB* on the system control machine and only by using the appropriate commands. The instructions in chapter 4 for stopping and starting processes remind you to alter *CellServDB* in the proper circumstances, as do the instructions in *AFS System Administrator's Guide* for starting up a new database server machine. Chapter 9 also specifies the commands you should use to change the file.

2.13.2.3. Ubik Expects All the Database Server Processes to Run Together

You need to run all four database server processes—Authentication Server, Protection Server, VL Server, and Backup Server—on all the database server machines. For example, it is not possible to run the VL Server on one extra machine without running the other three processes there too. Both the client and server portions of Ubik expect that all the database server machines listed in *CellServDB* are running all of the database server processes. There is no mechanism for indicating which particular processes are running on a certain machine.

2.14. Backing Up the System

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

2.14.1. Backup Volumes

The first facility is the Backup volume, which you create by cloning a ReadWrite volume. The Backup volume is read-only and so preserves the state of the ReadWrite 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. You should create a Backup version of each user volume and mount it as a subdirectory of the user's home directory. A common name for this mount point is *OldFiles*. Create a new version of the Backup volume (that is, reclone the ReadWrite) once a day to capture any changes that were made since the previous backup. If a user accidentally removes or changes data, he or she can restore it from the Backup volume, rather than having to ask you to restore it.

The *AFS User's Guide* does not mention Backup volumes, so users will not expect to find 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 may event 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.

Section 5.6 discusses Backup volumes in detail.

2.14.2. 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. See chapter 6 for detailed information about the AFS Backup System.

2.15. Using Standard Remote Services in the AFS Environment

Transarc distributes modified versions of several standard UNIX commands, daemons and programs that provide remote services. They include:

ftpd
inetd
login
rcp
rlogind
rsh

These modifications enable the commands to handle AFS authentication information (tokens). This allows 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. You may well find 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 **ftpd** and **rcp**.

Of all the commands, the AFS version of **login** is most likely to increase the usability of AFS in your cell. It authenticates users with AFS and logs them into the local UNIX file system in one step. If you do not use AFS **login**, your users will have to issue additional AFS authentication commands after logging in. However, AFS **login** does not include all of the features found in some proprietary versions of **login**. See section 2.11.2 for more information.

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 **rcp** only if you are using the AFS version of both **rcp** and **inetd**.

To learn more about the added functionality provided by the AFS versions of these commands, see the chapter on **Modified** UNIX **Commands** in the *AFS Command Reference Manual*. Restrictions on and requirements for their use also appear in that chapter.

The AFS distribution tape includes these modified remote service commands in the */usr/afsws/bin* and */usr/afsws/etc* directories.

2.16. Accessing AFS through NFS

Users of NFS client machines can access AFS files 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 with unsupported system types.

The NFS/AFS Translator is available from Transarc at an additional charge. Installation and maintenance documentation is included. Contact your AFS Product Support representative or call the number given on page xxviii of the introductory *About This Manual* chapter.

Managing File Server Machines

3. File Server Machine Processes and Files

This chapter lists the processes that run on your cell's file server and database server machines. It also describes the binary and configuration files that each server machine must have in various subdirectories of */usr/afs* on its local disk. See chapter 1 for more detailed descriptions of the various types of file server machines and the processes that run on them.

This chapter includes the following sections:

3.1 Brief Descriptions of the AFS Server Processes 3-2
3.2 Binaries in */usr/afs/bin* 3-10
3.3 Common Configuration Files in */usr/afs/etc* 3-12
3.4 Local Configuration Files in */usr/afs/local* 3-14
3.5 Replicated Database Files in */usr/afs/db* 3-15
3.6 Log Files in */usr/afs/logs* 3-16
3.7 Volume Headers on the */vicep**x*** Partitions 3-17

Note: This chapter describes only the files and processes on a file server or database server machine; it does not detail how to create them. See the *AFS Installation Guide* for information about configuring file servers and database servers during cell installation.

3.1. Brief Descriptions of the AFS Server Processes

This section briefly describes the different server processes that can run on a file server machine. In cells with multiple file server machines, not all processes 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 it is given when defined with **bos create** on that machine. Therefore, the name can differ from machine to machine. However, you are urged to use the same name for a process on all machines and to follow the naming conventions suggested in the *AFS Installation Guide* and the **bos create** command reference page. Examples of process names 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 this documentation, a process is referred to as "the X Server." Examples of process references are the *Basic OverSeer (BOS) Server*, the *Authentication Server*, and the *Volume Location Server*.

The following sections specify each name for the process as well as some of the administrative tasks in which you will use the process. For a more general description of the servers, see section 1.3.

3.1.1. 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 bound in a single process because they all operate on the same "database"—the files and directories in the file system and the volumes containing them. Binding together several component processes makes the *fs* process unusual, but it allows the components to easily coordinate their actions affecting files and volumes, so no inconsistencies are created.

Individually, the File Server component handles requests at the file and directory level, including all standard UNIX file manipulation commands. Its main duty is to deliver requested files to clients and store them again when the client is finished. It also maintains status and protection information about each file and directory. It runs continuously during normal operation. See section 1.3.1 for more complete information.

The Volume Server component operates at the level of whole volumes, in contract to the file/directory level. In response to a system administrator's commands, it creates, removes, moves, dumps and restores entire volumes, among other actions. It runs continuously during normal operation. See section 1.3.5 for more information.

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. See section 1.3.9.

3.1.1.1. Names

The process appears as *fs* in output from **bos status**, assuming you name it as suggested in the *AFS Installation Guide*. The three processes do not appear individually, although an auxiliary message may report the status of the File Server or Salvager component. See section 4.3 for more information on how this process appears in output from **bos status**.

The *fs* process does not appear in output from **ps**. Instead, its three components appear separately as:

- `/usr/afs/bin/fileserver`
- `/usr/afs/bin/volserver`
- `/usr/afs/bin/salvager`

The process does not have an everyday name other than *fs*. Its three components do have everyday names—File Server, Volume Server, and Salvager—because they operate independently.

3.1.1.2. When and How to Contact the Three Components

You contact the File Server component for several reasons, often by issuing **fs** commands:

- When accessing data or status information from any file in the file system (in this case your Cache Manager contacts the File Server on your behalf)
- When performing UNIX file manipulation commands such as **cp** and **ls** (in this case your Cache Manager contacts the File Server on your behalf)
- When administering the ACL of any directory in the file system. See chapter 19 to learn about ACLs.
- When installing new partitions. You must restart the *fs* process on the machine using **bos restart** after installing a new *vicep***x** partition. Restarting the File Server makes it aware of the new partition.
- When creating and deleting AFS mount points in order to make volumes visible in the file tree. See section 5.7 to learn about mount points.
- When setting volume quota and checking on volume and partition-related information (space used and availability of space). See section 5.12 to learn about volume quota.

Contact the Volume Server component by means of **vos** commands when manipulating volumes in any way—creating, removing, replicating, moving, renaming, converting to different formats, and salvaging. See chapter 5.

The Salvager normally runs automatically in case of a failure. You may invoke it directly with the **bos salvage** command; see section 5.11.

3.1.2. The *bosserver* Process: the Basic OverSeer Server

The *bosserver* process, which runs on every file server machine, is the Basic OverSeer (BOS) Server responsible for monitoring the other processes running on its file server machine. If a process dies, the BOS Server can restart it automatically, without intervention from a human operator. It takes interdependencies into account when restarting a process that has multiple component processes (such as the *fs* process described in section 3.1.1).

For a more complete description of the BOS Server, see section 1.3.2.

3.1.2.1. Names

Because the BOS Server cannot monitor and restart itself, it is not included in the list that appears from the **bos status** command.

It appears in the output from the **ps** command as `/usr/afs/bin/bosserver`.

Its everyday name is the Basic OverSeer (BOS) Server.

3.1.2.2. When and How You Contact the BOS Server

As a system administrator, you contact the BOS Server for several reasons, in each case by issuing **bos** commands:

- When defining which processes the BOS Server should monitor (by placing them in the file */usr/afs/local/BosConfig*). For more on *BosConfig*, see section 4.2.
- When stopping and starting processes on the file server machines. See Chapter 4.
- When defining your cell's database server machines in the file */usr/afs/etc/CellServDB*. See chapter 9.
- When changing the AFS server encryption keys that servers use to decrypt server tickets, by changing the file */usr/afs/etc/KeyFile*. See chapter 10.
- When granting system administrator privileges with respect to BOS Server, Volume Server, and Backup Server operations, by adding a user to the file */usr/afs/etc/UserList*. See section 20.5.
- When controlling whether or not the processes on a file server machine perform authorization checking. See chapter 8.

3.1.3. 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 user identity at login by requiring a password. It maintains all AFS server encryption keys and user passwords in the Authentication Database in the form of encryption keys. The Authentication Server's Ticket Granting Service (TGS) module creates the *tickets* that clients and servers use to establish secure connections.

See section 1.3.3 for a more complete description of the Authentication Server.

3.1.3.1. Names

The process appears as *kaserver* in output from **bos status** (assuming it is named as suggested in the *AFS Installation Guide*). The "*k*" reflects the fact that AFS's authentication protocols are based on Kerberos, originally developed by the Massachusetts Institute of Technology's Project Athena.

It appears in the output from the **ps** command as `/usr/afs/bin/kaserver`.

Its everyday name is the Authentication Server.

3.1.3.2. When and How to Contact the Authentication Server

As a system administrator, you contact the Authentication Server for several reasons, by issuing **kas** commands:

- When setting a user's initial password or changing the password at a later time. Because users normally change their own passwords, you may never need to change a user's password. See sections 17.5 and 17.7, respectively for more details.
- When setting the AFS server encryption key in the Authentication Database, which the TGS uses to seal server tickets. See chapter 10.
- When granting or revoking system administrator privileges with respect to the Authentication Server. See section 20.4.

3.1.4. 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 entries for every user and protection group in the cell. The File Server process consults the Protection Server for help in verifying if a user's identity or group memberships authorize the user to perform a requested action. The Protection Server also allocates AFS UIDs for new users/groups, and translates between them and user names for the benefit of other servers.

See section 1.3.4 for a more complete description of the Protection Server.

3.1.4.1. Names

The process appears as *ptserver* in output from **bos status**, assuming it is named as suggested in the *AFS Installation Guide*.

It appears in the output from the **ps** command as `/usr/afs/bin/ptserver`.

Its everyday name is the Protection Server.

3.1.4.2. When to Contact the Protection Server

As a system administrator, you contact the Protection Server for several reasons, by issuing **pts** commands:

- When creating a new user or group entry in the Protection Database. This includes allocating and recording the AFS UID. See sections 18.5 and 18.7.
- When manipulating user or group entries in the Protection Database for any reason (adding or removing members of a group or entire entries). See chapter 18.
- When granting or revoking system administrator privilege by changing the membership of the *system:administrators* group. See section 20.3.

3.1.5. The *vlserver* Process: the Volume Location Server

The *vlserver* process, which runs on database server machines, is the Volume Location (VL) Server that tracks the location of all volumes in the file system. Its automatic tracking of volume locations is what makes file access transparent. You are free to move volumes in order to balance the system workload without worrying that access transparency will suffer.

See section 1.3.6 for more information about the VL Server. For information about the Volume Server, remember that it runs as part of the combined *fs* process. See section 3.1.1.

3.1.5.1. Names

The process appears as *vlserver* in output from **bos status**, assuming you name it as suggested in the *AFS Installation Guide*.

It appears in the output from the **ps** command as `/usr/afs/bin/vlserver`.

Its everyday name is the VL Server.

3.1.5.2. When and How to Contact the Volume Location Server

As a system administrator, you contact the VL Server by issuing any **vos** command that changes the status of a volume. The VL server records status changes in the VLDB. There are some **vos** commands that allow you to contact the VL Server independent of the Volume Server; see chapter 5 for information about these commands.

3.1.6. The *upserver* and *upclient* Processes: the Update Server

The Update Server has two separate parts, each of which runs on a different type of file server machine. The *upserver* process is the server portion of the Update Server:

- In cells using the United States edition of AFS, it runs on the cell's system control machine in order to distribute the contents of that machine's */usr/afs/etc* directory to all the other file server machines. This guarantees that all machines have the same version of the common configuration files in that directory (see section 3.3 for more on the files themselves).
- It runs on a binary distribution machine of each CPU/operating system the cell uses as a file server machine in order to distribute the contents of each one's */usr/afs/bin* directory to the other file server machines of that type. This guarantees that all machines have the same version of AFS binaries (see section 3.2 for more details on the binaries themselves).

The *upclient* process is the client portion of the Update Server:

- In cells using the United States edition of AFS, one instance runs on every file server machine except the system control machine. It references the system control machine as the source for updates to the common configuration files in */usr/afs/etc*. It is commonly named *upclientetc*.
- One instance runs on every file server machine that is not the binary distribution machine for its CPU/operating system type. It references the appropriate binary distribution machine as the source for updates of the binaries in */usr/afs/bin*. It is commonly named *upclientbin*.

See section 1.3.7 for more information on the Update Server.

3.1.6.1. Names

In output from **bos status**, the server portion appears as *upserver* and the client portion(s) as *upclientbin* and *upclientetc*, assuming you named them as suggested in the *AFS Installation Guide*.

In the output from the **ps** command, the server portion appears as `/usr/afs/bin/upserver`, while the client portions appear as `/usr/afs/bin/upclient`.

In everyday use, these processes are referred to as the server and client portions of the Update Server.

3.1.6.2. When and How to Contact the Update Server

As a system administrator, you do not contact the Update Server directly once you have installed it. It is invoked automatically whenever you use the appropriate **bos** commands to change the files distributed by the Update Server. Sections 3.2 and 3.3 discuss the files and the associated commands.

3.1.7. The *buserver* Process: the Backup Server

The *buserver* process, which runs on database server machines, is the Backup Server that maintains information about the Backup System. It contacts the Backup Database any time Backup System configuration information is modified or data is dumped from or restored to the file system.

See section 1.3.8 for more information about the Backup Server.

3.1.7.1. Names

The process appears as *buserver* in output from **bos status**, assuming you name it as suggested in the *AFS Installation Guide*.

It appears in the output from the **ps** command as `/usr/afs/bin/buserver`.

Its everyday name is the Backup Server.

3.1.7.2. When and How to Contact the Backup Server

As a system administrator, you contact the Backup Server by issuing any **backup** command that manipulates information in the Backup Database. This includes all **backup** commands that change Backup System configuration information (such as the sets of volumes to be dumped and the schedule by which they are dumped) or that dump data from the file system to tape. You also contact the Backup Server when information is required from the Backup Database to restore data from tape to the file system. The Backup Server communicates all necessary Backup System information to the Backup Database. See Chapter 6 for information about the Backup System and **backup** commands.

3.1.8. The *runntp* Process: the Network Time Protocol Daemon

The *runntp* process, which runs on every file server machine, is a controller program for the Network Time Protocol Daemon (NTPD), which keeps the internal clocks on the file server machines synchronized. Database server machines must agree on the time (within a narrow range of variation) in order for AFS's distributed database management technology (called Ubik) to work properly. Section 2.13.2.1 explains why Ubik needs synchronized clocks.

Non-database server machines need the correct time because they set modification time stamps on files according to their local clock. Also, they distribute the time to the Cache Manager on client machines.

3.1.8.1. Names

The process appears as *runntp* in output from **bos status**, assuming you named it as suggested in the *AFS Installation Guide*.

It appears in the output from the **ps** command as `/usr/afs/bin/runntp`. When issued on the system control machine, the line should include arguments appropriate in your cell; see the **runntp** entry in the *AFS Command Reference Manual*. When issued on any other file server machine in the cell, there should be no arguments. Another process line starting with "*ntpd*" will also appear in the output from **ps**. Its exact form depends on how you run *runntp*.

Its everyday name is the Network Time Protocol Daemon (NTPD).

3.1.8.2. When and How to Contact the NTPD

As a system administrator, you do not contact the NTPD directly once you have installed it. See the *AFS Installation Guide* for instructions on installing it.

3.2. Binaries in */usr/afs/bin*

The directory */usr/afs/bin* on every file server machine's local disk contains AFS binaries appropriate for the machine's CPU/operating system type. The binaries are for both AFS server processes and command suites. 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 portion or component resides in a separate file.

To ensure predictable system performance, all file server machines should run the same version of a process (in the binary form appropriate for their CPU type). AFS ensures consistency by using the Update Server to distribute binaries from a "binary distribution machine" of each CPU/operating system type. Following are some rules to consider:

- Verify that the new binaries are available for every CPU/operating system type you use as a file server machine. You should compile or get from Transarc binaries for each system type you use.
- Install new binaries on the binary distribution machine for each CPU/operating system type for distribution to the other machines of that type. Chapter 11 outlines the proper installation procedures.
- Each binary distribution machine must run a server portion of the Update Server and each non-distribution machine must run a client portion of the Update Server that references the appropriate distribution machine's */usr/afs/bin* directory. The *AFS Installation Guide* explains how to install these processes.

You should not remove any process binaries from */usr/afs/bin*, even though not every process runs on every machine (the Protection Server, *ptserver*, for example, runs only on database server machines). It is easier to reconfigure your cell if you have all process binaries on every machine. For example, you may wish to convert a simple file server machine into a database server machine, and the binary for *ptserver* will already be available.

Similarly, do not remove command suite binaries from */usr/afs/bin*, even though you invoke them only rarely (you generally issue AFS commands from client machines rather than file server machines). Having copies of the command binaries on the local disk allows you to issue commands even if the file system is down.

The following list includes binary files found in */usr/afs/bin*. Note that it lists only files directly related to AFS server processes or command suites. Other files, particularly binaries for administrative programs (for example, **klog**), may also appear in this directory on a particular file server machine's disk or in a distribution from Transarc.

backup The set of commands used to communicate with the Backup Server; the binary for the Backup Server is *buserver*.

bos The set of commands used to communicate with the Basic OverSeer (BOS) Server; the binary for the BOS Server is *bosserver*.

bosserver The binary for the Basic OverSeer (BOS) Server. See sections 1.3.2 and 3.1.2.

buserver	The binary for the Backup Server. See sections 1.3.8 and 3.1.7.
fileserver	The binary for the File Server component of the *fs* process. See sections 1.3.1 and 3.1.1.
kas	The set of commands used to communicate with the Authentication Server; the binary for the Authentication Server is *kaserver*.
kaserver	The binary for the Authentication Server. See sections 1.3.3 and 3.1.3.
ntpd	The binary for the Network Time Protocol Daemon. See section 3.1.8.
ntpdc	A debugging utility furnished with *ntpd*.
pts	The set of commands used to communicate with the Protection Server; the binary for the Protection Server is *ptserver*.
ptserver	The binary for the Protection Server. See sections 1.3.4 and 3.1.4.
runntp	The binary for a "front end" program that Transarc provides along with *ntpd*. This program makes it easier to configure NTPD most appropriately for use with AFS. See the *AFS Installation Guide* for instructions on how to use this program when installing a file server machine.
salvager	The binary for the Salvager component of the *fs* process. See section 3.1.1.
udebug	The binary for a debugging program that allows you to track the operation of AFS's distributed database technology, which is called Ubik. See section 2.13.
upclient	The binary for the client portion of the Update Server. See sections 1.3.7 and 3.1.6.
upserver	The binary for the server portion of the Update Server. See sections 1.3.7 and 3.1.6.
vlserver	The binary for the Volume Location (VL) Server. See sections 1.3.6 and 3.1.5.
volserver	The binary for the Volume Server component of the *fs* process. See sections 1.3.5 and 3.1.1.
vos	The set of commands used to communicate with the Volume and Volume Location Servers; the binaries for the servers are *volserver* and *vlserver*, respectively.

3.3. Common Configuration Files in */usr/afs/etc*

The directory */usr/afs/etc* on every file server machine's local disk contains text and machine-independent binary files which the machine needs to run properly. To ensure predictable system performance, all file server machines in a cell must have the same version of each configuration file. In cells using the United edition of AFS, the recommended way to assure consistency is to run a client portion of the Update Server on each machine to update the local copy with the copy kept on the cell's system control machine (which runs the server portion of the Update Server).

Cells using the international edition of AFS must not use the Update Server to distribute the files in */usr/afs/etc*: data encryption routines are not available to the Update Server in this edition of AFS, and the contents of the files are too sensitive to cross the network unencrypted. Administrators in cells using the international edition of AFS must update the files in */usr/afs/etc* on each file server and database server machine individually, taking extra care to issue exactly the same **bos** command for each machine. 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 file server machines.

There are several implications for the administrator:

- Never edit any of the files in */usr/afs/etc* directly, except as directed by instructions for dealing with emergencies.
- Use the appropriate **bos** commands to change the files. The file descriptions below direct you to the appropriate commands.
- If using the United States edition of AFS, change the files only on the system control machine, except as directed by instructions for dealing with emergencies. If using the international edition of AFS, change the files on each machine individually as directed.
- If using the United States edition of AFS, run a server portion of the Update Server on the system control machine, and a client portion of the Update Server on every other file server machine. The client portion should reference the system control machine's */usr/afs/etc* directory. Do not run the client portion in cells using the international edition of AFS. The *AFS Installation Guide* includes instructions appropriate for the two editions.

The files in this directory include:

CellServDB This ASCII file names the cell's database server machines, which are selected to run the Authentication, Protection, Volume Location, and Backup Servers. You create the initial version of this file with the **bos setcellname** command during the installation of the cell's first file server machine. That command defines the cell name on the first line of the file and creates an entry designating the machine as the cell's first database server machine. It is very important to update this file when you change the identity of your cell's database server machines.

This *CellServDB* is not the same as the *CellServDB* that client machines keep in their */usr/vice/etc* directory. The client version lists the database server machines for every AFS cell that the system administrator chose to make visible from your cell. The server version of *CellServDB* (this one) lists only the local cell's database server machines, because server processes never contact processes in other cells.

Chapter 9 describes how to maintain this file with **bos** commands.

KeyFile This machine-independent, binary-format file lists the server encryption keys that AFS server processes need in order to decrypt tickets. The information in this file is extremely sensitive, as encryption keys and the tickets they seal are the basis of secure communication in your cell. Section 1.5 explains further the role of server encryption keys in security.

This file is specially protected so that only privileged users may read or change it. Chapter 10 describes the correct procedures.

ThisCell This ASCII file contains a single line defining the complete Internet domain-style name of the cell (such as `transarc.com`). As described in the *AFS Installation Guide*, you create this file with the **bos setcellname** command during the installation of your cell's first file server machine.

Do not change your cell's name at any point, because that would require rebuilding the entire Authentication Database: the Authentication Server combines the cell name it finds in this file with each password and converts the combination into encryption key form for storage in the Database. The login program makes the same conversion when users login. If the contents of this file change, all the keys in the Authentication Database must also change.

UserList This file contains the user names of the system administrators authorized to issue privileged **bos**, **vos**, and **backup** commands. Section 20.5 explains how to maintain this file with **bos** commands.

3.4. Local Configuration Files in */usr/afs/local*

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*. The most important file is *BosConfig*; it defines which server processes are to run on that machine.

As with the common configuration files in */usr/afs/etc*, **never edit these files directly**. Some files should not be altered at all; change others only with the appropriate **bos** commands.

The files in this directory include:

BosConfig This file lists the processes that should run on the file server machine. Use it to define which processes the BOS Server monitors and what to do with a process when it fails. It also defines the times at which the BOS Server automatically performs some maintenance tasks.

As you create server processes during a file server machine's installation, their entries are defined in this file automatically. The *AFS Installation Guide* outlines the **bos** commands to use. Chapter 4 describes the information in the file more completely, and explains how to control process status by making changes to it with **bos** commands.

NoAuth This 0-length file is seldom present because it instructs all the server processes running on the machine not to perform authorization checking. Thus, they perform any action for any user, even *anonymous*. This very unsafe 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 **bos setauth** to turn on authentication, the BOS Server removes this file. See Chapter 8 for more information.

SALVAGE.fs The presence of this 0-length file controls what the BOS Server does after 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 force a salvage if desired using **bos salvage**; see section 5.11.

salvage.lock This file guarantees that only one Salvager will run at a time. As the Salvager initiates (when invoked by the BOS Server or by issue of **bos salvage**), it creates this 0-length file and f-locks it. A Salvager can run only when it has this file locked, so only one Salvager can run when this file is present.

3.5. Replicated Database Files in */usr/afs/db*

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

- A file for each database, with a *.DB0* extension. This file contains the database itself.
- 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, Protection, VL, or Backup Server) maintains its own database and log files. The database files are not in human-readable form, so you must always access or alter them using commands from the **kas** suite (for the Authentication Database), **pts** suite (for the Protection Database), **vos** suite (for the VLDB), or **backup** suite (for the Backup Database).

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 section 2.13.

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

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.
bdb.DB0	The Backup Database file.
bdb.DBSYS1	The Backup Database log file.

3.6. Log Files in */usr/afs/logs*

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 *VolserLog*. Events are recorded at completion, so the server processes do not use these files to reconstruct failed operations (these log files have a different function than those in */usr/afs/db*).

The information in log files can help you a great deal in evaluating process failures and other problems. For instance, if you receive a timeout message when you try to access a volume, checking the *FileLog* might provide an explanation, showing that the File Server was unable to attach the volume. You should consult the logs as a first step in troubleshooting. The messages found there should be self-explanatory. The **bos getlog** command can be used to examine the log files on a file server machine; see section 4.10 in chapter 4.

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 instance, it labels a core file generated by an aborting Protection Server as *core.ptserver*. However, if two processes abort at about the same time, then the BOS Server may not be able to assign the correct extension. You should treat the process name extension as a suggestion, not as a guarantee that the core file did indeed come from the named process.

The directory contains the following files:

AuthLog	A log file generated by the Authentication Server.
BackupLog	A log file generated by the Backup Server.
BosLog	A log file generated by the BOS Server.
FileLog	A log file generated by the File Server.
SalvageLog	A log file generated by the Salvager.
VolserLog	A log file generated by the Volume Server.
*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: If you do not periodically shutdown and restart the database server processes, the corresponding log files may grow unmanageably large. To prevent this, you may periodically delete the contents of the log files using the UNIX **rm** command.

3.7. Volume Headers on the */vicepx* Partitions

Each file server machine partition that houses AFS volumes is mounted at a directory whose name has the form

*/vicep**x***

where ***x*** is a letter. By convention, the first AFS partition on a machine is mounted at */vicepa*, the second at */vicepb*, and so on.

In order for the File Server and Volume Server to access the volumes kept on such partitions, the corresponding directory must be a subdirectory of the machine's root (/) directory, not (for instance) of */usr*. Also, the directory must be correctly mapped to the partition's device name in the file server machine's file system registry file (*/etc/fstab* or equivalent).

It is possible, but not recommended, to store non-AFS files on these AFS partitions.

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

*V**volumeID number**.vol*	Each such file is a volume header. The ***volumeID number*** corresponds to the number shown by the **vos listvol** command.
FORCESALVAGE	The existence of this 0-length file tells the Salvager that it should salvage the entire partition. The **vfsck** program (a version of **fsck** modified for use with AFS) creates this file if it discovers corruption. Incidentally, you should **NEVER run the standard "fsck" program on an AFS file server machine**. It will remove files from directories that are not part of the UNIX file system, including the */vicep**x*** directories.

4. Monitoring and Controlling Server Processes

One of your most important responsibilities as a system administrator is making sure 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 (see section 4.2 to learn how).

You may sometimes need to take direct control of server process status to perform routine maintenance or to correct 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.

This chapter explains how to define the processes that run on a file server machine and how to start and stop them directly. It has the following sections:

4.1 Summary of Commands 4-1
4.2 Controlling and Checking Process Status 4-3
4.3 Listing Process Status and Information from *BosConfig* 4-7
4.4 Listing Which Machines Are Taking Special Roles 4-11
4.5 Creating and Removing Processes 4-14
4.6 Stopping and Starting Processes "Permanently" 4-19
4.7 Stopping and Starting Processes Temporarily 4-21
4.8 Stopping and Immediately Restarting Processes 4-23
4.9 Setting the BOS Server's Restart Times 4-25
4.10 Examining a File Server Machine's Log Files 4-28
4.11 Rebooting a File Server Machine 4-30

4.1. Summary of Commands

Examine process status . . . . **bos status**
Examine information from *BosConfig* . . . . **bos status -long**
Create a process instance . . . . **bos create**
Stop a process "permanently" . . . . **bos stop**
Start a "permanently" 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**

4.2. Controlling and Checking Process Status

You define which processes the BOS Server monitors by creating process entries in the */usr/afs/local/BosConfig* file on each file server machine's local disk. You should never need to run an AFS Server process without including it in *BosConfig* so that the BOS Server monitors it. The information in a process's entry defines how the process should run, so you control process status (running or not running) by changing the entry with the appropriate **bos** commands.

Note: Never edit the *BosConfig* file directly. Use only the commands described in this chapter.

4.2.1. The Information in *BosConfig*

A process's entry in *BosConfig* includes the following information:

- The process's ***name***. The instructions in both section 4.5 and the *AFS Installation Guide* recommend standard process names. For example, the name of a simple process usually matches the name of the binary file executed to run it (for example, *ptserver* for the Protection Server).
- Its ***type***. There are three types: ***simple***, ***fs***, and ***cron***. See section 4.2.2 for an explanation.
- Its ***status flag***, which tells the BOS Server

  – Whether it should start up the process as it (the BOS Server) initializes.

  – Whether it should restart the process if it (the process) fails.

  If the status flag is set to `Run`, the process should be running whenever possible—the BOS Server will start it initially and attempt to restart it automatically if it fails. If the flag is set to `NotRun`, the BOS Server will not start it initially and will not attempt to restart it. Only a system administrator, not the BOS Server, can change this flag. (This flag is for internal use only; it does not appear in the output from **bos status**.)

  You can also control process status independently of this flag. To learn how, see section 4.2.3.
- its ***command parameters***—the commands the BOS Server issues to run the process. The number of parameters depends on the type of the process: "simple" processes have one, the "fs" process three, and "cron" processes two; see section 4.2.2 for definitions of process types.

BosConfig also records the two times when the BOS Server automatically restarts itself and/or other processes. See section 4.9 for more information.

4.2.2. The Types of Processes

AFS server processes fall into three categories: simple processes, the *fs* process, and cron processes.

4.2.2.1. Simple processes

A ***simple*** process runs independently of any other on the file server machine. If several simple processes fail at the same time, the BOS Server does not have to coordinate their restarts. All standard AFS processes except the *fs* process are simple.

A simple process has a single command parameter—the pathname to its binary file. An entry for a simple process in *BosConfig* includes information similar to the following. This is an example for the Protection Server, but it does not imply anything about the format of an actual entry in *BosConfig*.

- `ptserver` as the name
- `simple` as the type
- `Run` as the status flag
- `/usr/afs/bin/ptserver` as the command parameter

4.2.2.2. The fs Process

The ***fs*** process type is reserved for a single AFS process whose standard name is also "*fs*." This process combines three components: the File Server, the Volume Server, and the Salvager. They are combined into a single process because

- They all operate on the same data, namely the files and directories stored in the AFS file tree. When combined, they can coordinate their operations on the data. They never attempt simultaneous operations on the same data, which would cause corruption.
- The BOS Server stops and restarts the processes in a particular 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 during this special circumstance or when you invoke it yourself using **bos salvage**; see section 5.11.) If only the Volume Server fails, it restarts as though it were simple. Order is not important when restarting the Volume Server and File Server at the same time.

The *fs* process has three command parameters, one for each component. Its entry in *BosConfig* contains information similar to the following (again, this list does not imply anything about the internal format of the file):

- `fs` as the name
- `fs` as the type
- `Run` as the status flag
- three command parameters:
  - `/usr/afs/bin/fileserver`
  - `/usr/afs/bin/volserver`

– /usr/afs/bin/salvager

4.2.2.3. Cron Processes

A ***cron*** process is like a simple process in that it involves only one process running independently of any other. The difference is that the BOS Server executes *cron* processes periodically, not continuously.

A cron process has two command parameters. The first parameter is the command to be executed, and the second parameter is the time at which the BOS Server is to execute it. There are no standard AFS cron processes, but an example of the type of information you need to specify in *BosConfig* follows:

- `cron-proc` as the name
- `cron` as the type
- `Run` as the status flag
- two command parameters:
  - `/usr/afs/bin/cron-proc`
  - `wed 16:00`

4.2.3. How the BOS Server Uses the Information in *BosConfig*

Whenever the BOS Server starts or restarts, it reads *BosConfig* to learn which processes it is to start and monitor. It transfers the information into memory and does not read *BosConfig* again until it next restarts.

Because of the transfer into memory, the BOS Server's memory state can change independently of *BosConfig*. Therefore, it is possible for a process to stop running even though its status flag in *BosConfig* is `Run`. This happens when you (or the BOS Server) change the process's status in the BOS Server's memory but not in *BosConfig*. Similarly, you can start a process running without setting its *BosConfig* status flag to `Run` by setting its memory state status flag to `Run`.

Always use the **bos** commands discussed in this chapter to change the BOS Server's memory state and/or the information in *BosConfig*. Never edit *BosConfig* directly.

4.2.4. About Starting and Stopping the Database Server Processes

Whenever you start or stop the Authentication Server (*kaserver*), Protection Server (*ptserver*), Volume Location Server (*vlserver*), or Backup Server (*buserver*) other than temporarily, you must follow the instructions in the *AFS Installation Guide* for installing or removing a database server machine. The instructions include all the steps necessary to guarantee continued correct functioning of all processes. The most important steps to follow are

- Starting/stopping all four database server processes on that machine. The other server processes and all client processes expect all four database server processes to run on the machines listed in *CellServDB*. You cannot indicate in *CellServDB* that a machine is running only some of the database server processes.

- Altering both the server and client versions of *CellServDB* to reflect the change.
- Restarting the database server processes on the other database server machines so that the lightweight Ubik processes they are running recognize the change in their peers. If you fail to do this, the databases can get out of sync. Perform this step on one machine at a time.

Note: Keep these guidelines in mind as you perform the steps described in this chapter. If the process you manipulate is a database server process, refer to this section for information about additional steps you must also perform.

4.2.5. About Starting and Stopping the Update Server

The *AFS Installation Guide* recommends that you designate the first file server machine you install in your cell as the system control machine, and the first machine of each CPU/operating system type as the binary distribution machine for its type. You accomplish both by installing the *upserver* process on the system control machine or binary distribution machine, and *upclientbin* and (if using the United States edition of AFS) *upclientetc* processes on other machines.

Note: If you are using the International version of AFS, you should not use an *upclientetc* process to distribute the contents of */usr/afs/etc*. You should ignore all references to the process in this chapter.

There is normally no reason to switch these roles to different machines, unless you remove the current machine from service entirely. Removing the *upserver* from either the system control machine or a binary distribution machine has far-reaching effects.

- For the system control machine, you must assign the role to another machine by running the *upserver* on it and then replace the *upclientetc* on every other file server machine to reference the new system control machine.
- For a binary distribution machine, you must assign the role to another machine of that CPU/operating system type (unless you are no longer using any file server machines of that type) and then replace the *upclientbin* on every machine of that type to reference the new binary distribution machine.

Note: Keep these guidelines in mind as you perform the steps described in this chapter. If the process you manipulate is an *upserver* process, refer to this section for information about additional steps you must also perform.

4.3. Listing Process Status and Information from *BosConfig*

It may be necessary at times to verify that all the processes that should be running are indeed in operation. Use the **bos status** command to examine the state of processes on a file server machine. If you use the **-long** flag with the command, the output displays most of the information in a process's entry in *BosConfig*, including its type and command parameter(s). It also displays a warning message if the mode bits on directories under */usr/afs* deviate from what the BOS Server expects to find (that is, if they do not match the standard, recommended values).

⇒ To learn the status of all processes on a file server machine

Step 1: Issue the **bos status** command.

% **bos status** <*machine name*>

where

stat is an acceptable abbreviation of **status**.

machine name specifies the file server machine for which to display process status.

Section 4.3.1 details the output from this command.

⇒ To learn the status of certain processes on a file server machine

Step 1: Issue the **bos status** command, specifying only certain processes.

% **bos status** <*machine name*> <*server process name*>$^+$

where

stat is an acceptable abbreviation of **status**.

machine name specifies the file server machine on which you want to examine process status.

server process name
specifies each process for which to display status. Refer to a process by the name assigned at creation.

Section 4.3.1 details the output from this command.

⇒ To examine information from *BosConfig*

Step 1: Issue the **bos status** command, adding the **-long** flag.

% **bos status** <*machine name*> **[**<*server process name*>$^{+}$**] -long**

where

stat is an acceptable abbreviation of **status**.

machine name specifies the file server machine on which you want to examine process status.

server process name
is optional here. If you do not provide it, the status and entry for every process in *BosConfig* is displayed. If you provide it, the output includes only the specified processes. Refer to a process by the name assigned at creation.

-long provides, in addition to status, information from the process's *BosConfig* entry: its type, its status flag, its command parameters, etc.

Section 4.3.2 details the output from this command.

4.3.1. Possible Process Statuses as Reported by bos status

If you do not provide the **-long** flag, **bos status** reports only the actual status of the process on the file server machine. The primary status possibilities are

- `currently running normally`. The process's status flag in *BosConfig* is `Run`. For cron entries, this message indicates that the command is still scheduled to run, not necessarily that it was executing as the **bos status** command was issued.
- `temporarily enabled`. This indicates that a system administrator has started the process running using **bos startup**. Its status flag in *BosConfig* is still `NotRun`.
- `temporarily disabled`. This indicates one of two things: either a system administrator stopped the process with the **bos shutdown** command, or the BOS Server ceased trying to restart the process after numerous failed attempts. In the latter case, the supplementary message "`stopped for too many errors`" appears. The status flag in *BosConfig* for temporarily disabled processes is still `Run`.
- `disabled`. Indicates that the process is not and should not be running, because a system administrator has set its status flag in the machine's *BosConfig* file to `NotRun`.

The output for the *fs* process always includes an "`Auxiliary status`," which may be

- `file server running`. The File Server and Volume Server components of the File Server process are running normally.
- `salvaging file system`. The Salvager is running, so the File Server and Volume Server are temporarily disabled. The BOS Server will restart them as soon as the Salvager is finished.

The output for a cron process includes an "`Auxiliary status`" that reports when the command will next execute; see the example that follows. In addition, the output for any process can include the supplementary message "`has core file`," which means the process failed and created a core file in */usr/afs/logs*. However, the failure did not prevent the BOS Server from restarting the process.

The following example includes a user-defined cron entry called "backupusers":

`%` **bos status fs3.transarc.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 Jan 8 02:00:00 1993.
```

4.3.2. Information from *BosConfig* as reported by bos status

If you include the **-long** flag on **bos status**, the output for each process includes the following additional information from *BosConfig*:

- The process's type.
- The time stamp of the last "`proc start`" and how many times the current BOS Server has re/started the process (called "`proc starts`").
- The "exit" time when the process (or one of the processes in the *fs* process) last failed. This does not appear if the process has not failed while the current BOS Server has been running.
- The "error exit" time when the process (or one of the processes in the *fs* process) last failed due to an error. A further explanation, such as "`due to shutdown request`" may appear. This line does not appear if the process has not failed while the current BOS Server has been running.
- The commands the BOS Server runs to start the process (as specified on its command parameter line(s) in *BosConfig*).

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 correct protections for the directories and files under */usr/afs* are

/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`

See the *AFS Installation Guide* for more information about setting the protections on these files and directories.

An example of a **-long** listing for an *fs* instance follows:

`%` **bos status fs3.transarc.com fs -long**

```
Instance fs, (type is fs), currently running normally.
    Auxiliary status is file server running
    Process last started at Mon Jan 2 8:29:19 1989 (3 proc starts)
    Last exit at Mon Jan 2 8:29:19 1989
    Last error exit at Mon Jan 2 8:29:19 1989, 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'
```

4.4. Listing Which Machines Are Taking Special Roles

It is assumed that you know which machines in your cell are the system control machine, binary distribution machines, and database server machines. Section 1.4 describes these roles if you need to review. Use the instructions in this section if you need to check which machine is taking a certain role.

⇒ To learn if a machine is a database server machine

Step 1: Inspect */usr/afs/etc/CellServDB* on any file server machine.

% **bos listhosts** <*machine name*>

where

listh is an acceptable abbreviation of **listhosts**.

machine name specifies a file server machine.

The machines listed in the output are the cell's database server machines.

Step 2: **(Optional).** If you want to verify that a machine listed in the output of **bos listhosts** is running the processes that define it as a database server machine, use **bos status** to verify that it is running *buserver*, *kaserver*, *ptserver*, and *vlserver*. Section 4.3 describes this command fully.

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

If *machine name* is a database server machine, the output from **bos status** will include

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

⇒ To learn which machine is the system control machine

Step 1: Issue **bos status** for any file server machine. Section 4.3 describes this command fully.

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

Step 2: See sections 4.4.1, 4.4.2, and 4.4.3 for a complete explanation of the output. 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.

⇒ To learn which machine is distributing binaries for a CPU/OS type

Step 1: Issue **bos status** for a file server machine of the CPU/operating system type you are checking (you should know the CPU/operating system type of each file server machine). Section 4.3 describes **bos status** fully.

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

Step 2: See sections 4.4.1, 4.4.2, and 4.4.3 for a complete explanation of the output. 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.

4.4.1. What Output Looks Like For a Simple File Server Machine

Interpreting the output of **bos status** is most straightforward if you have contacted a simple file server machine (one that is neither the system control machine nor a binary distribution machine). There is no *upserver* running on a simple file server machine, so you will see

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

There will be an "`Instance upclientbin`" entry and (if you are using the United States edition of AFS) an "`Instance upclientetc`" entry in the output. The following illustrates the output:

```
Instance upclientbin, (type is simple) currently running normally.
Process last started at Wed Jan 3 23:37:09 1993 (1 proc start)
Command 1 is ’/usr/afs/bin/upclient fs7.transarc.com
           -t 60 /usr/afs/bin’

Instance upclientetc, (type is simple) currently running normally.
Process last started at Mon Jan 1 05:23:49 1993 (1 proc start)
Command 1 is ’/usr/afs/bin/upclient fs1.transarc.com
           -t 60 /usr/afs/etc’
```

In the first example, the first `Command 1` illustrates that this machine is referencing */usr/afs/bin* on `fs7.transarc.com`. Therefore, `fs7.transarc.com` is the binary distribution machine for this machine’s CPU/operating system type.

In the second example, `Command 1` illustrates that this machine is referencing */usr/afs/etc* on `fs1.transarc.com`. Therefore, `fs1.transarc.com` is the system control machine.

4.4.2. What Output Looks Like for the System Control Machine

If you contact the system control machine, the output of **bos status** looks something like

```
Instance upserver, (type is simple) currently running normally.
Process last started at Mon Jan 1 05:23:54 1993 (1 proc start)
Command 1 is ’/usr/afs/bin/upserver’
```

If you followed AFS recommendations, the system control machine also serves as the binary distribution machine for its CPU/operating system type. A single *upserver* is enough to distribute both kinds of updates. In that case, the output contains the error 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 contains a listing for *upclientbin* and an error message for *upclientetc*.

```
Instance upclientbin, (type is simple) currently running normally.
Process last started at Mon Jan 1 05:23:49 1993 (1 proc start)
Command 1 is '/usr/afs/bin/upclient fs5.transarc.com
           -t 60 /usr/afs/bin'
bos: failed to get instance info for 'upclientetc' (no such entity)
```

4.4.3. What Output Looks Like For a Binary Distribution Machine

If you contact the binary distribution machine, the output of **bos status** looks something like

```
Instance upserver, (type is simple) currently running normally.
Process last started at Mon Jan 1 05:23:54 1990 (1 proc start)
Command 1 is '/usr/afs/bin/upserver'
```

It also includes an indication that no *upclientbin* instance is running.

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

Unless this machine also happens to be the system control machine, you should also see an "`Instance upclientetc`" that references the system control machine's */usr/afs/etc* directory. If output like the following is missing, this machine is the system control machine as well as the binary distribution machine for its CPU/OS type:

```
Instance upclientetc, (type is simple) currently running normally.
Process last started at Mon Jan 1 05:23:49 1990 (1 proc start)
Command 1 is '/usr/afs/bin/upclient fs3.transarc.com
           -t 60 /usr/afs/etc'
```

4.5. Creating and Removing Processes

To start up a new process on a file server machine, create an entry for it in *BosConfig* with the **bos create** command, which sets the process's status flag to `Run` and starts it running immediately. To do this, the binary file for the new process must be installed, preferably in the */usr/afs/bin* directory. See chapter 11 for details.

There are two ways to stop a process permanently. The first is to use the **bos stop** command to change the process's status flag to `NotRun`; the process appears as `disabled` in the output from **bos status**. The second is to use the **bos delete** command to remove the process from *BosConfig* altogether; the process no longer appears in status output at all. The creation and removal commands in this section change the process's status flag in *BosConfig* as well as the BOS Server's memory state, so the effect is both immediate and permanent.

Note: If you are starting or stopping a database server process in the manner described in this section, follow the complete instructions in the *AFS Installation Guide* for creating or removing a database server machine. If you run one database server process on a given machine, you must run them all; see section 4.2.4 for more information. Similarly, if you are stopping the *upserver* on the system control machine or a binary distribution machine, you must be aware of all necessary steps; see section 4.2.5 for more information.

⇒ To create and start up a new simple process

Step 1: Verify that you have the privilege necessary for creating processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that binaries for the process are available 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 */usr/afs/bin*.

% **ls /usr/afs/bin**

If the binaries are not present, install them on the binary distribution machine of the appropriate CPU/operating system type, and wait for the Update Server client portion to copy them to this machine. See chapter 11 for details.

Step 3: Create an entry in *BosConfig* for the new process, and start it running. Type the following command on a single line:

% **bos create** <*machine name*> <*server process name*> **simple** <*command lines*>$^+$
[-notifier <*Notifier program*>**]**

where

cr is an acceptable abbreviation of **create**.

machine name specifies the file server machine on which to create the process.

server process name
specifies the process you want to create and start. You may choose any name for a process; it is best to give the process the same name as its binary file (and to choose the same name on every machine running this process). The standard names, and those specified in the *AFS Installation Guide*, are

buserver for the Backup Server.

kaserver for the Authentication Server.

runntp for the process that controls the Network Time Protocol Daemon.

ptserver for the Protection Server.

upclientbin for the client portion of the Update Server that references */usr/afs/bin* on the binary distribution machine for this machine's CPU/operating system type.

upclientetc for the client portion of the Update Server that references */usr/afs/etc* on the system control machine.

upserver for the server portion of the Update Server.

vlserver for the Volume Location Server.

simple specifies the type of this process.

command lines specifies the command the BOS Server executes to start and restart this process. For a simple process, only one line specifying the pathname of the process's binary file (for example, */usr/afs/bin/ptserver* for the Protection Server) is necessary.

-notifier specifies the pathname of a program to execute when the process terminates. When a process with a corresponding *Notifier program* terminates, BOS Server will execute the program. See the entry on **bos create** in the *AFS Command Reference Manual* for more information on *Notifier program*s

⇒ To create and start a new fs-type process

Step 1: Verify that you have the privilege necessary for creating processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that binaries for the process are available 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 */usr/afs/bin*.

% **ls /usr/afs/bin**

If the binaries are not present, install them on the binary distribution machine of the appropriate CPU/operating system type, and wait for the Update Server client portion to copy them to this machine. See Chapter 11 for details.

Step 3: Create an entry in *BosConfig* for the new process, and start it running. Type the following command on a single line:

% **bos create** <*machine name*> <*server process name*> **fs**
/usr/afs/bin/fileserver /usr/afs/bin/volserver /usr/afs/bin/salvager
[-notifier <*Notifier program*>**]**

where

cr is an acceptable abbreviation of **create**.

machine name specifies the file server machine on which to create the process.

server process name
specifies the new fs-type process. The standard name is *fs*.

fs specifies the type of this process.

/usr/afs/bin/fileserver, **/usr/afs/bin/volserver**, and **/usr/afs/bin/salvager**
specify the commands the BOS Server executes to start and restart this process.

-notifier specifies the pathname of a program to execute when the process terminates. When a process with a corresponding *Notifier program* terminates, BOS Server will execute the program. See the entry on **bos create** in the *AFS Command Reference Manual* for more information on *Notifier program*s

⇒ To create and start a new cron process

Step 1: Verify that you have the privilege necessary for creating processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that binaries for the process are available 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 */usr/afs/bin*.

% **ls /usr/afs/bin**

If the binaries are not present, install them on the binary distribution machine of the appropriate CPU/operating system type, and wait for the Update Server client portion to copy them to this machine. See Chapter 11 for details.

Step 3: Create an entry in *BosConfig* for the new process, and start it running. Type the following command on a single line:

% **bos create** <*machine name*> <*server process name*> **cron** <*command lines*>$^+$
[-notifier <*Notifier program*>**]**

where

cr is an acceptable abbreviation of **create**.

machine name specifies the file server machine on which to create the process.

server process name
specifies the new cron process. Type either upper or lower case letters; do not include spaces within the name.

cron specifies the type of this process.

command lines specifies the two command parameters for this process. The first parameter is the command the BOS Server is to execute. It can be either

- A complete pathname (as used for a simple process).
- A standard AFS command or other command. For example, "*/usr/afs/bin/vos backupsys*" causes every volume in the file system to be backed up. In this case, enclose the parameter in quotes because it contains a space.

The second parameter specifies the time at which the BOS Server executes the first command parameter. Surround this parameter with quotes if it contains spaces. Legal values are

- *never*, preventing the entry from executing without actually removing it from *BosConfig*.
- *now*, resulting in immediate execution, but never again.
- A certain day of the week, at a certain time (specified in that order). The command executes once per week (for example, *"sun 3:50 am"*).
- A certain time. The command executes once per day (for example, *21:30*).

If you specify a day, it must appear first, in lowercase letters. Type either the whole name or the first three letters as an abbreviation (*sunday* or *sun*, *thursday* or *thu*, and so on).

When indicating time, separate the hours from the minutes with a colon (*4:00* is the default). Twenty-four hour time (military time) is understood by default, but you can also use only *1:00* through *12:00* if you add *am* (or *a.m.*) or *pm* (or *p.m.*).

-notifier specifies the pathname of a program to execute when the process terminates. When a process with a corresponding *Notifier program* terminates, BOS Server will execute the program. See the entry on **bos create** in the *AFS Command Reference Manual* for more information on *Notifier program*s

⇒ To stop a process and remove it from *BosConfig*

Step 1: Verify that you have the privilege necessary for removing processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Change each process's status flag in *BosConfig* to `NotRun`, and stop it from running on the machine. You must also do this for cron processes, even though they do not run continuously.

% **bos stop** <*machine name*> <*server process name*>+ **[-wait]**

where

sto is an acceptable abbreviation of **stop**.

machine name specifies the file server machine on which to stop the process.

server process name
specifies each process to disable on *machine name*. Use the name assigned to the process at creation. You can also specify a user-defined cron process.

-wait indicates that the command shell prompt is not to return until all specified processes have stopped. You may shorten it to **-w**. If you omit it, the prompt returns almost immediately, even if all processes are not yet stopped.

Step 3: Remove the process(es) from *BosConfig*.

% **bos delete** <*machine name*> <*server process name*>+

where

d is an acceptable abbreviation of **delete**.

machine name specifies the file server machine on which *BosConfig* is to be altered.

server process name
specifies each process to delete from *BosConfig*. Provide the same name(s) as in step 2.

4.6. Stopping and Starting Processes "Permanently"

You can stop a process so that the BOS Server no longer attempts to monitor it by using **bos stop** to set its status flag to `NotRun` in both the BOS Server's memory and in *BosConfig*. The process's entry remains in *BosConfig*, but it does not run again until you issue **bos start**, which changes its status flag back to `Run` in both the BOS Server's memory and *BosConfig*.

You cannot start and stop the BOS Server itself with these commands, because it is not listed in *BosConfig*. You can stop and immediately restart the BOS Server along with all other processes by using the **-bosserver** flag on **bos restart**. See section 4.8 for details.

Note: If you start or stop a database server process in the manner described in this section, follow the complete instructions in the *AFS Installation Guide* for creating or removing a database server machine. If you run one database server process on a given machine, you must run them all; see section 4.2.4 for more information. Similarly, if you stop the *upserver* on the system control machine or a binary distribution machine, you should be aware of all necessary steps; see section 4.2.5 for more information.

⇒ To stop processes by changing their status flags to `NotRun`

Step 1: Verify that you have the privilege necessary for stopping processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Stop each process, and set its status flag to `NotRun` in *BosConfig* and the BOS Server's memory.

% **bos stop** <*machine name*> <*server process name*>$^+$ **[-wait]**

where

sto is an acceptable abbreviation of **stop**.

machine name specifies the file server machine on which to stop the process.

server process name specifies each process to disable on *machine name*. Use the name assigned to the process at creation.

-wait indicates that the command shell prompt is not to return until all specified processes have stopped. You may shorten it to **-w**. If you omit it, the prompt returns almost immediately, even if all processes are not yet stopped.

⇒ To start processes by changing their status flags to `Run`

Step 1: Verify that you have the privilege necessary for starting processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Change the process's *BosConfig* status flag to `Run` in both *BosConfig* and the BOS Server's memory, and start it running.

% **bos start** <*machine name*> <*server process name*>+

where

start must be typed in full.

machine name specifies the file server machine on which to start running the process(es).

server process name
specifies each process you wish to start on *machine name*. Use the name assigned to the process at creation.

4.7. Stopping and Starting Processes Temporarily

It may sometimes be 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 consults *BosConfig* and changes its memory state to match what it finds there).

To stop processes temporarily, use the **bos shutdown** command. Each process stops running because its status flag in the BOS Server's memory changes to `NotRun`, even though its status flag in *BosConfig* may still be `Run`.

You can start temporarily stopped processes with **bos startup**, which changes the status flags in the BOS Server's memory to `Run`. The processes start even if their status flags in *BosConfig* remain `NotRun`. The newly started processes are completely new instances; if you install new binaries during the time the processes are shutdown, the new binaries are used when you issue **bos startup**.

Because the **bos startup** command also allows you to start processes without changing their *BosConfig* status flags to `Run`, it is useful for run-testing a server process without enabling it permanently. To stop and start processes by changing their status flags in *BosConfig*, see section 4.6; to stop and immediately restart a process, see section 4.8.

Note: Do not temporarily stop a database server process on all machines at once. Doing so will make the database completely unavailable.

⇒ To stop processes temporarily

Step 1: Verify that you have the privilege necessary for stopping processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: 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 an acceptable abbreviation of **shutdown**.
machine name	specifies the file server machine on which to stop processes temporarily.
instances	specifies each process to stop temporarily on *machine name*. Use the name assigned to the process at creation.
-wait	indicates that the command shell prompt is not to return until all specified processes have actually stopped. You may shorten it to **-w**. If you omit it, the prompt returns almost immediately, even if all processes are not yet stopped.

⇒ To start all stopped processes that have *BosConfig* status flag `Run`

Step 1: Verify that you have the privilege necessary for starting processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

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

Step 2: Start each process on a machine that has status flag `Run` in *BosConfig* 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 file server machine on which you wish to start all processes that have status flag `Run` in *BosConfig*.

⇒ To start specific temporarily-stopped processes

Step 1: Verify that you have the privilege necessary for starting processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

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

Step 2: Start specific processes by changing their status flags in the BOS Server's memory to `Run`. Their status flags in *BosConfig* remain unchanged.

`%` **bos startup** <*machine name*> <*instances*>$^+$

startup must be typed in full.

machine name names the file server machine on which to start temporarily-stopped processes.

instances specifies each process you wish to stop temporarily on *machine name*. Use the name assigned to the process at creation.

4.8. Stopping and Immediately Restarting Processes

It may sometimes be necessary to stop and immediately restart processes, rather than waiting until the BOS Server performs its daily check for new binary files. For instance, you may need to do so if new binaries are installed and you need to use them immediately. The **bos restart** command provides this functionality:

- To stop and restart all the processes on a file server machine that have their *BosConfig* status flag set to `Run` (including the BOS Server), use **bos restart** with the **-bosserver** flag.
- To stop and restart all processes except the BOS Server that have their *BosConfig* status flag set to `Run`, use **bos restart** with the **-all** flag.
- To stop and restart specific processes regardless of how their *BosConfig* status flags are set, use **bos restart** with the **-instance** flag.

Restarting processes causes a service outage. It may be 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 section 4.9.

⇒ To stop and restart all processes including the BOS Server

Step 1: Verify that you have the privilege necessary for restarting processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Stop and restart BOS Server, which then restarts every process that has status flag `Run` in *BosConfig*.

% **bos restart** <*machine name*> **-bosserver**

where

res	is an acceptable abbreviation of **restart**.
machine name	specifies the file server machine on which to restart all processes.
-bosserver	indicates that the current BOS Server should stop itself and all processes running on *machine name*. A new BOS Server instance starts; it then starts new instances of all processes marked with status `Run` in *BosConfig*.

⇒ To stop and immediately restart all processes except BOS Server

Step 1: Verify that you have the privilege necessary for restarting processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Stop and immediately restart every process that has status flag `Run` in *BosConfig*. The BOS Server does not stop running.

% **bos restart** <*machine name*> **-all**

where

res is an acceptable abbreviation of **restart**.

machine name specifies the file server machine on which to stop and restart processes.

-all indicates that the current BOS Server should stop and immediately restart all processes marked with status `Run` in *BosConfig*; the BOS Server process will not stop itself.

⇒ To stop and immediately restart specific processes

Step 1: Verify that you have the privilege necessary for restarting processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Stop and immediately restart each specified process, regardless of its status flag setting in *BosConfig*.

% **bos restart** <*machine name*> <*instances*>$^+$

where

res is an acceptable abbreviation of **restart**.

machine name names the file server machine on which to restart the specified processes.

instances specifies each process you want to stop and immediately restart. Use the name assigned to the process at creation.

4.9. Setting the BOS Server's Restart Times

If a process runs continuously for a long period of time, its performance can degrade due to core leaks. (A ***core leak*** develops as a process continues to allocate virtual memory but does not free it again. When the memory is completely exhausted, the machine crashes.) To minimize core leaks, the BOS Server automatically restarts itself weekly and then restarts all the other server processes on its machine that have status flags of `Run` in the *BosConfig* file. This is equivalent to issuing the **bos restart** command with the **-bosserver** flag. The default weekly restart time is Sunday at 4:00 a.m.

The BOS Server also 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 */usr/afs/bin* is more recent than the time at which the process last started on the file server machine, it restarts the process so that a new instance starts using the new binary file. The default daily 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 people using those processes on that machine. The longest outages result from restarting the *fs* process; they can last as long as half an hour, depending on the number of volumes on the file server machine.

The */usr/afs/local/BosConfig* file on each file server machine records the two restart times. Because the information is local, the times may differ between machines. You can learn the current setting of both restart times with one command, **bos getrestart**. You may reset both times using **bos setrestart**. As with other information in *BosConfig*, <u>do not edit the restart times directly</u>. Always use the **bos setrestart** command.

Note: If the time set with **bos setrestart** is within one hour of the current time, the BOS Server will not perform the restart until that time tomorrow.

⇒ To learn the current setting of the two restart times

Step 1: Issue **bos getrestart**.

% **bos getrestart** **<***machine name***>**

where

getr is an acceptable abbreviation of **getrestart**.

machine name specifies the file server machine on which you want to check the restart times.

⇒ To set the weekly restart time

Step 1: Verify that you have the privilege necessary for setting restart times (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Set the restart time by issuing **bos setrestart** with the **-general** flag.

% **bos setrestart** <*machine name*> **"**<*time to restart server*>**" -general**

where

setr is an acceptable abbreviation of **setrestart**.

machine name specifies the file server machine.

time to restart server
specifies the day and time at which the BOS Server is to restart itself and all other processes. Enclose it in quotes because it contains spaces.

Specify the day first, in lowercase letters. Either type the whole name or the first three letters as an abbreviation (*sunday* or *sun*, *thursday* or *thu*, and so on).

When specifying time, separate the hours from the minutes with a colon (*4:00* is the default). Twenty-four hour time (military time) is understood by default, but you can also use *1:00* through *12:00* and add *am* (or *a.m.*) or *pm* (or *p.m.*).

Note: If this time is within one hour of the current time, the BOS server will not perform the restart until tomorrow.

-general identifies this as the weekly restart time (rather than the daily binary-checking time).

⇒ To set the scheduled time for daily binary checking

Step 1: Verify that you have the privilege necessary for setting restart times (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Set the binary checking time.

% **bos setrestart** <*machine name*> **"**<*time to restart server*>**" -newbinary**

where

setr is an acceptable abbreviation of **setrestart**.

machine name specifies the file server machine whose BOS Server is to check binaries at the indicated time.

time to restart server
indicates the time each day that the BOS Server performs its check through */usr/afs/bin* and restarts server processes as necessary. Enclose it in quotes if it contains spaces.

When indicating time, separate the hours from the minutes with a colon (for instance, *5:00* is the default). Twenty-four hour time is understood, but you can also use *1:00* through *12:00* and add *am* (or *a.m.*) or *pm* (or *p.m.*).

Note: If this time is within one hour of the current time, the BOS server will not perform the restart until tomorrow.

-newbinary distinguishes this restart time from the weekly one.

4.10. Examining a File Server Machine's 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. The directory contains the following log files:

- The *AuthLog* file generated by the Authentication Server.
- The *BackupLog* file generated by the Backup Server.
- The *BosLog* file generated by the BOS Server.
- The *FileLog* file generated by the File Server.
- The *SalvageLog* file generated by the Salvager.
- The *VolserLog* file generated by the Volume Server.

The **bos getlog** command can be used to display one of these log files on a file server machine. The command assumes the file to be examined resides in the */usr/afs/logs* directory on the specified machine. For that reason, the name of the file (for example, *FileLog*) is sufficient to indicate the file to be examined. If the command is to be used to examine a file in a different directory, a pathname to the file must be provided.

⇒ To examine a server process log file

Step 1: Verify that you have the necessary privilege for reading the log file (inclusion in */usr/afs/etc/UserList* on the machine on which the file to be examined resides). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Issue the **bos getlog** command to examine the log file.

% **bos getlog** <*machine name*> <*log file to examine*>

where

getl is an acceptable abbreviation of **getlog**.

machine name specifies the file server machine on which the log file to be examined resides.

log file to examine
names the log file to be displayed. If a plain file name (one with no slashes) is specified, the file is assumed to reside in the */usr/afs/logs* directory. The usual log files in the directory are *AuthLog*, *BackupLog*, *BosLog*, *FileLog*, *SalvageLog*, and *VolserLog*. A full or relative pathname (one with at least one slash) is allowed. Relative pathnames are interpreted relative to */usr/afs/logs*.

Note: If you do not periodically shutdown and restart the database server processes, the corresponding log files may grow unmanageably large. To prevent this, you may periodically delete the contents of the log files using the UNIX **rm** command.

4.11. Rebooting a File Server Machine

While not difficult, rebooting a file server machine should not be common practice. It represents an extreme solution, and it is hoped that you will be able to find other ways to solve AFS-related problems (restarting server processes may help, for example).

Rebooting also causes a service outage. If the machine is a database server machine, the entire file system can become unavailable to everyone; if it is a regular file server machine, people using volumes on that machine are inconvenienced.

You can reboot a file server either by typing commands at its console or by using **bos exec** from a remote machine. If you reboot a workstation while at its console, you can track the status of the reboot as it happens; this is not possible with remote rebooting. Otherwise, remote rebooting is just as effective, and it does not require you to leave your present location. (Note that remote rebooting works only because the BOS Server, which executes any command invoked with **bos exec**, is recognized in the UNIX file system as "root"—the only user allowed to reboot a machine.)

In either case, the AFS server processes restart automatically after the reboot, provided the machine's */etc/rc* file contains the following instruction to restart the BOS Server:

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

⇒ To reboot a file server machine from its console

Step 1: Verify that you have the privilege necessary for shutting down AFS server processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Become "root" in the machine's UNIX file system, if you are not already, by issuing the **su** command.

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

Step 3: Have the BOS Server shut down the other AFS server processes running on the machine. The BOS Server will not shut itself down, but terminates safely when you turn off the machine.

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

where

shut is an acceptable abbreviation of **shutdown**.

machine name is the file server machine you are about to turn off.

-wait indicates the command shell prompt is not to return until all processes actually shut down. If you omit it, the prompt returns almost immediately, even if all processes have not yet shut down.

Step 4: Reboot the machine. On many system types, the appropriate command is **/etc/reboot**; consult your UNIX administrator's guide.

\# **/etc/reboot**

⇒ To reboot a file server machine remotely

Step 1: Verify that you have the privilege necessary for shutting down AFS server processes and executing commands remotely (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Have the BOS Server shut down the other AFS server processes running on the machine. The BOS Server does not shut itself down, but terminates safely when you turn off the machine.

% **bos shutdown** <*machine name*> **[-wait]**

where

shut is an acceptable abbreviation of **shutdown**.

machine name is the file server machine you are about to reboot.

-wait indicates the command shell prompt is not to return until all processes actually shut down. If you omit it, the prompt returns almost immediately, even if all processes have not yet shut down.

Step 3: Reboot the machine remotely. On many system types, the appropriate command is **/etc/reboot**; consult your UNIX administrator's guide.

% **bos exec** <*machine name*> **/etc/reboot**

where *machine name* is the file server machine you wish to reboot.

5. Managing Volumes

This chapter explains how to manage the ***volumes*** in your cell. The volume is the designated unit of administration in AFS, so a large amount of your time will be spent managing them.

This chapter has the following sections:

5.1 Summary of Commands 5-1
5.2 About Volumes. 5-3
5.3 Creating ReadWrite Volumes 5-10
5.4 Clones and Cloning 5-13
5.5 Creating and Releasing ReadOnly Volumes (Replication) 5-14
5.6 Creating Backup Volumes 5-20
5.7 Mounting Volumes (Creating, Removing, and Listing Mount Points) 5-24
5.8 Listing Information About Volumes 5-31
5.9 Moving Volumes 5-40
5.10 Synchronizing the VLDB and Volume Headers 5-42
5.11 Salvaging Volumes. 5-45
5.12 Setting and Listing Volume Quota and Current Size 5-49
5.13 Removing Volumes and their Mount Points 5-53
5.14 Dumping and Restoring Volumes 5-58
5.15 Renaming Volumes 5-63
5.16 Unlocking and Locking VLDB Entries 5-65

5.1. Summary of Commands

Create a ReadWrite volume . . . . . **vos create**
Create a ReadOnly volume . . . . . **vos addsite** and/or **vos release**
Create a 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**
Learn a volume’s name . . . . . **fs listquota** or **fs examine**
Learn a volume’s ID number . . . . . **fs examine** or **vos examine** or **vos listvol**
List partition’s size and space available . . . . . **vos partinfo**
Learn a volume’s location . . . . . **fs whereis** or **vos examine**
Create a mount point . . . . . **fs mkmount**
Remove a mount point . . . . . **fs rmmount**
Examine a mount point . . . . . **fs lsmount**
Move a ReadWrite volume . . . . . **vos move**
Synchronize VLDB and volumes . . . . . **vos syncvldb** and **vos syncserv**
Set volume quota . . . . . **fs setvol** or **fs setquota**
Examine volume quota . . . . . **fs quota** or **fs listquota** or **fs examine**
Learn a volume’s current size . . . . . **fs listquota** or **fs examine**

List all volumes on a machine/partition . . . . **vos listvol**
Remove a ReadWrite volume . . . . **vos remove** and **fs rmmount**
Remove a ReadOnly volume . . . . **vos remove**
Remove a Backup volume . . . . **vos remove** and **fs rmmount**
Remove a volume; no VLDB change . . . . **vos zap**
Remove a mistaken ReadOnly site definition. . . . . . . **vos remsite**
Remove a VLDB entry; no volume change . . . . **vos delentry**
Dump a volume. . . . . **vos dump**
Restore a dumped volume. . . . . **vos restore**
Rename a volume . . . . **vos rename**, **fs rmmount** and **fs mkmount**
Unlock a volume. . . . . **vos unlock**
Unlock multiple volumes . . . . **vos unlockvldb**
Lock a volume . . . . **vos lock**

5.2. About Volumes

A ***volume*** is a logical unit of disk space that functions like a "container" for the files in a 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 certain directory location in the file tree. This association (or "link") between the volume and its file tree location is referred to as a ***mount point***, and because of AFS's internal workings it looks and acts just like a standard directory element. Thus users can access and manipulate a volume's contents in the same ways they access and manipulate the contents of a standard UNIX directory. For more on the relationship between volumes and directories, see section 5.2.6.

Many of your daily activities will involve manipulating volumes, since they are the basic storage and administrative unit of AFS. Section 5.2.2 explains some of the ways volumes can make your job easier.

5.2.1. The Three Types of Volumes

There are three types of volumes in AFS: ReadWrite, ReadOnly, and Backup. These types are described below:

- Every volume has a single ***ReadWrite*** version, which houses the modifiable versions of the files and directories in that volume. The ReadWrite version of a volume is often referred to as the "ReadWrite source" because volumes of the other two types are derived from it by a copying procedure called ***cloning***. Section 5.3 explains how to create ReadWrite volumes.
- A ***ReadOnly*** volume is a copy of the ReadWrite source volume and may exist at multiple ***sites*** throughout the file system (a site is a particular partition on a particular file server machine). Placing the same data on more that one file server machine in this way is referred to as ***replication***; section 5.2.2 explains its advantages. As its name suggests, a ReadOnly volume cannot change dynamically. All copies of the ReadOnly volume are the same and share the same name, which is the same as the ReadWrite source's name with the addition of a `.readonly` extension. Section 5.5 describes how to create ReadOnly volumes.
- A ***Backup*** volume is a clone of the ReadWrite source volume stored at the same site as the source. As explained further in section 5.2.2, a Backup version is useful because it records the state of the ReadWrite source at a certain time, allowing recovery of data that is later mistakenly changed or deleted. A Backup volume has the same name as its ReadWrite source with the addition of a `.backup` extension. Only one Backup volume can exist at any time. Section 5.6 explains how to create Backup volumes.

  Note: A Backup volume is not the same as the backup <u>of</u> 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. See chapter 6 for information on backing up a volume using the AFS Backup System.

As the preceding paragraphs indicate, the three types of volumes are related to one another: ReadOnly and Backup volumes are both derived from a ReadWrite volume through a process called cloning. ReadOnly and Backup volumes are exact copies of the ReadWrite source at the time they are created. Thus the different types can be thought of as different versions of a volume.

5.2.2. How Volumes Improve System Efficiency

The use of volumes makes your cell easier to manage and more efficient in three ways:

- Volumes are easy to move from one partition and/or one machine to another because they are (by definition) smaller than a partition. One important reason to move volumes is to balance their distribution among all the file server machines. An uneven distribution of volumes can result in a minority of file server machines filling most of the file transfer requests (because they house more volumes or volumes containing more popular files than other machines). This is inefficient because performance-slowing bottlenecks can form.

  The relatively small size of AFS volumes, though, makes it easy to find room on other partitions for some of the volumes from an overcrowded partition. You can perform volume moves as often as necessary without worrying that users will be affected, because the volume's contents are unavailable for only a few seconds, and the Volume Location Server assures that volume access always remains transparent. See section 5.9 to learn how to move volumes.

- Volumes are the unit of replication in AFS. ***Replication*** is the creation a ReadOnly clone from the ReadWrite source and the distribution 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 makes the availability of data independent of the status of a single file server machine; if one machine housing the ReadOnly version is unavailable, other machines containing a ReadOnly copy can still provide the data. In general, only volumes containing popular system files and directories are worth replicating, though it is possible to replicate any ReadWrite volume. Section 5.5 contains information on replicating volumes.

- Volumes are the unit of backup in AFS. Backup has two meanings in AFS. It refers to permanent storage on tape and to the creation of a Backup volume to record the state of a ReadWrite source at a specified time. Chapter 6 provides additional information on the former, section 5.6 of this chapter, provides additional information on the latter.

  You can mount the Backup version of a volume in the file system to make its contents available to users. This can make your job easier because users can consult the Backup version to restore data they have accidentally changed or deleted, freeing you for more important tasks. If you arrange to make new Backup versions of all volumes once a day (presumably overwriting the former Backup), then users will always be able to retrieve the previous day's version of a file. See section 5.6 for instructions.

5.2.3. The Role of the Volume Location Server and VLDB

The Volume Location Database (VLDB) records information about all volumes in a cell. One of the most important pieces of information found in the VLDB is the volume location. Whenever a user requests a file, the Cache Manager contacts the Volume Location (VL) Server, which maintains the VLDB, in order to learn which file server machine is housing the volume that contains the file. The Cache Manager can then request the file from the File Server running on that file server machine. This automatic location of volumes means that access is transparent in AFS and users do not have to keep track of file location.

5.2.3.1. The Information in VLDB Entries

There is an entry in the VLDB for each ReadWrite volume that also contains information about the ReadOnly and Backup versions of the volume. ReadOnly and Backup volumes do not have their own VLDB entries. (The exception to this rule is that a ReadOnly volume may have its own VLDB entry if its ReadWrite source has been removed.)

The following information is available from a VLDB entry:

- The ***volume name*** of the ReadWrite version of the volume. The ReadOnly and Backup versions automatically have the same name with a `.readonly` and `.backup` extension, respectively.
- The ***ReadWrite (RWrite) volume ID number***, if one exists. A volume ID number is simply an identification number guaranteed to be unique within a cell. In almost all cases, the VL Server allocates volume ID numbers automatically, but some commands allow the issuer to assign volume IDs explicitly. Each of the three versions of a volume has a different volume ID number; they often run consecutively, especially if the VL Server assigned them, but this is not a requirement.
- The ***ReadOnly (ROnly) volume ID number***, if one or more exist. All copies of the ReadOnly version share the same volume ID.
- The ***Backup volume ID number***, if one exists.
- The ***ReleaseClone (RClone) volume ID number***, if one exists. The presence of a ReleaseClone indicates that an error occurred during replication. See section 5.5 for more information.
- A ***site count***, which tells how many ReadWrite and ReadOnly volumes exist. There can be only one ReadWrite volume, but as many as eleven ReadOnly copies may exist.
- One or more ***site definitions***, each of which specifies a ***file server machine name*** and ***partition name***. Site definitions specify the location of the ReadWrite version and each copy of the ReadOnly version (if any). No site definition is necessary for the Backup version because it always resides at the same site as the ReadWrite source. There can be only one ReadWrite definition but up to eleven ReadOnly definitions; the number of site definitions should equal the site count.
- A ***type flag*** associated with each site. The type flag indicates the type of volume residing at the site and can have one of the following values: `RW` for ReadWrite or `RO` for ReadOnly.
- A ***release flag*** associated with each site. The presence of a release flag indicates that an error occurred during replication. See section 5.5 for more information.
- An indication if the VLDB entry is locked. Since being unlocked is the default state, there is no explicit indicator if the VLDB entry is unlocked. See section 5.16 for descriptions of the **vos lock** and **vos unlock** command.

The **vos listvldb** command displays much of the information from the VLDB. The **vos examine** command also displays VLDB information in combination with volume header information. See section 5.8 for descriptions of their output.

5.2.4. The Information in Volume Headers

While there is only a single VLDB entry for each ReadWrite volume and its ReadOnly and Backup versions, there is a separate ***volume header*** at the site of each copy of a volume no matter its version. The volume header is the volume in a sense — it is a data structure that records the physical memory addresses on the partition that store the files in the volume. The volume header binds all the files into a logical unit without requiring that they be stored in contiguous memory blocks. In addition to data location, the volume header stores other information about the volume, some of it redundant with the VLDB, so that the Volume Server can access the information even when the VLDB is unavailable.

The information in the volume header includes the following:

- The ***volume name*** of this copy of the volume, with a `.readonly` or `.backup` extension if appropriate.
- The ***volume ID number***.
- The ***type flag*** that identifies the type of volume (`RW` for ReadWrite, `RO` for ReadOnly, `BK` for Backup).
- The volume's ***size*** in kilobytes (`1024` equals a megabyte).
- The volume's ***status*** at the site, which is unrelated to the locked/unlocked status of the VLDB entry. There are three possibilities:
  - `On-line` means that the volume is fully accessible through the file system.
  - `needs salvage` means that the volume is probably corrupted. Run the Salvager using the **bos salvage** command. See section 5.11 for more information on salvaging volumes.
  - `Off-line` means that the volume is not accessible, but there is no reason to suspect that it is corrupted.
- A ***ReadWrite (RWrite) volume ID number***, if one exists; if no ReadWrite volume exists, a zero is displayed. This is the volume ID of this volume's ReadWrite source. If this volume is the ReadWrite version itself, this ID should match the volume ID number.
- A ***ReadOnly (ROnly) volume ID number***, if one exists; if no ReadOnly volumes exist, a zero is displayed. If this volume is a ReadOnly version itself, this ID should match the volume ID number.
- A ***Backup volume ID number***, if one exists; if no Backup volume exists, a zero is displayed. This is the volume ID of the Backup version of this volume. If this volume is the Backup version itself, this ID should match the volume ID number.
- A ***ReleaseClone (RClone) volume ID number***, if one exists; if no ReleaseClone exists, no RClone label is displayed in the output. This is the volume ID number of the last clone made from this volume's ReadWrite source for the purposes of replication. If a ReleaseClone volume ID is displayed, it indicates that the previous replication attempt failed.
- A ***MaxQuota***, which is the maximum amount of disk space the ReadWrite version of the volume may occupy. It does not apply sensibly to ReadOnly or Backup volumes, but is reported for convenience anyway.

- The volume's ***Creation date***, which is the date and time when this copy of the volume was created (for ReadOnly and Backup copies, this is the clone date and time). If the volume has been restored with **backup diskrestore**, **backup volrestore**, or **vos restore**, this is the restore time.
- The date of the volume's ***Last Update***, which is the date and time when the contents of the volume last changed. For ReadOnly and Backup volumes, this should match the creation date, since they are not allowed to change.
- The ***number of times the volume has been accessed*** since the later of the following two times:

  – 12:00 a.m. on the day the command is issued.

  – The last time the volume changed location.

  An access is defined as a fetch or store operation on any file system object stored in the volume.
- The data necessary to generate two statistical tables entitled *Raw Read/Write Stats* and *Writes Affecting Authorship*. These tables provide the following information:

  – The *Raw Read/Write Stats* table contains information on the number of reads (fetches) and writes (stores) made on the specified volume. This information is categorized by reads and writes initiated by a client on the same network as the server (Same Network) versus those initiated from a client on a different network than the server (Diff Network). Both of these categories are further broken down into all reads and writes versus those initiated by authenticated users and processes.

  Note: Diff Network versus Same Network is determined by comparing the network portion of the IP address of the client and server based on the class of network (A, B, or C). If the network portion of the IP addresses match, it is the same network; otherwise, it is considered a different network.

  – The *Writes Affecting Authorship* table contains information on writes made to files and directories in the specified volume. The table categorizes the data to compare writes to files (File Authorship) and writes to directories (Directory Authorship). Both of these categories are further broken down to compare writes made by the owner of the files or directories (Same), and writes made by users/processes that were not the owners of the files or directories (Diff). The time divisions indicate the time between the recorded write and the previous update to the affected file (or directory).

The **vos listvol** command displays information from the volume header. The **vos examine** command also displays information from the volume header, in combination with VLDB information. See section 5.8 for descriptions of their output.

5.2.5. 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 the VL Server gives the Cache Manager an incorrect volume location, the latter will not be able to find the volume. To keep the VLDB accurate, all **vos** commands that affect volume status automatically record the change in the appropriate VLDB entry.

Whenever you issue a command that changes volume status, the Volume Server and VL Server cooperate in the following sequence of steps:

1. The VL Server "locks" the VLDB entry. The lock advises other operations not to manipulate any of the volume versions (ReadWrite, ReadOnly, or Backup), preventing the confusion that could 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 things to their pre-salvage state. See section 5.11 for more information on salvaging.)
3. The Volume Server manipulates the volume. It may set an "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 releases the lock set in step 1 and removes the intention flag set in step 2.

If **vos** operations end prematurely, it is possible for the VLDB and the volumes on file server machines to get out of sync. Section 5.10 explains how to resynchronize them.

5.2.6. About Mounting Volumes

If you want a volume's contents to be visible and accessible to users, you must "mount" the volume at a directory location in your cell's file tree. When you mount a volume, you create an association between the volume and a directory location; this association is called a ***mount point***. To mount a volume, you must have INSERT and ADMINISTER rights in the directory where the mount point is to be created.

An AFS mount point appears and functions like a regular UNIX file system directory, but structurally it is simply a one-line file that indicates the name of the volume associated with the directory. A mount point appears and acts just like a directory because of the way the Cache Manager handles it.

Consider the common case where the Cache Manager needs to retrieve a file requested by a user's application program. In order to determine where the file resides, the Cache Manager has to traverse the file's complete pathname, starting at the root */afs* and continuing all the way to the file itself. When the Cache Manager encounters a mount point during the traversal, it reads it to learn which volume is associated with that directory location. The Cache Manager then looks in the indicated volume and fetches its root directory. The ***root directory*** of a volume lists all the files, subdirectories, and mount points that reside in the volume. The Cache Manager looks up the next element in the pathname in the root directory listing, continuing to interpret any further mount points it encounters until it reaches the volume that houses the requested file.

It is the Cache Manager's ability to interpret mount points that makes them look and act just like directory elements. Mount points are the "glue" that connects the file system tree together, creating the illusion of a single, seamless file tree even though volumes may 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 file tree. Second, it can confuse the Cache Manager about which pathname it followed in order to reach the file (causing unpredictable output from the **pwd** command, for example). If you mount a volume at more than one directory, the ACL associated with the root directory of the volume applies to all of its 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. Section 5.6 discusses mount points and how to create, remove, and examine them.

5.2.7. About Volume Names

You may choose any name for a volume that best indicates its contents to you, keeping two restrictions in mind:

- You may not explicitly add the extensions `.readonly` and `.backup` to volume names, even if they are appropriate. These extensions are reserved for ReadOnly and Backup versions of a volume, respectively; the Volume Server adds them automatically when necessary.
- Volume names may be no longer than 31 characters in length. This means that the Volume Server will not allow you to create a ReadWrite volume with a name longer than 22 characters, to guarantee that the nine-character extension `.readonly` can be added if necessary.

One convention you may find helpful is to name all user volumes "`user`.*user name*" where *user name* 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. Section 2.5 presents more conventions for volume names.

5.3. Creating ReadWrite Volumes

Creating a ReadWrite volume is the first step in creating any type of volume. The **vos create** command is used to create ReadWrite volumes; as part of this creation, the command

- Creates a VLDB entry for the ReadWrite volume (see section 5.2.3 for more on VLDB entries).
- Allocates a volume ID number for the ReadWrite volume and reserves the next two consecutive volume ID numbers for ReadOnly and Backup versions of the ReadWrite source.
- Assigns the volume name you designate.
- Creates a volume header at the site you designate. This is enough to establish the volume's existence at the site, even though the volume does not yet contain any data (see section 5.2.4 for more on volume headers).
- Creates space in the volume header for storage of information about the volume's root directory. This space is not filled with a directory name until you actually mount the volume (see step 6 below). However, the header does record an access control list (ACL) for the future root directory which, by default, grants all seven access rights to *system:administrators*.
- Assigns a default space quota of 5000 kilobyte blocks to the volume.

After creating a volume, you should mount it at a directory location in the file system by creating a mount point there. If you fail to do so, the volume's contents remain invisible and inaccessible. The following instructions include directions for mounting the volume. Section 5.2.6 provides more information on mount points.

⇒ To create a ReadWrite volume and mount it in the file system

Step 1: Verify that you have the privilege necessary for creating volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the ADMINISTER and INSERT rights for the directory in which you wish to mount the volume. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl** <*dir/file path*>

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the INSERT right if necessary.

Step 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 will help guide your decision. You may wish to put the volume on the emptiest partition that meets your other criteria. You can learn how much space is available on a file server machine's partitions (and their total size) by using **vos partinfo**.

% **vos partinfo** <*machine name*> **[**<*partition name*>**]**

where

p is an acceptable abbreviation of **partinfo**.

machine name specifies the file server machine about which you want partition information.

partition name is optional. If you provide it, the output displays the size and space available on the one partition. If you omit it, the output displays the size and space available for all partitions on *machine name*.

Note: The total partition size reported by this command may not agree with the same figure in the output of the standard UNIX **df** command. The **df** total size includes some reserved space that does not show up in this report, and so is likely to be about 10% larger.

Step 4: Select a name for the volume of 22 characters or less. The Volume Server will not allow you to create a volume with a longer name. Your cell probably already has established conventions for volume names. Section 5.2.7 outlines some global restrictions on volume names.

Step 5: Create the volume.

% **vos create** <*machine name*> <*partition name*> <*volume name*>

where

cr is an acceptable abbreviation of **create**.

machine name specifies the file server machine on which to place the new volume.

partition name specifies the disk partition on *machine name* to receive the new volume.

volume name is the name of the new volume.

Step 6: Mount the new volume in the file tree to make its contents visible.

% **fs mkmount** <*directory*> <*volume name*> **[-rw]**

where

mk is an acceptable abbreviation for **mkmount**.

directory names the mount point you wish to create. Do not create a file or directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory.

volume name is the full name of the volume (see step 5).

-rw should be included only in the rare cases where you want the mount point to be a ReadWrite one. See section 5.7.1 for an explanation of ReadWrite mount points.

Step 7: **(Optional).** Verify that you have created the mount point correctly with the **fs lsmount** command described fully in section 5.7.

% **fs lsmount** <*directory*>

5.3.1. Default Settings for a New Volume

Two defaults are set when a ReadWrite volume is created with **vos create**:

- The access control list for the new volume's root directory automatically grants all seven access rights to *system:administrators*.
- The new volume's space quota is automatically set at 5000 kilobyte blocks. (If the volume is created using the **uss** commands, you must set this default; see section 16.2.)

Once created and mounted, a volume's default ACL and default quota can be changed. The *system:administrators* group can be removed from the volume's root directory using **fs setacl**; see section 19.4. A system administrator can change the volume quota using the **fs setquota** command; see section 5.12.

5.4. Clones and Cloning

During the creation of Backup and ReadOnly volumes, a ***clone*** is created from the ReadWrite source volume; this process is referred to as ***cloning***. The creation of a clone is completely internal to the commands that create ReadOnly and Backup volumes; you will never need to directly create a clone as a step in the creation of ReadOnly or Backup volumes.

A clone is not a copy of the data in the ReadWrite source volume. Instead, it is a copy of the ReadWrite volume's ***vnode index***. A vnode index is a table of pointers between each file in the volume and the physical disk blocks where that file's data resides on the partition. From the clone, Backup and ReadOnly volumes are created in the following manner:

- ReadOnly volumes that occupy the same site as their ReadWrite source (also known as ***ReadOnly clones***) and Backup volumes are created by attaching a volume header to a copy of the clone. Since these volumes occupy the same site as the ReadWrite source, they are able to share files with the ReadWrite source; therefore, they require less disk space than the ReadWrite source and ReadOnly volumes that do not occupy the same location as the ReadWrite source. Of course, if the ReadWrite source changes, the actual disk space required by a Backup or ReadOnly clone will grow because the overlap among shared files diminishes. (Figure 5-1 illustrates the relationship among a ReadWrite source volume and a Backup or ReadOnly clone.)
- ReadOnly volumes that do not occupy the same site as the ReadWrite source are copies of the clone and the files in the ReadWrite source.

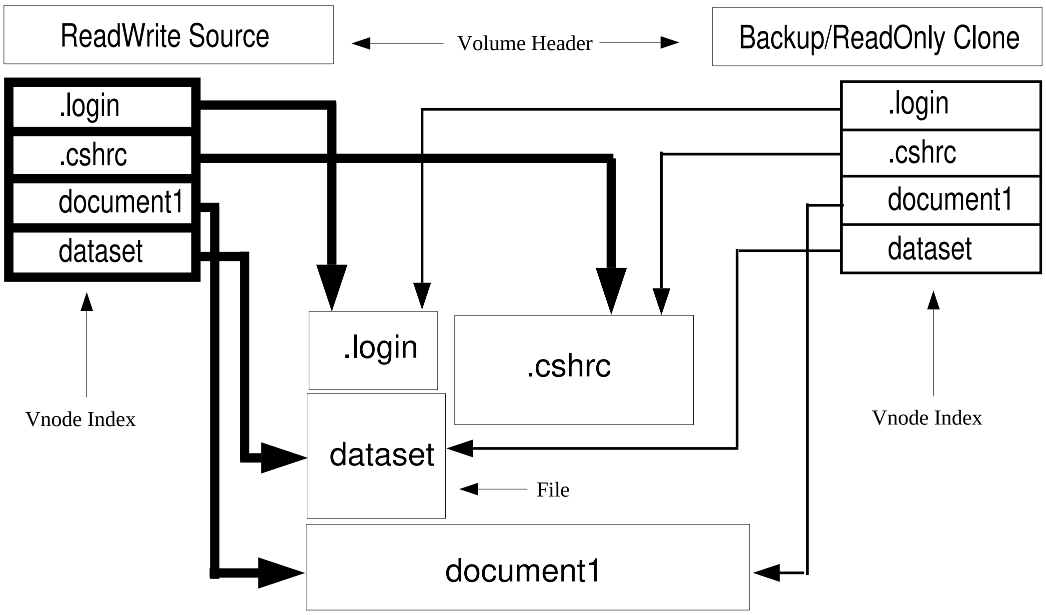

Figure 5-1. File Sharing Among a ReadWrite Source and a Backup/ReadOnly Clone

5.5. Creating and Releasing ReadOnly Volumes (Replication)

Replication is the creation of a clone of the ReadWrite volume and the distribution of the clone to one or more sites. Replication increases system efficiency and boosts data availability by making a volume's contents accessible on more than one file server machine.

A replicated volume takes up as much disk space as its source ReadWrite volume unless it is placed on the same partition as its source. That is why it is recommended that you place a ReadOnly volume at the same site as the ReadWrite source. This ReadOnly clone provides all of the benefits of a stand-alone ReadOnly volume but takes up less disk space.

5.5.1. The Types of Volumes to Replicate

You will probably want to replicate only volumes containing files and directories that satisfy the following criteria:

- The files/directories are heavily used. For example, binary files for text editors and other popular application programs and heavily traversed directory paths like the paths leading to user home directories.
- The files/directories change infrequently. The reason for this criterion is that the contents of ReadOnly volumes cannot change and they should match their ReadWrite source at all times if system performance is to be predictable. Therefore, when the ReadWrite source changes, you will have to replicate it again to replace the old ReadOnly volumes.

5.5.2. The Two Stages of AFS Replication

In AFS, replicating a volume is a two stage process:

Stage 1: You define the site(s) where the ReadOnly copies will reside. Use the **vos addsite** command, which alters the site definition field in the VLDB entry for the ReadWrite source volume. This stage corresponds to step 6 in the instructions below.

Stage 2: You clone the ReadWrite source volume and release the clone to the defined sites using the **vos release** command. This stage corresponds to step 8 in the instructions below.

5.5.3. Determining if Replication Is Successful

It is important to know if replication (Stage 2 in particular) is completely successful or not because all copies of the ReadOnly version must be identical to one another and to their ReadWrite source. The point of replicating a volume is to allow the Cache Manager to access the copy of the volume that is easiest to reach at a given time; and the user should not have to worry that the data retrieved might vary depending on which copy the Cache Manager reaches.

To guarantee that all the ReadOnly copies match each other and the ReadWrite version, Stage 2 must be "all-or-nothing" — either all ReadOnly sites should receive the new version or they all should keep the version they already have. If an error occurs during the execution of **vos release** (corresponding to Stage 2), an error message tells you which of the defined ReadOnly sites did not receive the new version.

You can also determine if a release is successful by checking for the presence of two flags — `Old release` and `New release` —in the VLDB entry. The Volume Server and VL Server set these flags for their own use as they execute sub-steps in Stage 2, but the flags can serve as a useful failure signal. The Volume Server and VL Server set the flag as follows (these substages are internal and do not correspond to any action on your part after you issue the **vos release** command):

Stage 2.1: Before cloning begins, the VL Server marks each ReadOnly site in the VLDB entry with an `Old release` flag, and the ReadWrite site with a `New release` flag. (Recall that you already defined the ReadOnly sites in Stage 1 with the **vos addsite** command.)

Stage 2.2: The Volume Server clones the ReadWrite version of the volume. The clone is called the "ReleaseClone" version of the volume and is assigned its own volume ID number temporarily. The VL Server records that number in the VLDB entry, as the "ReleaseClone ID" and the Volume Server records it in the volume header as the "RClone." (Recording this ID allows re-use of the ReleaseClone if necessary; see the explanation of the **-f** flag that appears below.)

Stage 2.3: As soon as each ReadOnly site successfully receives the ReleaseClone, the VL Server changes the site's site flag in the VLDB from `Old release` to `New release`. The ReleaseClone takes on the regular ReadOnly volume ID as soon as it reaches the site.

Stage 2.4: After all the ReadOnly sites have the new clone, the VL Server removes the `New release` flag from all of the ReadOnly versions and from the ReadWrite version. Because the release was successful, the ReleaseClone is no longer needed and the Volume Server deletes it from the system and from the VLDB.

The presence of `New release` or `Old release` flags or the ReleaseClone ID in the VLDB after the completion of Stage 2 indicates that the release was not successful. You must fix the problem; otherwise, it is quite possible that Cache Managers will access different versions of a volume on different file server machines.

The best solution if a release is incomplete is to continue issuing the **vos release** command until the `Old release` and `New release` flags and the ReleaseClone ID no longer appear in the VLDB entry. You can examine the VLDB entry with the **vos examine** command (described in step 2 of the instructions below).

5.5.4. Using the -f Flag to Force Creation of a New Clone

By default, the Volume Server does not try to make a new clone every time you issue the **vos release** command. Instead, its behavior depends on whether or not all of the ReadOnly sites have identical copies of the volume:

- If all the ReadOnly sites currently have the same version, the Volume Server assumes that the previous **vos release** command must have completed successfully, and that you have defined no new ReadOnly sites since. The Volume Server makes a new clone and distributes it to all the defined sites.

- If all the ReadOnly sites do not have the same copy, the Volume Server concludes that the previous release must have failed in the middle of the release process. To save time, the Volume Server does not make an entirely new clone before attempting another release. Instead, it locates and distributes the ReleaseClone version of the volume, which still exists because the previous release was unsuccessful (see the above description of sub-stages 2.2 through 2.4 for more on the ReleaseClone). Distributing the ReleaseClone to the sites that do not yet have it will make all of their copies identical again, without the expense of making a brand new clone.

You may want to guarantee that the Volume Server makes a brand new clone no matter what the status of copies at the ReadOnly sites. To do this, include the **-f** flag with **vos release**.

5.5.5. Variations on Replication

Because replication is a two stage process, you can choose to perform only some of the steps depending upon the current situation:

- If you are replicating a ReadWrite source volume for the first time, you should follow all the instructions below (steps 1 through 10).
- If you are releasing a new clone to sites that already exist, you may skip steps 3 through 6. You will probably want to execute step 2 to verify that the sites you want are already defined.
- If you wish to add one or more new sites to the existing ones, follow all the instructions (steps 1 through 10). In step 6, you need to issue **vos addsite** for the new sites only.
- If you want to define sites but want to wait to release a clone, simply stop after step 6 and continue when you are ready.
- If you want to remove one or more sites before releasing a new clone to the remaining sites, follow the instructions in section 5.13 for site removal and then start with step 7.

5.5.6. Using ReadOnly Volumes

When the Cache Manager encounters the mount point for a ReadWrite volume, it contacts the VL Server to learn where the volume resides. If the mount point is regular and resides in a ReadOnly volume, the Cache Manager also checks if the volume exists in a ReadOnly version. If it does, the Cache Manager attempts to access any of the available ReadOnly copies of that volume. This is referred to as "going down a ReadOnly path." Using files on ReadOnly volumes puts less of a load on the File Server because only one callback is required per ReadOnly volume as opposed to a callback per file for ReadWrite volumes. The Cache Manager gets on a ReadOnly path by default because it is biased toward accessing the ReadOnly version of `root.afs` if it is available. (See section 5.7.1 for more information about the rules that the Cache Manager follows when accessing volumes.)

Explicitly mounting a ReadOnly volume (creating a mount point that indicates the ReadOnly rather than the ReadWrite version of a volume) is not generally necessary nor is it recommended. The Cache Manager will "find" the ReadOnly version of a volume even though it is not itself mounted.

You should place a ReadOnly copy of a volume at the same site where the ReadWrite volume resides because the ReadWrite volume will not be used as a substitute for ReadOnly volumes should other ReadOnly volumes become unavailable. Furthermore, a ReadOnly copy located at the same site as the ReadWrite volume does not require as much actual disk space as the ReadWrite volume because the two volumes will share inodes when possible. (The ReadOnly volume could potentially require as much disk space as the ReadWrite volume, if the latter changes drastically.)

Only one ReadOnly volume can be placed on any single server.

⇒ To replicate a ReadWrite volume (create a ReadOnly volume)

Step 1: Verify that you have the privilege necessary for creating volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Learn where the ReadWrite version and each ReadOnly version of the volume reside:

% **vos examine** <*volume name or ID*>

where

e is an acceptable abbreviation of **examine**.

volume name or ID
identifies the ReadWrite version of the volume, either by name or volume ID number.

The last lines in the output display the volume’s current sites. Section 5.8.3 shows an example of the output.

Step 3: Determine if your cell has designated one or more file server machines as "ReadOnly" servers. This will probably be true only if your cell is quite large.

Step 4: Determine how big the ReadOnly version of the volume will be; the partition you choose must have at least that much empty space (unless it contains the ReadWrite source, in which case it needs much less). The first line of output from the **vos examine** command shows the size in kilobyte blocks of a volume; section 5.8.3 shows an example of the output. You can learn how much space is available on a file server machine’s partitions (and their total size) with **vos partinfo**.

% **vos partinfo** <*machine name*> **[**<*partition name*>**]**

where

p is an acceptable abbreviation of **partinfo**.

machine name specifies the file server machine about which you want partition information.

partition name is optional. If you provide it, the output displays the size and space available on that one partition. If you omit it, the output displays the size and space available for all partitions on *machine name*.

Note: The total partition size reported in this command may not agree with the same figure in the output of the standard UNIX **df** command. The **df** total size includes some reserved space that does not show up in this report, and so is likely to be about 10% larger.

Step 5: Select the new ReadOnly site(s), based on what you learned in steps 2, 3, and 4.

Step 6: Define each new ReadOnly site in the VLDB. **Repeat this command for each site.**

% **vos addsite** <*machine name*> <*partition name*> <*volume name or ID*>

where

ad is an acceptable abbreviation of **addsite**.

machine name defines the file server machine for the new site.

partition name specifies the disk partition of *machine name* for the new site.

volume name or ID
identifies the ReadWrite volume that is to be replicated, either by its complete name or its volume ID number.

Step 7: **(Optional).** Verify that the *fs* and *vlserver* processes are functioning normally on each file server machine where you have defined a ReadOnly site. The *fs* process includes the Volume Server and *vlserver* is the VL Server. Knowing that they are functioning eliminates two possible sources of failure for the release. Issue **bos status**, described fully in section 4.3, on each file server machine housing a ReadOnly site for this volume.

% **bos status** <*machine name*> **fs vlserver**

Step 8: Clone the ReadWrite source volume and release the clone to the ReadOnly site(s).

% **vos release** <*volume name or ID*> **[-f]**

where

rel is an acceptable abbreviation of **release**.

volume name or ID
identifies the ReadWrite volume to clone, either by its complete name or volume ID number. The ReadOnly version will take the same name with a `.readonly` extension. All copies of the ReadOnly version share the same ReadOnly volume ID number.

-f should be included only if you want to force the Volume Server to make and release a brand new clone. See section 5.5.4 for a complete explanation of this flag.

Step 9: **(Optional).** Verify that no `Old release` or `New release` flags are present in the VLDB entry. If anything went wrong with the release (perhaps because a machine or process was down or a site inaccessible), you should have seen an error message. If you want to be sure that the release was successful, the **vos listvldb** command, described completely in section 5.8.1, reveals the presence of `Old release` and `New release` flags.

% **vos listvldb** <*volume name or ID*>

Step 10: If any `Old release` or `New release` flags appear, repeat steps 7 through 9 until the flags disappear. Do not attempt releases when you know that any of the ReadOnly sites are inaccessible (either because of file server machine or server process outage).

5.6. Creating Backup Volumes

As explained in section 5.2.2, ***backing up*** a volume refers to creating a clone of the ReadWrite version of a volume and placing it at the same site as the ReadWrite version. Creating a Backup version of a volume has two purposes:

- It is the first step in backing up a volume permanently on tape. Dumping the Backup version of a volume does not cause the volume's contents to become unavailable during the dump, as dumping the ReadWrite version would. See chapter 6 for more details.
- It enables users to restore mistakenly deleted or changed data themselves (if the Backup version of the volume is mounted), freeing you for more crucial tasks. The Backup version captures the state of its ReadWrite source at the time the Backup is made, and its contents cannot change. Users can therefore restore a file to its state at the time you made the Backup. Further suggestions on using Backup volumes in this way appear below.

Note: Creating a Backup volume is not the same as permanent storage of the data in that volume on tape; however, making a Backup volume is one step in the latter task. See chapter 6 to learn about the AFS Backup System.

5.6.1. Backing Up Multiple Volumes at Once

The **vos backupsys** command allows you to backup multiple ReadWrite volumes at once. This command is useful when preparing for large-scale backups to tape (see chapter 6 for information about the AFS Backup System). When using **vos backupsys**, you determine which volumes will backup by combining the command's **-prefix**, **-server**, and **-partition** arguments and the **-exclude** flag in different ways. To create a Backup version of:

- Every ReadWrite volume in the system, omit all arguments and flags. This could take a rather long time to execute, depending on the number of volumes.
- Every ReadWrite volume whose name begins with a certain character string (for example, `sys.` or `user.`), regardless of site, use **-prefix**.
- Every ReadWrite volume on a file server machine, specify the file server name with **-server**.
- Every ReadWrite volume that resides on a partition of the same name (for instance, on */vicepa* on any file server machine), specify the partition name with **-partition**.
- Every ReadWrite volume on a certain partition of a file server machine, specify both **-server** and **-partition**.
- Every ReadWrite volume with a specific prefix that resides on a file server machine, combine **-prefix** and **-server**;
- Every ReadWrite volume with a specific prefix that resides on a specific server and partition, combine **-prefix**, **-server**, and **-partition**.
- Every ReadWrite volume that resides on a file server machine (<u>except</u> those with a specific prefix), combine **-prefix** and **-server** and attach the **-exclude** flag.

- Every ReadWrite volume that resides on a specific server and partition (except those with a specific prefix), combine **-prefix**, **-server**, and **-partition** and attach the **-exclude** flag.

To backup a single volume, use **vos backup**, which employs a more streamlined technique for finding a single volume.

5.6.2. Automating Creation of Backup Volumes

Most cells find that it is best to make a new Backup version of each volume every day at the same time. A time when usage is low is preferable, because making the Backup causes the ReadWrite volume to be unavailable momentarily.

You can either issue the necessary **vos backupsys** or **vos backup** commands at the console or create a "cron" entry in *BosConfig* on each file server where you want to back up volumes. Recall from section 4.2.2 that a cron process is one that executes only at the time you specify. The advantage of using a cron process is that the backups can be made during a low-usage period without requiring a system administrator to be present to issue commands.

The following example command would create a cron process called "*backupusers*" in *BosConfig* on the machine *fs3.transarc.com*. The process would execute every day at 1:00 a.m., making a Backup version of every volume in the file system whose name started with "`user`". The **-localauth** flag is necessary so that the process, which runs unauthenticated, is allowed to execute the privileged **vos backupsys** command. Note that it is necessary to provide the complete pathname of the command to be executed.

```
% bos create fs3.transarc.com backupusers cron
      -cmd "/usr/afs/bin/vos backupsys  -prefix user -localauth"
      "1:00"
```

5.6.3. Making the Contents of Backup Volumes Available to Users

Unlike ReadOnly volumes, it does make sense to mount Backup volumes to make their contents available to users. The most sensible place to mount the Backup version of a user volume is at a subdirectory of the user's home directory (step 4 below explains how to mount a Backup volume). By making each user's Backup volume available in this way, the user can restore files accidentally deleted or changed since the last Backup was made, without having to request help from you.

Suitable names for this directory might be *OldFiles* or *Backup*. The subdirectory you create will look just like the user's own home directory, with all file and subdirectories in the same relative positions.

If you do create and mount Backup volumes for your users, you will need to make them aware of these volumes. The *AFS User's Guide* does not mention Backup volumes because making them available to users is optional. You should explain to users how often you will 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 copy it into their regular (ReadWrite) volume and modify it there. (They can use the standard UNIX **cp** command to copy the file from the *OldFiles* subdirectory into their regular home directory.) You might also want to reassure users that the data in their Backup volumes does not count against their ReadWrite volume quota.

⇒ To create a single Backup volume and mount it in the file system

Step 1: Verify that you have the privilege necessary for creating volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the INSERT and ADMINISTER access rights for the directory in which you wish to mount the volume. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the INSERT right if necessary.

Step 3: Create a Backup version of the desired ReadWrite source volume.

% **vos backup** <*volume name or ID*>

where

backup must be typed in full.

volume name or ID
identifies the ReadWrite volume to backup, either by its complete name or volume ID number. The Backup volume will take the same name, with an additional `.backup` extension. It has its own volume ID number.

Step 4: **(Optional).** Mount the Backup volume. While this step is optional, the contents of the volume will be inaccessible if you do not mount it.

% **fs mkmount** <*directory*> <*volume name*>**.backup**

where

mk is an acceptable abbreviation of **mkmount**.

directory names the mount point you wish to create. Do not create a file or directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory. For the Backup version of user volumes, the suggested name is *OldFiles* and the suggested place is the user’s home directory.

*volume name***.backup**
is the full name of the Backup volume.

Step 5: **(Optional).** Verify that you have created the mount point correctly by issuing the **fs lsmount** command, described fully in section 5.7.

% **fs lsmount** <*directory*>

⇒ To create multiple Backup volumes at once

Step 1: Verify that you have the privilege necessary for creating volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Create a Backup version of every ReadWrite volume that shares the same prefix or site. The effects of combining the three arguments are described in section 5.6.1.

% **vos backupsys [-prefix** <*common prefix on volume(s)*>**]**
[-server <*machine name*>**]**
[-partition <*partition name*>**] [-exclude]**

where

backups is an acceptable abbreviation of **backupsys**.

common prefix on volume(s)
defines the initial string in the name of each ReadWrite volume you wish to backup. May be combined with *machine name* and/or *partition name* and the **-exclude** flag.

machine name specifies the file server machine housing the volumes you wish to backup. May be combined with *prefix* and/or *partition name*.

partition name specifies the partition housing the volumes you wish to backup. May be combined with *prefix* and/or *machine name*.

-exclude indicates that all volumes except those indicated with the **-prefix** argument are to be backed up. If this flag is used, the **-prefix** argument also must be used.

5.7. Mounting Volumes (Creating, Removing, and Listing Mount Points)

Section 5.2.6 explains why it is necessary to mount a volume, and what the Cache Manager does when it encounters a mount point during pathname interpretation. This section provides additional information on mount points, including information on the rules of mount point traversal, a discussion of the three types of mount points, their purposes, and how to distinguish between them, and instructions for creating, removing, and examining mount points.

5.7.1. The Rules of Mount Point Traversal

This section contains information on the rules that the Cache Manager follows as it traverses the AFS mount points. There are three basic rules:

5.7.1.1. Rule 1: Access Backup and ReadOnly 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 will never access the ReadWrite version of a volume if the mount point names the Backup version. If the specified version is inaccessible, the Cache Manager reports an error.

5.7.1.2. Rule 2: Follow the ReadOnly Path When Possible

The Cache Manager has a built-in bias toward accessing a ReadOnly copy of the `root.afs` volume which corresponds to the top-level */afs* directory. If a ReadOnly copy of `root.afs` is available, it will be accessed in preference to the ReadWrite volume. When the Cache Manager encounters a mount point located in a ReadOnly volume, it attempts to access a ReadOnly copy of the volume that the mount point references; if the referenced volume is not replicated, it accesses the ReadWrite volume. The Cache Manager is thus forced onto a "ReadOnly path" because while on it the Cache Manager accesses ReadOnly versions when available, and accesses the ReadWrite volume only when no ReadOnly copies exist, increasing system performance and improving file availability.

Note: You must replicate both `root.afs` and `root.cell` (corresponding to */afs* and */afs/***cellname**, respectively) to ensure the Cache Manager's access to the other replicated volumes in your cell.

5.7.1.3. Rule 3: Once on a ReadWrite Path, Stay There

When the Cache Manager encounters a mount point in a ReadWrite volume, the Cache Manager only attempts to access a ReadWrite volume referenced by that mount point; failing this, the access attempt fails. While the Cache Manager traverses ReadWrite volumes, it is said to be following the "ReadWrite path;" once on a ReadWrite path, the Cache Manager cannot return to a ReadOnly path. (Exception: see the discussion on cellular mount points in section 5.7.2.1.)

5.7.2. The Three Types of Mount Points

There are three types of mount points, each of which is appropriate for different purposes and each of which will follow the rules of mount point traversal to a different extent:

- ***regular*** — As long as the Cache manager is traversing regular mount points, the Cache Manager follows all rules of mount point traversal.
- ***ReadWrite*** — When the Cache Manage encounters a ReadWrite mount point, the Cache Manager always accesses the ReadWrite version of the indicated volume; thereafter, it remains on the ReadWrite path. (Of course, if a mount point specifies a `.readonly` or `.backup`, the Cache Manager will access that version of the volume.)
- ***cellular*** — A cellular mount point tells the Cache Manager that it has crossed cell boundaries. When the Cache Manager crosses a cellular mount point, it is automatically put on a ReadOnly path regardless of the path it was on prior to crossing the mount point.

5.7.2.1. The Purposes of the Three Types of Mount Points

Regular Mount Points

The vast majority of your mount points should be **regular** mount points without any extension. Using regular mount points as much as possible promotes the most efficient use of both replicated and nonreplicated volumes and, assuming your volumes are distributed in a balanced manner, the most efficient use of your system resources.

This is because there are normally multiple ReadOnly copies of a volume but only one ReadWrite version. It makes sense to access one of the more numerous ReadOnly copies when available, rather than a single ReadWrite version. And since a regular mount point is not solely a "ReadOnly" mount point, the Cache Manager may still access the ReadWrite volume when it is on a ReadWrite path or it encounters a regular mount point and no ReadOnly copies of the referenced volume are available.

To create a regular mount point, use the **fs mkmount** command.

Note: You must replicate all volumes that the Cache Manager will access while traversing the path to a replicated volume; otherwise, the Cache Manager will never access it because it will have already switched to a ReadWrite path.

ReadWrite Mount Points

By convention, you should use only one **ReadWrite** mount point in your file tree: to mount your cell's `root.cell` volume at the second level just below */afs*. If you do this, it is highly recommended that you also mount `root.cell` with a regular mount point at the same level. The two mount points are typically distinguished by the placement of a period at the start of the ReadWrite mount point's name.

The existence of a ReadWrite mount point in the file tree—the one for `root.cell`—is sufficient to put the Cache Manager on a ReadWrite path all the way down the pathnames in your cell when a user needs to access ReadWrite volumes specifically. Meanwhile, the existence of a regular mount point for `root.cell` allows the Cache Manager to follow the ReadOnly path when possible.

Mounting a ReadOnly or Backup volume with a ReadWrite mount point is possible, but unnecessary. If the mount point references one of those versions explicitly, then the Cache Manager accesses it explicitly, ignoring whether their mount point is regular or ReadWrite. (See **Rule 1**.)

You designate a mount point as ReadWrite by using the **-rw** flag on the **fs mkmount** command. By convention, you also put a period at the front of the mount point name. See the instructions in the following section.

Cellular Mount Points

A **cellular** mount point indicates to the Cache Manager that the referenced volume resides in a specified foreign cell. If a mount point is not cellular, then the Cache Manager assumes that the referenced volume resides in the same cell as the mount point does.

Normally, you will use cellular mount points only at the second level in your cell's file tree, just below */afs*, to mount the `root.cell` volumes of foreign cells that you wish to make visible to your cell's users. It is possible to create cellular mount points (mount foreign volumes) at other levels in the tree; however, doing so is not recommended as it is easy for users to become confused if the Cache Manager switches from cell to cell at different points in a pathname.

Cellular mount points can be either regular or ReadWrite:

- A regular cellular mount point not only tells the Cache Manager to cross into a foreign cell, but also to access the ReadOnly version of the indicated volume if possible. The advantage is that the Cache Manager traverses a "ReadOnly path" in the foreign cell, even if the mount point to the indicated volume resides in a ReadWrite volume in the local cell.

  To create a regular cellular mount point, you use the **-cell** argument to specify the cell name.

- A ReadWrite cellular mount point tells the Cache Manager to cross into a foreign cell and access the ReadWrite version of the volume (assuming that the volume does not have a `.backup` or `.readonly` extension). Using this type of mount point is discouraged, because accessing ReadWrite volumes means the File Server must issue many callbacks, an extra load to impose on the servers of a foreign cell. In general, only a cell's own administrators need to access the ReadWrite versions of replicated volumes.

  To create a ReadWrite cellular mount point, use the **-cell** argument to specify the cell name and add the **-rw** flag. Because this is not recommended, no example of it appears below.

5.7.2.2. Distinguishing between Regular, ReadWrite, and Cellular Mount Points

You can distinguish regular and ReadWrite mount points from one another in two ways:

- When you issue the **fs lsmount** command on a regular mount point, a pound sign (#) appears in front of the volume name. A ReadWrite mount point will have a percent sign (%) instead.
- By convention, a period should be placed in front of the directory (mount point) name when creating a ReadWrite mount point, to distinguish it from the regular mount point to the same volume. An advantage of preceding ReadWrite mount points with a period is that it makes them invisible when regular users do a plain **ls** of a directory that contains them. Users do not usually need to access the ReadWrite version of a replicated volume, and seeing ReadWrite mount points may confuse them.

If a mount point is cellular, then the cell name and a colon will appear just in front of the volume name in the output of **fs lsmount**. (The pound sign or percent sign also appears, with the same meaning as for non-cellular mount points.)

The following examples show how these indicators work together:

% **fs lsmount /afs/transarc.com/usr/terry**
```
'/afs/transarc.com/usr/terry' is a mount point for
volume '#user.terry'
```

% **fs lsmount /afs/.transarc.com**
```
'/afs/.transarc.com' is a mount point for volume
'%transarc.com:root.cell'
```

% **fs lsmount /afs/transarc.com**
```
'/afs/transarc.com' is a mount point for volume
'#transarc.com:root.cell'
```

The first mount point is a regular, non-cellular mount point; the second mount point is a ReadWrite, cellular mount point; the third mount point is a regular, cellular mount point. The volume `root.cell` is mounted in both ways so that when users want to make sure that they access the ReadWrite version of `root.cell`, they specify */afs/.transarc.com*; if they do not care which version of the volume they access, they specify */afs/transarc.com* and rely on the mount point traversal rules to get them to the appropriate version (in this case the ReadOnly version).

5.7.3. Creating, Removing, and Listing Mount Points

⇒ To create a regular mount point

Step 1: Verify that you have the INSERT and ADMINISTER access rights for the directory where you wish to mount the volume. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

Step 2: Mount the volume.

% **fs mkmount** <*directory*> <*volume name*>

where

mk	is an acceptable abbreviation for **mkmount**.
directory	names the mount point you wish to create. Do not create a file or directory with the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory.
volume name	is the full name of the volume. If it is a Backup volume, be sure to include the `.backup` extension.

⇒ To create a ReadWrite mount point

Step 1: Verify that you have the INSERT and ADMINISTER access rights for the directory in which you wish to mount the volume. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

Step 2: Mount the volume.

% **fs mkmount .**<*directory*> <*volume name*> **-rw**

where

mk	is an acceptable abbreviation for **mkmount**.
directory	names the mount point you wish to create. Do not create a file or directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory. Put a period in front of the name to distinguish this ReadWrite mount point from the regular mount point to the same volume.
volume name	is the full name of the volume.
-rw	indicates that the mount point is ReadWrite. See section 5.7.1.3 for an explanation.

⇒ To create a cellular mount point

Step 1: Verify that you have the INSERT and ADMINISTER access rights for the directory in which you wish to mount the volume. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

Step 2: Mount the volume.

% **fs mkmount** <*directory*> <*volume name*> <*cell name*>

where

mk	is an acceptable abbreviation for **mkmount**.
directory	names the mount point you wish to create. Do not create a file or directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory.
volume name	is the full name of the volume. Normally, this will be `root.cell`.
cell name	names the cell in which the volume resides.

⇒ To remove a mount point

Step 1: Verify that you have the DELETE access right for the directory from which you wish to remove the mount point. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the DELETE right if necessary.

Step 2: Remove the mount point. The volume still exists, but its contents are inaccessible unless you create another mount point.

% **fs rmmount** <*directory*>

where

rm	is an acceptable abbreviation of **rmmount**.
directory	names the mount point/directory to be removed. Provide a pathname unless *directory* is a subdirectory of the working directory.

⇒ To examine a mount point

Step 1: At the command shell prompt (%), type

% **fs lsmount** <*directory*>

where

ls	is an acceptable abbreviation of **lsmount**.
directory	names the mount point (directory name) you wish to examine.

Examples of the output from this command appear in the introduction to this section. If *directory* is not a mount point, an error message will tell you so. In that case, you may wish to examine *directory*'s parent directory to discover if it is a mount point.

5.7.4. Creating Mount Points in a Foreign Cell

Creating a mount point in foreign cells is accomplished in the same way as creating a mount point in the local cell.

5.8. Listing Information About Volumes

This section contains instructions for using commands to access information about volumes. Some of these commands allow you to examine information from the VLDB, the volume header, or both; these commands require that you know the volume's name or volume ID. Other commands provide volume information even if you only know the name of a file or directory in the volume. Though this information is more limited in scope, it does provide you with the information to map file and directory names to volume names and ID numbers.

Section 5.12 contains instructions for listing a volume's quota.

5.8.1. Examining the VLDB Entry for One or More Volumes

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. This process can take a very long time, depending on the number of entries.
- To display every VLDB entry that mentions a specific file server machine as the site of a ReadWrite or ReadOnly version of a volume, specify the machine's name with **-server**.
- To display every VLDB entry that mentions a certain partition on any file server machine as the site of a ReadWrite or ReadOnly version of a volume, specify the partition name with **-partition**.
- To display every VLDB entry that mentions a certain partition on a certain file server machine as the site of a ReadWrite or ReadOnly version of a volume, combine **-server** and **-partition**.
- To display a single VLDB entry, specify a volume name or ID number with *volume name or ID*.
- To display the VLDB entry only for volumes with locked VLDB entries, use the **-locked** flag with any of the site definitions mentioned just above.

⇒ To list information from the VLDB

Step 1: At the command shell prompt (%), type

% **vos listvldb [**<*volume name or ID*>**] [-server** <*machine name*>**]**
[-partition <*partition name*>**] [-locked]**

where

listvl is an acceptable abbreviation of **listvldb**.

volume name or ID
identifies one volume either by its complete name or volume ID number. Do <u>not</u> combine this argument with the others listed here.

machine name specifies a file server machine. You must include the **-server** switch. Provide this argument alone or with *partition name*.

partition name specifies a partition. You must include the **-partition** switch. Provide this argument alone or with *machine name*.

-locked indicates that VLDB information should be displayed only for those volumes that have locked VLDB entries. Combine this flag with any of the other arguments.

The output of the command includes:

- A LOCKED indicator if the VLDB entry is locked. Otherwise the top line is blank (the normal state).
- The volume's name.
- The volume ID of the ReadWrite (RWrite) version. A *0* indicates that no ReadWrite version exists.
- The volume ID of the ReadOnly (ROnly) version. A *0* indicates that no ReadOnly versions exist.
- The volume ID of the Backup version. A *0* indicates that no Backup version exists.
- The volume ID of the ReleaseClone (RClone) version, if one currently exists. The existence of a ReleaseClone ID indicates that the last release was not successful. See section 5.5.
- The number of sites at which ReadWrite and ReadOnly versions of the volume exist.
- A site definition (file server machine/partition) for the ReadWrite (`RW`) and any ReadOnly (`RO`) versions of the volume. The presence of `New release` and `Old release` flags by the site definitions indicate a failed release. See section 5.5.

An example of this command and its output:

`%` **vos listvldb user.terry**

```
user.terry
    RWrite: 50489902    ROnly        0   Backup: 50489904
    number of sites -> 1
       server fs3.transarc.com partition /vicepa RW Site
```

5.8.2. Examining the Volume Header for One or More Volumes

The **vos listvol** command displays the volume header for every volume on a file server machine or for a specified partition on that file server machine. To get this information, **vos listvol** contacts the Volume Server on the specified machine. You can control the amount of information displayed by including the **-fast**, the **-long**, or the **-extended** flag; see the definition and examples below.

To see the volume header of one volume only, use the **vos examine** command, described in section 5.8.3.

⇒ To examine the volume header for all volumes on a machine/partition

Step 1: At the command shell prompt (`%`), type

`%` **vos listvol** <*machine name*> **[**<*partition name*>**]**
[-fast] [-long] [-extended]

where

listvo is an acceptable abbreviation of **listvol**.

machine name specifies the file server machine for which volume headers should be displayed. Provide this argument alone or with *partition name*.

partition name specifies one partition on *machine name*. Provide this argument with *machine name* if at all.

-fast indicates that the output should display volume ID numbers only. Include this flag, **-long**, or **-extended**.

-long indicates that the output should display additional information not available by default. See the output description and example below. Include this flag, **-fast**, or **-extended**.

-extended indicates that the output should display all information available, including read and write statistics not included when the **-long** flag is specified.

Output with* -fast *Flag

When you use the **-fast** flag, the output displays only the volume ID number of each volume, and reports the total number of volumes in the listing.

Output with No Flags

When you do not use the **-fast**, the **-long**, or the **-extended** flag, the information includes the volume's

- Name.
- Volume ID number.
- Type (`RW` for ReadWrite, `RO` for ReadOnly, and `BK` for Backup).
- Size in kilobytes (`1024` equals one megabyte).
- Status at the server. Possible values include the following:
  - `On-line`: the volume is fully accessible through the file system
  - `needs salvage`: the volume is likely to be corrupted. Run the Salvager as described in section 5.11.
  - `Off-line`: the volume is not accessible, but there is no reason to suspect that it is corrupted. Run the Salvager as described in section 5.11. Also see section 5.2.4 for more on this flag.

The first line of the output tells how many volumes appear in the listing, and the last line how many of those are on-line, off-line, and busy.

Output with* -long *Flag

When you include the **-long** flag, the output gives additional information:

- The volume's site (file server machine and partition).
- The volume ID numbers associated with the volume. One of them should match the volume ID that appears on the first line of the listing:
  - `RWrite`: the volume ID of the ReadWrite version of this volume; a *0* indicates that no ReadWrite version exists.
  - `ROnly`: the volume ID of the ReadOnly version of this volume; a *0* indicates that no ReadOnly versions exist.
  - `Backup`: the volume ID of the Backup version of this volume; a *0* indicates that no Backup version exists.
  - `RClone`: the volume ID of the ReleaseClone version of this volume; if one currently exists. The existence of a ReleaseClone ID indicates that the last release was not successful. See section 5.5.
- The maximum space quota allotted to the ReadWrite version of the volume, in kilobytes.
- The date when the volume was created.
- The date when the contents of the volume last changed. For ReadOnly and Backup versions of a volume, this should match the creation date.

Output with* -extended *Flag

When you include the **-extended** flag, the output contains the following two statistical tables entitled *Raw Read/Write Stats* and *Writes Affecting Authorship*. These tables provide the following information:

- The *Raw Read/Write Stats* table contains information on the number of reads (fetches) and writes (stores) made on the specified *volume name or id*. This information is categorized by reads and writes initiated by a client on the same network as the server(Same Network) versus those initiated from a client on a different network than the server different network (Diff Network). Both of these categories are further broken down into all reads and writes versus those initiated by authenticated users and processes.

  Note: Diff Network versus Same Network is determined by comparing the network portion of the IP address based on the class of network (A, B, or C). If the network portion of the IP addresses for the client and server match, it is the same network; otherwise, it is considered a different network.

- The *Writes Affecting Authorship* table contains information on writes made to files and directories in the specified *volume name or id*. The table categorizes the data to compare writes to files (File Authorship) and writes to directories (Directory Authorship). Both of these categories are further broken down to compare writes made by the owner of the files or directories (Same), and writes made by users/processes that were not the owners of the files or directories (Diff). The time divisions indicate the time between the recorded write and the previous update to the affected file (or directory).

An example of the output when the **-fast** flag is included:

% vos listvol fs3.transarc.com /vicepa -f

```
Total number of volumes on server fs3.transarc.com
   partition /vicepa: 37
50489902
50489904
   .
   .
35970325
49732810
```

An example of the output when no flags are included:

% vos listvol fs2.transarc.com b

```
Total number of volumes on server fs2.transarc.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
```

An example of the output when the **-long** flag is included:

% vos listvol fs2.transarc.com b -long

```
Total number of volumes on server fs2.transarc.com
                                    partition /vicepb: 66
      .                   .      .         .   .    .
      .                   .      .         .   .    .

user.pat             1969534536 RW      17518 K On-line
     fs2.transarc.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.transarc.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
```

An example of the output when the **-extended** flag is included:

```
common.bboards              1969535592 RW    23149 K used 9401 files On-line
    ernie.transarc.com /vicepa
    RWrite 1969535592 ROnly          0 Backup 1969535594
    MaxQuota      30000 K
    Creation    Wed Mar  7 14:26:05 1990
    Last Update Mon Jan  4 09:20:43 1993
    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 |
          |-------------------------------------------|
```

5.8.3. Examining Both the VLDB Entry and Volume Header for a Volume

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.

Note: Because this command retrieves information from the volume header, you need to specify which version of the volume you are interested in (that is, add the `.readonly` or `.backup` extension if necessary). The information from the VLDB will be the same no matter which version you look at. An error message results if the ReadWrite version no longer exists and you forget to specify the `.readonly` or `.backup` extension.

⇒ To get complete information about one volume at a time

Step 1: At the command shell prompt (%), type

% **vos examine** <*volume name or ID*>

where

e is an acceptable abbreviation of **examine**.

volume name or ID
identifies one volume either by its complete name or volume ID number. It may be a ReadWrite, ReadOnly, or Backup type. Use the `.backup` or `.readonly` extension if appropriate.

The top part of the output lists the same information from a volume header as when you issue **vos listvol** with the **-long** flag. The bottom part of the output lists the same information from the VLDB as when you issue **vos listvldb**. See sections 5.8.1 and 5.8.2, respectively.

Below is an example for a volume whose VLDB entry is currently locked. If it were not locked, the "`LOCKED`" line would not appear.

% **vos examine user.terry**

```
user.terry                    536870981 RW   3459 K On-line
     fs3.transarc.com /vicepa
     RWrite 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)
        LOCKED
     RWrite: 5360870981   Backup: 536870983
     number of sites -> 1
        server fs3.transarc.com partition /vicepa RW Site
```

5.8.4. Translating between Volume Names and IDs, Directory Names, and Locations

The commands listed below are used to help you obtain information about a volume when you know only a few things about it, such as the name of a file in it, or its volume ID number.

⇒ To learn a volume's name, given a directory or file name

Step 1: Issue the **fs listquota** command.

% **fs listquota [**<*dir/file path*>**]**

where

lq is an acceptable alias for **listquota**.

dir/file path names the directory or file in the unknown volume. Omit this argument to use the current working directory.

An example of the output:

% **fs listquota /afs/transarc.com/usr/terry**

```
Volume Name     Quota    Used    % Used   Partition
user.terry      15000    5071       34%         86%
```

⇒ To learn a volume's ID number, given a directory or file name

Step 1: Issue the **fs examine** command.

% **fs examine [**<*dir/file path*>**]**

where

exa is an acceptable abbreviation of **examine**.

dir/file path names the directory or file in the unknown volume. Omit this argument to use the current working directory.

An example of the output from this command:

% **fs examine /afs/transarc.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 numbers that appear in the last line of this output may not always agree with the corresponding numbers in the output of the standard UNIX **df** command. The **df** output reflects the state of partitions exactly when the command is issued. The numbers in this command's output may be as much as five minutes old, as the Cache Manager polls the File Server for partition information at that frequency. In addition, the partition size reported by the UNIX **df** command includes some reserved space that does not show up in this report of partition size and so it is likely to be about 10% larger.

⇒ To learn all the volume ID numbers for a volume, given its name

Step 1: Issue the **vos examine** command. Section 5.8.3 fully describes the syntax and output of this command.

% **vos examine** <*volume name or ID*>

⇒ To learn a volume's ID number, given its location only

Step 1: Issue the **vos listvol** command, including the file server machine (and optionally, partition) that houses the volume. You must know the volume's name to know what to look for. Section 5.8.2 fully describes the syntax and output of this command. Be aware that this command lists every other volume at that site, too, which can take time.

% **vos listvol** <*machine name*> **[**<*partition name*>**]**

⇒ To learn a volume's location, given a directory or file name

Step 1: Learn the volume's name using **fs listquota**, as fully described above.

% **fs listquota [**<*dir/file path*>**]**

Step 2: Issue **vos listvldb** using the volume name you just learned. Section 5.8.1 fully describes the syntax and output of this command.

% **vos listvldb** <*volume name or ID*>

Note: If you only want to know which machine houses the volume and are unconcerned about the partition, the **fs whereis** command performs the translation in one step:

% **fs whereis [**<*dir/file path*>**]**

where

whe is an acceptable abbreviation of **whereis**.

dir/file path names the file or directory in each volume whose machine location you wish to learn. If this argument is omitted, the location of the current directory is displayed.

An example:

```
% fs whereis /afs/transarc.com/user/terry
File /afs/transarc.com/usr/terry is on host fs2.transarc.com
```

⇒ To learn a volume's location(s), given its name or volume ID number

Step 1: Issue the **vos listvldb** command, the output of which is described in section 5.8.1 above.

% **vos listvldb** <*volume name or ID*>

5.9. Moving Volumes

There are three main reasons to move volumes:

- A machine or partition needs to be removed for repair or replacement. In that case, you would want to move all of its volumes to other partitions/machines.
- A partition is becoming overcrowded. One symptom of overcrowding is when users are unable to save files even though they have not exceeded their own quota. The following error message confirms the problem:

  ```
  afs: failed to store file (partition full)
  ```

  You can track the amount of a partition's space in use by using the AFS monitoring program called **Scout**, other AFS commands, and UNIX commands. See Chapter 7 for more information on **Scout**.
- 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.

5.9.1. Moving the Three Types of Volumes

ReadWrite volumes are the only ones you can actually move, using the **vos move** command. It is not possible to move the Backup version of a volume. When you move a ReadWrite volume, the Backup version is deleted automatically. You are strongly encouraged to replace the deleted Backup version with a new one at the new site. To move a copy of the ReadOnly version of a volume from one site to another, you actually remove it from the current site (following the instructions in section 5.13) and then release a new clone of the ReadWrite source volume to the newly defined ReadOnly site (following the instructions in section 5.5).

Note: You cannot move a ReadWrite volume to a partition if that partition does not have enough space to house that volume. The **vos move** command will warn you if a partition does not have enough space to contain a volume using the following message:

`vos: no space on target partition` *dest part* `to move volume` *volume*

⇒ To move a ReadWrite volume

Step 1: Verify that you have the privilege necessary for moving volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Move the volume. Type the following command on a single line.

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 an acceptable abbreviation of **move**.

volume name or ID
identifies the ReadWrite volume to be moved, either by its complete name or its volume ID number.

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 the volume will move.

partition name on destination
names the partition to which the volume will move.

Note: Unless you have a very good reason for doing so, you should not interrupt a **vos move** command in progress. For more information, see the entry on **vos move** in the *AFS Command Reference Manual*.

Step 3: **(Optional).** Confirm that the move was successful using the **vos listvldb** command fully described in section 5.8.1.

% **vos listvldb** <*volume name or ID*>

Step 4: If a Backup version of the volume existed at the ReadWrite's previous site, create a new Backup at the new site (recall that moving the ReadWrite source automatically deletes the Backup version). Section 5.6 discusses **vos backup** in detail.

% **vos backup** <*volume name or ID*>

5.10. Synchronizing the VLDB and Volume Headers

AFS provides transparent file access to its users because the VLDB constantly tracks volume locations. Whenever the Cache Manager needs a file, it contacts the VL Server in order to learn the current location of the volume containing the file. Therefore, the VLDB must accurately reflect the state of volumes on the file server machines as much as possible.

5.10.1. Keeping the VLDB Synchronized with Volume Headers

To keep the VLDB accurate, all **vos** commands that affect volume status automatically record the change in the appropriate VLDB entry. Whenever you issue a command that will change volume status, the Volume Server and VL Server follow the steps listed in section 5.2.5.

5.10.2. Some Indications that the VLDB is Out of Sync

Several symptoms and situations can alert you that a **vos** operation did not complete normally, with the probable result that the VLDB is out of sync with volumes on the servers. The symptoms and situations include:

- Error messages that indicate the operation terminated abnormally.
- You were forced to stop the operation in the middle (perhaps with **^C**).
- A file server machine or server process was down at the time you issued the **vos** command. The **bos status** command, described in section 4.3, shows the current status of a process.
- A subsequent **vos** operation fails because the failure left a VLDB entry locked. An error message on the following operation may report that it encountered a locked volume. The **vos listvldb** command, described in section 5.8.1, also tells you if a VLDB entry is locked.
- A subsequent **vos** operation fails because the failure left a volume marked "Off-line." The **vos listvol** command, described in section 5.8.2, shows if a volume is marked off-line.

5.10.3. How to Synchronize the VLDB with Volume Headers

If a **vos** operation fails while flags are being set or removed (corresponding to steps 1, 2, or 4 in section 5.10.1), then a discrepancy between VLDB and volumes may result. You can resynchronize the VLDB and volumes using the **vos syncvldb** and **vos syncserv** commands.

If a **vos** operation fails while the Volume Server is actually manipulating the volume (corresponding to step 3 in section 5.10.1), then it is not enough to just synchronize the VLDB and the volume headers: the volume may actually have become corrupted (left in an intermediate state). You must run the Salvager before attempting to resynchronize the VLDB and volume headers. See section 5.11.

5.10.4. The vos syncvldb Command

This command examines the volume header of each on-line volume on the file server machine (or, a single partition on the machine) that you specify. It looks at the VLDB entry that references that volume, to check that the VLDB properly records the volume's actual status at the site. If the VLDB is incorrect, this command alters it. It can even create VLDB entries if there is no entry for an on-line volume. The command has two other effects:

- If it finds an "orphaned" Backup volume whose ReadWrite source no longer exists at the same site, it removes the Backup from the site.
- It records the highest volume ID number it encounters as the new value of the counter that the Volume Server uses when allocating volume ID numbers. The next volume created will receive a volume ID number one higher than this counter.

5.10.5. The vos syncserv Command

This command examines every VLDB entry that mentions a site on the file server machine (and optionally, partition) that you specify. It verifies that the VLDB information is correct compared to the state of the volume at the site. There are several kinds of discrepancies it can solve:

- A volume that appears normal in the VLDB is marked "Off-line" on the file server machine. The volume is brought on-line. (Note that the "Off-line" flag does not necessarily indicate the volume is corrupted.)
- Two different sites have a copy of the ReadWrite version of a volume, both marked "Off-line." The probable cause is that the **vos move** command was interrupted. The command prints a message to inform you about the two volumes, but does nothing to the volumes or their VLDB entries.
- A volume version or site recorded in the VLDB does not actually exist at the indicated site. To solve the discrepancy, the VLDB entry is changed—in the case of ReadWrite, Backup, and the last site for ReadOnly.

.

Note: <u>For best results, you should run both of these commands on every file server machine, in the order specified in the following instructions.</u> If you do not, you cannot be sure that the entire VLDB accurately reflects the state of volumes on every file server machine. However, it is possible to run one command without the other, and to run them on selected machines or partitions only.

If the only problem with a volume is that its VLDB entry is locked, it is probably not necessary to resynchronize an entire partition. Use the **vos unlock** or **vos unlockvldb** command instead. See section 5.16 for more information on these commands.

⇒ To resynchronize the VLDB and volume headers

Step 1: Verify that you have the privilege necessary for resynchronizing the VLDB (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Issue **vos syncvldb** to make the VLDB reflect the true state of volumes on the specified partition/machine.

Note: For best results, repeat this command on every file server machine in your cell, before step 3.

% **vos syncvldb** <*machine name*> **[**<*partition name*>**] [-v** >> <*file*>**]**

where

syncv is an acceptable abbreviation of **syncvldb**.

machine name specifies the file server machine on which to inspect on-line volumes.

partition name is optional and not recommended: omitting it allows synchronization of all the volumes on *machine name*. If you do provide it, only that partition on *machine name* is checked.

-v >> *file* requests that a detailed trace of the synchronization be sent to *file*, which may either be in AFS or on the local disk of the machine from which you are issuing the command. During the process, the output scrolls on the screen very quickly. Saving the output in a file allows you to check it later.

Step 3: Issue **vos syncserv** to inspect each volume mentioned in the VLDB as having a version on the specified machine/partition.

Note: For best results, repeat this command on every file server machine in your cell, after step 2.

% **vos syncserv** <*machine name*> **[**<*partition name*>**] [-v** >> <*file*>**]**

where

syncs is an acceptable abbreviation of **syncserv**.

machine name specifies the file server machine the command interpreter should search for in the VLDB.

partition name specifies one partition on *machine name*. In general, you should omit this argument and allow synchronization of all the volumes on *machine name*.

-v >> *file* requests that a detailed trace of the synchronization be sent to *file*, which may either be in AFS or on the local disk of the machine from which you are issuing the command. During the process, the output scrolls on the screen very quickly. Saving the output in a file allows you to check it later.

5.11. Salvaging Volumes

Section 5.10 mentioned that during certain stages of a volume operation a failure can leave the data in volumes actually corrupted, rather than just cause the VLDB and volume headers to get out of sync. In particular, if the Volume Server or File Server is interrupted while it is actually manipulating the data in a volume, the data may be left in an intermediate state. For example, 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.

Section 5.11.1 discusses the kinds of volume corruption and their symptoms. Section 5.11.2 explains how the Salvager handles volume corruption.

5.11.1. The Kinds of Volume Corruption and Their Symptoms

Normally, the BOS Server automatically invokes the Salvager component whenever the Volume Server or File Server fails. The Salvager checks for and, if possible, repairs any inconsistency. Sometimes, however, you may notice behavior that indicates corruption serious enough to create problems but not serious enough to cause the File Server component to fail. In these cases you may need to invoke the Salvager manually.

The two main causes of this type of corruption and their corresponding symptoms are listed below:

- A file appears to exist in a directory but there is no corresponding data on disk. This can happen when a user creates a new file but the Volume or File Server crashes before the new file's data is actually stored on disk.

  Symptom: A properly authenticated and authorized user attempts to access a file stored on a fully functioning file server machine (i.e., all of its processes are running); however, the user is unable to access the file and error messages indicate that the file does not exist.

- Two files or versions of a file are sharing the same disk blocks, possibly because of the type of overwrite failure mentioned above. This type of corruption can be very dangerous. If the Volume Server or File Server is unsure of the status of the affected disk blocks, both may try to write yet more data there, but be unable to, resulting in loss of the data. This effect can cascade, causing you to lose all data on a partition.

  Symptom: The volumes appear to be off-line. This is often due to the fact that the Volume Server and File Server will refuse to access volumes that appear corrupted.

5.11.2. How the Salvager Handles Corruption

To correct the first kind of corruption mentioned in section 5.11.1, the Salvager will simply remove the errant file from the directory listing.

To correct the second kind of corruption, the Salvager may be forced to delete data from the corrupted disk blocks. Losing that amount of data is preferable to losing an entire partition.

5.11.3. How to Run the Salvager

When you notice symptoms such as those described in section 5.11.1, you should invoke the Salvager yourself, using **bos salvage**, before corruption can spread. (The Salvager is invoked in the **bos** command suite even though it concerns volumes because running the Salvager requires that the BOS Server temporarily shut down the Volume Server and File Server.)

Salvaging the volumes on a partition or entire machine causes a service outage because the File and Volume Server must shut down during the salvage. Salvaging a single volume renders only that volume inaccessible. If possible, inform users that you are about to run the Salvager.

You can choose how many of the volumes on the server to salvage by combining the **bos salvage** command's arguments in different ways. To salvage:

- All volumes on a file server machine, provide the **-server** argument and the **-all** flag. A file system outage results because the BOS Server stops the File Server and Volume Server components while the Salvager is running.
- All the volumes on one partition, provide **-server** and **-partition**. A file system outage results because the BOS Server stops the File Server and Volume Server components while the Salvager is running.
- Only one volume, combine **-server**, **-partition**, and **-volume**. Only that volume goes off-line during the salvage.

By default, up to four Salvager processes execute to salvage four or more partitions; if fewer than four partitions are being salvaged, only that number of Salvager processes execute. To increase or decrease the number of Salvager processes that execute, use the **-parallel** argument to specify an integer value between *1* and *32*.

You may want to increase the number of Salvager processes if you need to salvage a large number of partitions; however, you should not specify more Salvager processes than there are partitions to salvage because the excess processes will terminate automatically. Set the number of Salvager processes to *1* if you want to salvage the partitions sequentially.

The Salvager produces a trace of its actions, which it normally sends to stdout (standard out). You may direct the trace into a file if you wish.

During the salvage, the output of a **bos status** command on the *fs* process reports an auxiliary status of "`Salvaging file system`." When the salvage is complete, the BOS Server restarts the File Server and Volume Server components of the *fs* process.

⇒ To salvage all the volumes on a file server machine

Step 1: Verify that you have the privilege necessary for salvaging volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Salvage the volumes.

bos salvage <*machine name*> **[-file** <*salvage log output file*>**] -all**
[-parallel <*# of max parallel partition salvaging*>**]**

where

sal is an acceptable abbreviation of **salvage**.

machine name specifies the file server machine on which to run the Salvager.

salvage log output file
is optional. If you provide it, a trace of the salvage will be logged in the file. Provide a pathname unless *salvage log output file* is to reside in the working directory. Include the **-file** switch.

-all indicates that you want to salvage all of the volumes on *machine name*.

-parallel specifies the number of Salvager processes to be run in parallel. By default, four Salvager processes run if four or more partitions require salvaging; if fewer than four partitions require salvaging, a Salvager process will be run for each partition. Use this argument to increase or decrease this default number to a value between *1* and *32*.

⇒ To salvage all the volumes on a partition

Step 1: Verify that you have the privilege necessary for salvaging volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Salvage the volumes.

bos salvage <*machine name*> <*salvage partition*>
[-file <*salvage log output file*>**]**

where

sal is an acceptable abbreviation of **salvage**.

machine name specifies the file server machine on which to run the Salvager.

salvage partition
specifies which partition on *machine name* to salvage.

salvage log output file
is optional. If you provide it, a trace of the salvage will be logged in the file. Provide a pathname unless *salvage log output file* is to reside in the working directory. Include the **-file** switch.

⇒ To salvage a specific volume

Step 1: Verify that you have the privilege necessary for salvaging volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Learn the volume's ID number and location, if you do not know it already. If necessary, issue **vos listvldb**, which is described fully in section 5.8.1.

% **vos listvldb** <*volume name or ID*>

Step 3: Salvage the volume.

bos salvage <*machine name*> <*salvage partition*>
<*salvage volume number or volume name*>
[<*salvage log output file*>**]**

where

sal is an acceptable abbreviation of **salvage**.

machine name specifies the file server machine on which to run the Salvager.

salvage partition
specifies the partition housing the volume to be salvaged.

salvage volume number or volume name
the decimal-format volume ID number or the name of the volume to be salvaged.

salvage log output file
is optional. If you provide it, a trace of the salvage will be logged in the file. Provide a pathname unless *salvage log output file* is to reside in the working directory.

5.12. Setting and Listing Volume Quota and Current Size

By default, each newly created volume is allotted a ***maximum quota*** (or simply ***quota***) of 5000 kilobyte blocks. This means that the volume may not grow any larger than that. Depending on how you configure your cell and how much storage is available, 5000 kilobyte blocks may be too large for some volumes and too small for others. You can modify volume quota using either **fs setquota** or **fs setvol**.

Note: Volumes should generally remain small for easy load-balancing. Obviously, the size of a partition places an absolute limit on volume size. However, you should not allow a volume to grow beyond 250 megabytes in size; and volumes of 150 megabytes and smaller are the easiest to manage.

It is generally safe to "overpack" partitions by putting more volumes on them than would fit if all the volumes reached their maximum quota. However, only experience will help you determine 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.

Several commands allow you to examine a volume's quota. They differ in how much related information they produce.

⇒ To set maximum volume quota on a single volume

Step 1: Verify that you have the privilege necessary to set volume quota (membership in the *system:administrators* group). If necessary, issue the **pts membership** command, which is fully described in section 20.3.

% **pts membership system:administrators**

Step 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 that is to receive the indicated quota (for example, the volume's root directory). If you do not provide a full pathname, *dir/file path* is assumed to be a subdirectory of, or a file in, the working directory. If you omit this argument, the current working directory is used; in this case, the **-max** switch must be used.

max quota in kbytes
sets the volume's maximum 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.

⇒ To set maximum quota on one or more volumes

Step 1: Verify that you have the privilege necessary to set volume quota (membership in the *system:administrators* group). If necessary, issue the **pts membership** command, which is fully described in section 20.3.

% **pts membership system:administrators**

Step 2: Issue the **fs setvol** command to set the maximum 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 in each volume that is to have the indicated quota(s) (for example, the volume's root directory). If you do not provide a full pathname, each *dir/file path* is assumed to be a subdirectory of, or file in, the working directory. If you omit this argument, the current working directory is used.

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 list percent quota used

Step 1: At the command shell prompt (%), type

% **fs quota [**<*dir/file path*>**]**

where

q is an acceptable abbreviation of **quota**.

dir/file path names the directory or file in the volume for which percent quota used is to be reported. Omit this argument to use the current working directory.

An example:

% **fs quota /afs/transarc.com/usr/terry**

```
34% of quota used.
```

⇒ To list maximum quota, current size, and other information

Step 1: At the command shell prompt (%), type

% **fs listquota [**<*dir/file path*>**]**

where

lq is an alias for **listquota**.

dir/file path names the directory or file in the volume for which quota size is to be reported. Omit this argument to use the current working directory.

The output of this command reports:

- The name of the volume.
- The quota, in kilobyte blocks.
- The amount of space used, in kilobyte blocks.
- The percent of quota used.
- The percent of the volume's partition that is already full.

An example:

% **fs listquota /afs/transarc.com/usr/terry**

```
Volume Name      Quota    Used    % Used   Partition
user.terry       15000    5071       34%         86%
```

⇒ To list maximum quota, current size, and more

Step 1: At the command shell prompt (%), type

% **fs examine [**<*dir/file path*>**]**

where

exa is an acceptable abbreviation of **examine**.

dir/file path names the directory or file in the volume for which quota information should be reported. Omit this argument to use the current working directory.

The output of this command reports:

- The volume ID number of the volume.
- The name of the volume.
- The maximum quota, in kilobyte blocks.
- The amount of space used, in kilobyte blocks.
- The amount of space available and total size of the partition housing the volume, in kilobyte blocks.

An example:

% **fs examine /afs/transarc.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 numbers that appear in this output may not always agree with the corresponding numbers in the output of the standard UNIX **df** command. The **df** output reflects the state of partitions exactly when the command is issued. The numbers in this command's output may be up to five minutes old, as the Cache Manager polls the File Server for partition information at that frequency. Additionally, the partition size reported by the UNIX **df** command includes some reserved space that does not appear in this report of partition size, and so is likely to be about 10% larger.

5.13. Removing Volumes and their Mount Points

The **vos remove** command allows you to remove all three kinds of volumes. You can remove the ReadWrite version of a given volume and leave the ReadOnly intact, and vice versa. Removing the ReadWrite version automatically removes the Backup, but you may remove the Backup by itself.

5.13.1. Removing a ReadWrite Volume

The effect of removing the ReadWrite version of a volume depends on whether any copies of the ReadOnly version also exist.

If there are no ReadOnly copies

- The entire VLDB entry is erased from the VLDB.
- The Backup version is also removed from its site.

If there are ReadOnly copies

- The ReadWrite volume ID is retained in case you decide to restore the ReadWrite volume later.
- The Backup version is removed from its site, but its volume ID is retained in case you decide to restore the ReadWrite version and make another Backup version later.
- The ReadOnly copies are unaffected, but cannot ever be altered.

When you remove a ReadWrite volume (with no ReadOnly copies), you should also remove its mount point from the file tree, so that users will not continue to think they can access the volume's contents. Do not remove the mount point if copies of the ReadOnly version are going to remain in the system; doing so will make the copies inaccessible.

5.13.2. Removing a ReadOnly Volume

The effect of removing a copy of the ReadOnly version of a volume depends on whether other copies still exist at other sites and whether any other versions of the volume exist.

If other ReadOnly sites still exist

- The site information for each copy being removed disappears from the VLDB entry.
- The number of sites counter decrements by one for each copy removed.

If no other ReadOnly sites exist

- The site information for each copy being removed disappears from the VLDB entry.

- The number of sites counter decrements by one for each copy removed.
- The volume ID of the ReadOnly site is set to 0 in the VLDB, but its actual volume ID number is still recorded in case you decide to make a ReadOnly version again.

If no other sites of any kind exist, the entire VLDB entry disappears.

It is not necessary to remove copies of the ReadOnly version from their sites if you only need to release a new clone. Simply use the **vos release** command described in section 5.5.

There is no reason to remove a mount point when removing the last copy of a volume's ReadOnly version, unless the ReadWrite version also no longer exists. In that case, remove the mount point to any copies of the volume, as described in step 5 of the ReadWrite removal instructions.

5.13.3. Removing a Backup Volume

Removing the Backup version of a volume frees the (usually) small amount of space it occupies on disk and sets the Backup volume ID to *0*, but its volume ID is still recorded in case you want to create a Backup version later.

You probably will have mounted the Backup version separately from the ReadWrite version. When you remove it, you should remove its mount point from the file tree, so that users will not continue to think they can access the volume's contents.

5.13.4. Other Removal Commands

The **vos remove** command works only when a VLDB entry and volume header both exist for the volume. Under normal circumstances, you should use **vos remove** because it automatically records its effects in the VLDB.

There are several commands that allow you to remove a VLDB entry or a volume header without requiring that both the VLDB entry and the volume header exist; however, these commands can put the VLDB and volumes out of sync. You should use these commands only in special situations (like those described here), not in normal situations. These commands are discussed below and described fully in the *AFS Command Reference Manual*.

The **vos zap** command allows you to remove a volume from its site (i.e., remove the volume header) even though the volume's VLDB entry no longer exists. You might discover a missing VLDB entry by noticing that **vos listvol** shows the volume header but **vos listvldb** reveals no VLDB entry for the volume. Removing the volume header allows you to resynchronize the VLDB and volume site by running **vos syncvldb** and **vos syncserv**.

The **vos remsite** command removes a ReadOnly site definition from the VLDB without affecting the volume on the file server machine. Use this when you have mistakenly defined a ReadOnly site using **vos addsite**, but have not released a copy to that site using **vos release**. If you have actually released a volume to the site, you should use **vos remove** instead.

The **vos delentry** command removes the entire VLDB entry that mentions a particular volume. If versions of the volume actually exist, 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 **vos zap** during an emergency), and do not want to take the time to resynchronize the entire VLDB with **vos syncvldb** and **vos syncserv**.

⇒ To remove a ReadWrite volume and unmount it

Step 1: Verify that you have the privilege necessary for removing volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the DELETE access right for the directory from which you wish to remove the volume's mount point. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the DELETE right if necessary.

Step 3: **(Optional).** Dump the volume in preparation for saving it permanently on tape. This step is recommended, since you may wish to restore the volume later. Section 5.14 discusses the **vos dump** command further.

% **vos dump** <*volume name or ID*> **0** <*arg*>

where

volume name or ID
: identifies the volume you are about to remove, either by its complete name or volume ID number.

0
: specifies that you wish to dump the complete contents of the volume.

arg
: specifies the file in AFS into which you want to dump the volume. The file should be located outside the volume you are about to remove. If you do not provide a complete pathname, *arg* will go into the working directory.

Step 4: Remove the ReadWrite volume.

% **vos remove** <*machine name*> <*partition name*> <*volume name or ID*>

where

remo
: is an acceptable abbreviation of **remove**.

machine name
: specifies the file server machine on which the volume resides.

partition name
: specifies the partition on *machine name* where the volume resides.

volume name or ID
: identifies the ReadWrite volume to be removed, either by its complete name or its volume ID number.

Step 5: Remove the mount point, but not if any copies of the ReadOnly version will remain in the file system and you wish them to be accessible.

% **fs rmmount** <*directory*>

where

rmm is an acceptable abbreviation of **rmmount**.

directory is the mount point for the volume. Provide a pathname unless *directory* is a subdirectory of the working directory.

Step 6: **(Optional).** If you created a dump file in step 3, transfer it to tape. The preferred method is to use the AFS Backup System. See chapter 6.

Note: Recall that the Backup version was also removed automatically in step 4. Removing the mount point for the ReadWrite volume in step 5 is sufficient to remove the Backup version's mount point as well, if (as recommended) you mounted the Backup at a subdirectory of the ReadWrite volume's root directory. If you mounted the Backup at a completely separate directory, you need to repeat step 5 explicitly for the Backup's mount point.

⇒ To remove a ReadOnly volume

Step 1: Verify that you have the privilege necessary for removing volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Remove the ReadOnly volume from the desired site. **Repeat this command for each ReadOnly site you wish to remove.**

% **vos remove** <*machine name*> <*partition name*>
<*volume name or ID*>**.readonly**

where

remo is an acceptable abbreviation of **remove**.

machine name specifies the file server machine on which the volume resides.

partition name specifies the partition on *machine name* where the volume resides.

*volume name or ID***.readonly**
: specifies the complete name of the ReadOnly volume you wish to remove. If you provide a volume ID instead, do not add the **.readonly** extension.

⇒ To remove a Backup volume

Step 1: Verify that you have the privilege necessary for moving volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the DELETE access right for the directory from which you wish to remove the volume's mount point. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the DELETE right if necessary.

Step 3: Remove the Backup volume. This is not necessary if you have already removed its ReadWrite source.

% **vos remove** <*machine name*> <*partition name*>
<*volume name or ID*>**.backup**

where

remo is an acceptable abbreviation of **remove**.

machine name specifies the file server machine on which the volume resides.

partition name specifies the partition on *machine name* where the volume resides.

*volume name or ID***.backup**
specifies the complete name of the Backup volume you wish to remove. If you provide a volume ID instead, do not add the **.backup** extension.

Step 4: Remove the Backup volume's mount point.

% **fs rmmount** <*directory*>

where

rmm is an acceptable abbreviation of **rmmount**.

directory is the mount point for the volume. Provide a pathname unless *directory* is a subdirectory of the working directory.

5.14. Dumping and Restoring Volumes

When you dump a volume, its contents are translated into ASCII format and stored in a file. Dumping does not affect the status of the volume at its site or in the VLDB. You can dump any of the three types of volumes. When you restore a volume, its contents are placed back into the file system from the dump file.

5.14.1. About Dumping Volumes

If you wish to examine the contents of the dump file, keep in mind that it may have special characters in it. The **cat** command probably cannot recognize these characters, though some text editors may be able to.

Dumping is useful when you

- Are preparing a volume to be backed up permanently on tape. This is perhaps the most common reason for dumping. See chapter 6 to learn about the AFS Backup System.
- Need to save a "snapshot" of a volume, perhaps because it is to be removed from the file system, but may later need to be restored. (See the instructions in section 5.13 for removing a ReadWrite volume.)
- Need to translate a volume to a different format (different types of machines may use different volume formats). ASCII is a "common language" from which you can restore the volume into the format used by the new machine.
- Wish to copy only certain parts of a volume, based on modification time stamps. See the paragraph on incremental dumps below.

Note: The contents of a volume are unavailable during a dump. It is recommended that you dump only the Backup version of a volume, as not to interrupt access to the ReadWrite version. Backup versions tend to be accessed much more rarely—usually only when someone needs to recover a lost file—so you are unlikely to inconvenience anyone by dumping a Backup version.

5.14.1.1. Full versus Incremental Dumps

A full dump contains the entire contents of the volume. An incremental dump contains only those files and directories that were modified after a certain date. The **vos dump** command allows you to create both full and incremental dumps; step 3 below explains how to specify a full versus an incremental dump.

5.14.2. About Restoring Volumes

Although you can dump any of the three types of volumes, you can restore a dump file only as a ReadWrite volume. This is useful when the ReadWrite version of a volume is corrupted and you want to dump a Backup or ReadOnly volume and restore it as a ReadWrite.

You can put dumped volumes back into the file system with the **vos restore** command, which automatically translates the dump file's contents from ASCII back into the volume format appropriate for the file server machine to which you are restoring the volume. You can restore a dumped volume in two ways:

- As the same volume from which you dumped it. In this case, the contents of the dump file overwrite the current contents of the volume (after you are prompted to confirm if that is acceptable). This is convenient if the current contents are corrupted or wrong in some other way, because it allows you to replace them with a coherent version from the past or from one of the volume's clones.
- In a brand new volume, which is created under the name you provide.

You can specify a volume ID number as you restore the volume, though in general you should allow the Volume Server to determine the appropriate volume number. One reason to specify the volume ID is if a volume's VLDB entry has disappeared for some reason, but you know the former ReadWrite volume ID number and want to reuse it.

The creation date of the restored volume, which is stored in the volume's header, is set to the date and time at which it is restored.

⇒ To dump a volume

Step 1: Verify that you have the privilege necessary for dumping volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the INSERT access right for the directory in which you wish to place the dump file. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the INSERT right if necessary.

Step 3: Dump the volume.

% **vos dump** <*volume name or ID*> <*dump from time*> <*arg*>

where

volume name or ID
identifies the volume to be dumped, either by its complete name or volume ID number. If you want to dump the ReadOnly or Backup version, be sure to specify its volume ID number or add the appropriate extension (`.readonly` or `.backup`) to the name.

dump from time specifies the date for incremental dumps; only files and directories with modification time stamps greater than *dump from time* will be dumped. There are three types of legal values:

- *0* causes a full dump. Any other value causes an incremental dump.
- *mm/dd/yy* specifies 12:00 a.m. on the indicated date (month/day/year). Only elements with modification time stamps later than this time are dumped. Examples: *1/23/90*, *10/7/89*.
- *"mm/dd/yy hh:mm"* specifies a time "hours:minutes" on the indicated date (month/day/year). Only files with modification time stamps later than this time are dumped. The time should be in 24-hour format (for example, 20:30 is 8:30 p.m.) Date format is the same as for a date alone. Surround the entire instance with quotes because it contains a space. Examples: *"1/23/90 22:30"*, *"10/7/89 3:45"*.

arg is the pathname of the file into which the volume's contents should be dumped. If you give only a name without a path, *file* will go into the working directory.

⇒ To restore a dump file, overwriting an existing volume

Step 1: Verify that you have the privilege necessary for restoring volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the READ access right for the directory where the dump file resides. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the READ right if necessary.

Step 3: Restore the contents of the dump file into a ReadWrite volume, overwriting the current contents. The volume retains its current volume ID number. Type the following command on a single line.

% **vos restore** <*machine name*> <*partition name*>
<*name of volume to be restored*> <*dump file*>

where

res is an acceptable abbreviation of **restore**.

machine name specifies the file server machine where the current volume already exists.

partition name specifies the partition on *machine name* where the current volume resides.

name of volume to be restored
is the complete name of the ReadWrite volume you wish to overwrite.

dump file is the dump file to be restored. Provide a pathname unless it resides in the working directory.

The Volume Server will prompt you to confirm the overwrite.

Step 4: If ReadOnly copies of the former ReadWrite volume exist, you need to replace them with copies of the new volume contents. Use the **vos release** command described fully in section 5.5.

% **vos release <***volume name or ID***>**

Step 5: If a Backup version was made from the former ReadWrite volume, you need to replace it with a Backup version of the new volume contents. Use the **vos backup** command described fully in section 5.6.

% **vos backup <***volume name or ID***>**

⇒ To restore a dump into a new volume and mount it

Step 1: Verify that you have the privilege necessary for restoring volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers <***machine name***>**

Step 2: Verify that you have the READ access right for the directory where the dump file resides, and the INSERT and ADMINISTER rights in the directory where you wish to create a mount point. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [<***dir/file path***>]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the READ and INSERT rights if necessary.

Step 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 will help guide your decision. You may wish to put the volume on the emptiest partition that meets your other criteria. You can learn how much space is available on a file server machine's partitions (and their total size) with **vos partinfo**.

% **vos partinfo <***machine name***> [<***partition name***>]**

where

p	is an acceptable abbreviation of **partinfo**.
machine name	specifies the file server machine about which you want partition information.
partition	is optional; if you provide it, the output displays the size and space available on that one partition. If you omit it, the output displays the size and space available for all partitions on *machine name*.

Note: The total partition size reported in this command may not agree with the same figure in the output of the standard UNIX **df** command. The **df** total size includes some reserved space that does not show up in this report, and so is likely to be about 10% larger.

Step 4: Create a new volume and restore the dump file into it. Type the following command on a single line.

% **vos restore** <*machine name*> <*partition name*>
<*name of volume to be restored*> <*dump file*>
[<*volume ID*>**]**

where

res is an acceptable abbreviation of **restore**.

machine name specifies the file server machine where you wish to place the new ReadWrite volume.

partition name specifies the partition on *machine name* where the new volume will reside.

name of volume to be restored
is the name you want to give the new volume. It may not be longer than 22 characters.

dump file is the dump file to be restored. Provide a pathname unless it resides in the working directory.

volume ID if provided, specifies the new volume's volume ID number. It should not be the ID number of an existing volume. In most cases, you should allow the Volume Server to allocate this number itself.

Step 5: Mount the new volume in the file tree, so that its contents are visible.

% **fs mkmount** <*directory*> <*volume name*> **[-rw]**

where

mk is an acceptable abbreviation for **mkmount**.

directory names the mount point you wish to create. Do not create a file or directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory.

volume name is the full name of the volume, as defined with *name of volume to be restored* in step 4.

-rw should be included only in the rare cases where you want the mount point to be a ReadWrite one. See section 5.7.2 for an explanation of ReadWrite mount points.

Step 6: **(Optional).** Verify that you have created the mount point correctly with the **fs lsmount** command, described fully in section 5.7.

% **fs lsmount** <*directory*>

5.15. Renaming Volumes

The **vos rename** command allows you to rename a volume. For example, you should rename a volume if you change the user ID of a user whose home directory is in that volume. Section 17.9 gives complete instructions for changing user IDs.

You may change the name of a volume's ReadWrite version only, but the associated ReadOnly and Backup names change automatically to match. You will need to remove the mount point that indicates the old volume name and replace it with one that indicates the new name.

⇒ To rename a volume

Step 1: Verify that you have the privilege necessary for renaming volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the ADMINISTER, DELETE and INSERT access rights for the directory in which you will replace the volume's mount point. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself the DELETE and INSERT rights if necessary.

Step 3: Rename the volume.

% **vos rename** <*old volume name*> <*new volume name*>

where

ren is an acceptable abbreviation of **rename**.

old volume name
is the current name of a ReadWrite volume.

new volume name
is the new name for the volume. It must be no longer than 22 characters, or the operation will fail. (This restriction allows the addition of a `.readonly` or `.backup` without violating the absolute length restriction of 31 characters.)

Step 4: Remove the mount point indicating the volume's old name.

% **fs rmmount** <*directory*>

where

rmm is an acceptable abbreviation of **rmmount**.

directory	is the current mount point for the volume. Provide a pathname unless *directory* is a subdirectory of the working directory.

Step 5: Create a new mount point to indicate the volume's new name.

% fs mkmount <*directory*> <*volume name*> **[-rw]**

where

mk	is an acceptable abbreviation for **mkmount**.
directory	names the mount point you wish to create. Do not create a file or directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory.
volume name	is the full name of the volume, matching *new volume name* from step 3.
-rw	should be included only in the rare cases where you want the mount point to be a ReadWrite one. See section 5.7.2 for an explanation of ReadWrite mount points.

5.16. Unlocking and Locking VLDB Entries

The VL Server locks the 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 this prevents the inconsistency or corruption that could result from multiple simultaneous operations on a volume.

5.16.1. Locking VLDB Entries

The **vos lock** command allows you lock a VLDB entry to prevent any one else from executing a volume operation that would require writing the VLDB. You may want to lock a VLDB entry when you suspect something is wrong with it and you want to make sure that it does not change while you are investigating the problem.

5.16.2. Unlocking VLDB Entries

If a volume operation fails prematurely, VLDB entries may remain locked, preventing you from executing commands that would correct problems resulting from the failure. You must unlock the VLDB entry before proceeding. The **vos examine** and **vos listvldb** commands allows you to examine VLDB entries even if they are locked.

The **vos unlock** command allows you to unlock a specific VLDB entry.

The **vos unlockvldb** command allows you to unlock a set of entries, depending on the arguments you provide:

- If you provide no arguments, all locked entries are unlocked.
- If you specify a file server machine with **-server**, all locked entries that mention the machine in a site definition are unlocked.
- If you specify a partition with **-partition**, all locked entries that mention that partition (on any machine) in a site definition are unlocked.
- If you specify both **-server** and **-partition**, all locked entries that mention that site are unlocked.

⇒ To determine if a VLDB entry is locked

Step 1: Issue the **vos listvldb** command.

% **vos listvldb** <*volume name or ID*>

where *volume name or ID* specifies the volume whose VLDB entry you wish to inspect.

If the VLDB entry is locked, the word "`LOCKED`" will appear alone on the first line of the output.

⇒ To unlock a single VLDB entry

Step 1: Verify that you have the privilege necessary for unlocking VLDB entries (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: 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 may be any of the three versions of the volume.

⇒ To unlock a set of VLDB entries

Step 1: Verify that you have the privilege necessary for unlocking VLDB entries (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Unlock the desired entries. The introductory material above discusses the effect of combining the arguments in different ways.

% **vos unlockvldb [**<*machine name*>**] [**<*partition name*>**]**

where

unlockv	is an acceptable abbreviation of **unlockvldb**.
machine name	specifies a file server machine.
partition name	specifies a partition.

⇒ To lock a VLDB entry

Step 1: Verify that you have the privilege necessary for locking VLDB entries (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Lock the entry.

% **vos lock** <*volume name or ID*>

where

lo is an acceptable abbreviation of **lock**.

volume name or ID
identifies the volume to be locked, either by its complete name or volume ID number. It may be any of the three versions of the volume.

6. Backing Up the System

The AFS Backup System – when combined with a set of well-designed backup procedures – can greatly simplify the process of backing up AFS data and restoring it to AFS should the original data be lost or corrupted. This chapter contains instructions for using the Backup System and some recommendations for creating a set of backup procedures.

The chapter contains the following sections:

6.1 Summary of Commands 6-1
6.2 Introduction to the Backup System 6-3
6.3 Configuring the Backup System. 6-12
6.4 Interfaces to the Backup System 6-28
6.5 Listing Backup Information 6-35
6.6 Backing Up Data 6-43
6.7 Procedures for Automating the Backup System. 6-50
6.8 Restoring and Recovering Data 6-55
6.9 The Backup Database 6-59

6.1. Summary of Commands

Start Tape Coordinator . . . . **butc**
Stop Tape Coordinator . . . . **^c**
Check status of Tape Coordinator. . . . . **backup status**
Determine tape and EOF marker size . . . . **fms**
Enter interactive mode . . . . **backup**
Leave interactive mode. . . . . **(backup) quit**
List operations in interactive mode. . . . . **(backup) jobs**
Cancel operation in interactive mode . . . . **(backup) kill**
Define Tape Coordinator in Backup Database . . . . **backup addhost**
Add volume set . . . . **backup addvolset**
Add volume entry to volume set. . . . . **backup addvolentry**
Delete volume set . . . . **backup delvolset**
Delete volume entry from volume set. . . . . **backup delvolentry**
Add dump level. . . . . **backup adddump**
Change expiration date on existing dump level . . . . **backup setexp**
Delete dump level . . . . **backup deldump**
Label tape . . . . **backup labeltape**
Read label on tape. . . . . **backup readlabel**
Remove Tape Coordinator from Backup Database . . . . **backup delhost**
Check status of Backup Database. . . . . **backup dbverify**
List volume sets and entries . . . . **backup listvolsets**
Display dump hierarchy . . . . **backup listdumps**
Display backup information . . . . **backup dumpinfo**

List Tape Coordinators' port offsets	**backup listhosts**
Display volume's dump history	**backup volinfo**
Scan contents of tape	**backup scantape**
Back up data	**backup dump**
Delete dump set from Backup Database	**backup deletedump**
Restore volume	**backup volrestore**
Restore partition	**backup diskrestore**
Save Backup Database	**backup savedb**
Restore Backup Database	**backup restoredb**

6.2. Introduction to the Backup System

The AFS Backup System allows you to perform two main functions:

- Back up data in AFS to tape.
- Restore backup data from tape to AFS.

The AFS Backup System allows you to control many aspects of the backup process including how often backups are performed, which volumes are backed up, and how often full versus partial (or incremental) backups are made. To some extent, you can also control the level of automation of the AFS Backup System by using or not using certain features.

Before learning how to use the AFS Backup System commands, you should familiarize yourself with some basic concepts. Sections 6.2.1 through 6.2.11 explain the basic concepts of the AFS Backup System and also provide some guidelines for establishing backup procedures.

Note: Some operating systems have their own **backup** command. If your operating system has a **backup** command, make certain you use the complete pathname when issuing the AFS **backup** commands.

6.2.1. Volume Sets and Volume Entries

Data in AFS is backed up on a volume basis. Before the contents of a volume can be backed up, that volume must be associated with a volume set. A ***volume set*** is a grouping of one or more volumes that are backed up together as a single unit. If a volume is not associated with a defined volume set, the data it contains cannot be backed up using the AFS Backup System.

Volumes are associated with a volume set by defining one or more volume entries for that volume set. A ***volume entry*** identifies the name and location of one or more volumes using three fields: the file server machine name, the partition name, and the volume name. All of the fields support the use of metacharacters so that a volume entry can be used to identify a single volume or a group of volumes with similar names and/or locations.

The volumes in a volume set are treated as a single unit for backup purposes. They are backed up together and stored on the same tape or set of tapes. Therefore, you should only group volumes into the same volume set if you intend to store them together and back them up at the same time and with the same frequency.

See section 6.3.4 for information on defining volume sets and volume entries.

6.2.2. Dumps and Dump Sets

A ***dump*** is a single backup of a single volume set. There are several types of dumps:

- A ***full dump*** is a complete copy of all files in the specified volume set.
- An ***incremental dump*** is a copy of those files in the specified volume set that have changed since a previous parent dump was made.
- A ***parent dump*** is a reference point for incremental dumps that is used to determine which files in a volume set need to be backed up and which do not. A parent dump can itself be either a full dump or an incremental dump.

A ***dump set*** is a grouping of one or more dumps placed on one or more tapes; the tapes in a dump set are collectively named using information about the first dump in the dump set. A dump set is created when a dump is backed up onto a new tape or a tape containing another dump set which is overwritten in the process.

Once a dump set is created, additional dumps may be appended to it. The first dump in a dump set is referred to as the ***initial dump***. Each subsequent dump placed on the resulting dump set is called an ***appended dump***. Appending dumps to existing dump sets cuts down on the amount of unutilized tape and the number of tape changes.

When a dump is appended to a dump set, it becomes bound to that dump set and shares the fate of that dump set. The Backup Database record associated with an appended dump cannot be deleted directly; to delete the appended dump record, you must delete the entire dump set record (by deleting the initial dump record associated with the dump set). Each tape in a dump set is also bound to that dump set. You cannot reuse a tape in a dump set without removing all record of the entire dump set from the Backup Database. (See section 6.2.11 for information on the Backup Database.)

6.2.3. Dump Hierarchies and Dump Levels

A ***dump hierarchy*** is defined by the dump levels that compose it; a ***dump level*** is specified by its pathname. In general, a dump hierarchy is a schedule that indicates the rate and time at which dumps should be made. A dump hierarchy defines the relationship among the full, incremental, and parent dump levels in that hierarchy. The top level in a dump hierarchy always represents a full dump; each lower level in the hierarchy is an incremental dump that refers to the next higher level as its parent. The following is an example of a dump hierarchy:

```
/sunday1
      /monday
      /tuesday
      /wednesday
      /thursday
      /friday
/sunday2
      /monday
      /tuesday
      /wednesday
      /thursday
      /friday
```

In the example, */sunday1* and */sunday2* are full dumps; */sunday1/monday*, */sunday1/tuesday*, */sunday1/wednesday*, */sunday1/thursday*, and */sunday1/friday* are incremental dumps that refer to *sunday1* as their parent dump; and */sunday2/monday*, */sunday2/tuesday*, */sunday2/wednesday*, */sunday2/thursday*, and */sunday2/friday* are incremental dumps that refer to */sunday2* as their parent dump.

Although it is not necessary for every element in a dump hierarchy to be unique, every dump level must be unique. In the example, the name *monday* is repeated twice in two separate dump levels; however, each dump level that contains it is unique. For example, */sunday1/monday* is different from */sunday2/monday*.

You should use dump level names that correspond to real-world times. These names can help you remember when to perform backups; however, the Backup System does not automatically back up volumes according to the names assigned in the dump hierarchy. For example, the Backup System does not automatically perform a dump on a Thursday simply because there is a dump level called */thursday*.

If you intend to backup different volume sets at different rates, times, and levels, you may need to define several "subhierarchies" within your dump hierarchy. For example, the previous example hierarchy might be useful for volume sets that change frequently, like user volume sets. However, you might also want to include the subhierarchies shown below for volume sets that change less frequently (like program binary volumes):

```
/sunday_bin1
      /monday
            /wednesday
                  /friday
/sunday_bin2
      /monday
            /wednesday
                  /friday
```

When dumping a volume set, <u>do not</u> alternate between incremental dumps from different subhierarchies.

Section 6.3.5 contains instructions for creating dump hierarchies.

6.2.4. Dump Names and Tape Names

A dump is named using the dumped volume set name and the name of the final, or lowest, element in the dump level, as follows:
<*volume set name*>.<*dump level name*>

For example, a dump of the volume set *user* at the dump level */sunday/friday* would be named
user.*friday*

Because dump names are not unique, a unique ***dump ID*** is associated with each dump when the dump is created. The dump ID is the number of seconds since 12:00 a.m. on January 1, 1970 (the UNIX time equivalent of 0).

The tape(s) that contain a dump set are named using the initial dump's name and dump ID and the numerical position of the tape in the dump set, as follows:
<*volume set name*>.<*dump level name*>.<*tape index*> (*Dump ID*)

For example, if the volume set *uservol* is dumped at the dump level */sunday/monday* and the resulting dump ID is 123455555, the first tape in the dump set is named
uservol.monday.1 (*123455555*)

No matter how many additional dumps are appended to a dump set, the names of the tapes in the dump set will always be based on the volume set name, the dump level name, and the dump ID of the initial dump.

6.2.5. Expiration Dates

When created, a dump level should be assigned an ***expiration date***; the expiration date associated with a dump level is in turn assigned to all dumps that are backed up at that dump level. Expiration dates help to prevent accidental overwriting of backup information and can also be useful in establishing a tape recycling schedule. (See section 6.2.6 for information on tape recycling schedules.)

Expiration dates can be specified using absolute (for example, June 5, 1997 at 23:59:59) or relative (for example, three weeks from the time of the backup) values. The expiration date associated with a dump level can be changed; however, once an expiration date is assigned to a dump it cannot be changed for that dump.

A dump set inherits the most-future expiration date associated with its constituent dumps. None of the tapes in a dump set can be overwritten until the most-future expiration date associated with that dump set expires. The only way to overwrite a tape associated with an unexpired dump set is to relabel the tape's magnetic label. When you relabel a tape, you remove all record of the entire dump set, including appended dumps, from the Backup Database.

Note: The expiration date of the initial dump in a dump set is stored on the tape label. The expiration dates of appended dumps are stored in the Backup Database. If the record of a dump set is removed from the Backup Database, only the expiration date on the tape will be used in determining if a dump set is expired.

Section 6.3.5.1 contains instructions for defining and changing expiration dates for dump levels.

6.2.6. Tape Recycling Schedules

Because most cells do not have an unlimited number of backup tapes (or an unlimited amount of storage space for those tapes), it is often necessary to define tape recycling schedules. A tape recycling schedule defines the rate at which tapes containing old backup data can be overwritten with new data without compromising the ability to successfully restore a specified amount of backup data.

When setting up tape recycling schedules, 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 restoration of user data at a daily level for up to two weeks, you cannot recycle the tapes in a dump set containing a daily user dump for at least two weeks after the dump is created. If you decide to guarantee restoration of user data at a weekly level for up to to four weeks, you cannot recycle the tapes in a dump set containing a weekly user dump for at least four weeks after the dump is created. The following example dump hierarchy would allow such a recycling schedule. (The expiration date of the daily incremental dumps are set to 13 days; the expiration date of weekly full dumps are set to 27 days.)

```
/sunday1   expires in 27d
      /monday   expires in 13d
      /tuesday   expires in 13d
      /wednesday   expires in 13d
      /thursday   expires in 13d
      /friday   expires in 13d
/sunday2   expires in 27d
      /monday   expires in 13d
      /tuesday   expires in 13d
      /wednesday   expires in 13d
      /thursday   expires in 13d
      /friday   expires in 13d
/sunday3   expires in 27d
      /monday   expires in 13d
      /tuesday   expires in 13d
      /wednesday   expires in 13d
      /thursday   expires in 13d
      /friday   expires in 13d
/sunday4   expires in 27d
      /monday   expires in 13d
      /tuesday   expires in 13d
      /wednesday   expires in 13d
      /thursday   expires in 13d
      /friday   expires in 13d
```

There are two good reasons to build recycling schedules into the structure of the dump hierarchy in this way:

- You cannot overwrite a tape until all of the dumps associated with the tape's dump set have expired. The only way to create a new dump set on an unexpired tape is to relabel that tape.

  For example, assume that you dump the volume set *user* at the */sunday1/monday* dump level to create a new dump set. Assuming that no dumps with future expiration dates are appended to the dump set, the tapes in the dump set can be recycled two weeks (13 days) later. This would be just in time to reuse the tapes to back up the same data at the */sunday3/monday* dump level.

- You cannot overwrite a tape during the creation of a new dump set unless the initial dump in the new dump set has the same name as the initial dump in the old dump set. The only way to backup an initial dump whose name conflicts with the name of the initial dump currently on a tape is to relabel that tape.

  For example, assume that the dump *user.monday* was the initial dump in a dump set. The tapes in that dump set could only be recycled by dumping the *user* volume set at a dump level whose final element is */monday* so that the new initial dump would also be *user.monday*.

If you decide to append dumps from multiple days and/or with different expiration dates to the same dump set, your recycling schedule will become much more complicated. For example, say you dumped the *user* volume set at the */sunday1/monday* dump level on November 1, 1993 to create a new dump set. On the following two days, you append the *user.tuesday* and *user.wednesday* dumps to the initial *user.monday* dump.

According to the rules for recycling, you would be unable to recycle this dump set until three weeks after its creation (that is, November 22, 1993). The expiration date of the *user.wednesday* dump (13 days) would prevent you from recycling the tapes until at least Monday, November 17, 1993. Furthermore, the name of the initial dump (*user.monday*) would prevent you from recycling the tapes until the following Monday, November 22, 1993 (when you could recycle the tapes by dumping the *user* volume set at the */sunday4/monday* dump level).

Note: Each time an initial dump is placed on a tape, the Backup System updates the `useCount` recorded in the tape's label. The `useCount` records the number of times a dump has been written to the tape. It can be displayed with the **backup readlabel**, **backup scantape**, or **backup dumpinfo** commands (if **-id** and **-verbose** are included with the command); all of these commands are described later in this chapter. Because there is a limit to the number of times any tape can be overwritten, you should not dump to a tape more times than the tape's manufacturer suggests.

6.2.7. Tape Archiving

While some backup data may be overwritten on a regular basis, other backup data must be saved for long (sometimes undefined) periods of time. The long term storage of backup data is called ***archiving***. Some AFS cells archive backup data on a regular basis; other cells archive backup data on an as-needed basis. The Backup System is flexible enough to accommodate both approaches.

One way to archive data on a regular basis is to create date-specific dump levels in your dump hierarchy. For instance, if your cell has decided to archive all data on a monthly basis, you might create the following dump level scheme in your dump hierarchy:

```
/jan93
/feb93
/mar93
/apr93
```

To archive data on an as-needed basis, you must remove the complete dump set containing the relevant data from any recycling schedule, make any manual adjustments to the tapes to make them read only, and place them in your archive storage area. If your cell frequently archives data in this manner, it is important to keep your dump sets small so that the total amount of irrelevant data that might be associated with the relevant data in the archived dump set is kept to a minimum. This will help you save both tape and storage space.

6.2.8. Restoring Backed Up Data

Once data is dumped onto tape using the AFS Backup System, that data can be restored back into AFS (assuming the tapes are undamaged). The Backup System can restore data in two ways; it can restore

- One or more specific volumes.
- An entire disk partition.

There are two ways to restore specific volumes:

- A ***full restore***, which recreates a volume as it was at its last dump (using the most recent full dump and any subsequent incremental dumps).
- A ***date-specific restore***, which recreates a volume as it was at its last dump prior to the specified date (the Backup System uses the last full dump and any subsequent incremental dumps done before the specified date, so the volume is as close as possible in content to what it was on the specified date).

When restoring a volume, the Backup System first restores a full dump of the volume. It then checks any subsequent incremental dumps and alters the restored volume according to any changes it finds in the incremental dumps.

A disk partition can only be restored if the backup version of the volumes on that partition reside on the same type of tape. When restoring a disk partition, the Backup System consults the VLDB to determine which volumes go on that partition (see section 5.2.3 for more information on the VLDB). Therefore, it is important that you do not run the **vos syncserv** command if a disk or disk partition becomes corrupted or if you suspect that a disk or disk partition has become corrupted.

See section 6.8 for information on restoration.

6.2.9. Tape Labels, Dump Labels, and EOF Markers

There are two ***tape labels*** associated with a backup tape: the internal ***magnetic label*** and the external ***paper label***. The magnetic label is an integral part of the tape itself. It contains a variety of information on the tape and its contents, including, among other things:

- The name of the tape, indicating its contents. There are three acceptable entries for this field:
  - The complete name of the tape.
  - An indicator of "" or NULL, which can be created only when the **backup labeltape** command is issued without specifying a tape name. The Backup System replaces the empty indicator with the correct name when it puts a dump onto the tape.
  - No name, indicating the tape is unused. Again, the Backup System generates the correct name when it transfers a dump onto the tape. (In this case, the tape contains no label.)
- The size, or capacity, of the tape. A letter indicates the unit of measure (`k` or `K` for kilobytes, `m` or `M` for megabytes, or `g` or `G` for gigabytes). See section 6.3.2 for information on configuring the tape size.

The paper label on a tape should contain some text to make the tape easy to identify for backup and restore purposes. You might want to include the following information: the tape name, the initial dump ID, the backup date, and the expiration date.

Prior to placing either an initial or appended dump on a tape, the Backup System will create a ***dump label*** on the tape. (The dump label on the first tape in a dump set is also the tape label.) Among other things, the dump label contains the following information:

- The name of the tape containing the dump.
- The date and time that the dump was created.
- The cell origin of the dump.
- The size of the dump in kilobytes.
- The dump level of the dump.
- The dump ID of the dump.

The Backup System also appends an ***EOF (end-of-file) marker*** after each volume dumped to tape. The size of this marker affects the amount of space available for backup data. EOF marker size can vary from 2 kilobytes to more than 2 megabytes, depending on the type of tape drive used. The size of the EOF marker, along with the total size of the tape, must be configured within the Backup System. (See section 6.3.1 for information on configuring the tape capacity and EOF marker size.)

6.2.10. Tape Coordinator Machines

Backups and restores are physically performed on a ***Tape Coordinator machine***. To qualify as a Tape Coordinator machine, a machine must

- Be in a physically secure location.
- Be an AFS client machine.
- Have one or more tape drives attached.
- Be configured as a Tape Coordinator machine (as described in section 6.3.2).
- Run one instance of the Tape Coordinator (initialized with the **butc** program described in section 6.4.3) for each tape drive.

You must start one Tape Coordinator for each tape drive you want to use. The Tape Coordinator must be active when you issue a **backup** command that involves the process's tape drive. Each Tape Coordinator controls the behavior of its associated drive and accepts service requests from the **backup** program.

The Backup System can accommodate a maximum of 58,511 Tape Coordinator/tape drive pairs at any time. A unique ***port offset number*** identifies each Tape Coordinator/tape drive. When you issue a **backup** command that involves a tape drive, specify the Tape Coordinator process associated with the drive by specifying its port offset number using the command's **-portoffset** argument.

6.2.11. The Backup Database

A large amount of the data used by the Backup System is stored in the ***Backup Database***. There is only one Backup Database per cell; however, a copy is stored on every database server machine. Synchronization of the Backup Database and its copies is controlled by the Ubik Coordinator (see section 2.13 for information about Ubik).

The Backup Database contains the following information:

- The **host name**/**port offset number** pair associated with each defined Tape Coordinator/tape drive on each Tape Coordinator machine.
- The **dump levels** associated with the dump hierarchy and the **expiration date** associated with each dump level.
- The **volume sets** that can be backed up and the **volume entries** that make up each volume set.
- A record of each **dump**, the name of each tape containing that dump, each volume contained in that dump, the dump level of the dump, the expiration date of the dump, and the dump ID of the initial dump in the dump set if the dump is an appended dump or an initial dump with one or more appended dumps.

6.3. Configuring the Backup System

Before the Backup System can be used to back up and restore data, certain configuration tasks must be performed. This section contains instructions for performing the following configuration tasks:

- Determining the tape size and EOF marker size of each tape drive.
- Configuring Tape Coordinator machines.
- Defining volume sets and volume entries.
- Configuring dump levels to create a dump hierarchy.
- Labeling tapes.

If you have already performed these tasks, you can skip this section and proceed to the instructions in section 6.6 for backing up volumes. If you need to check whether the configuration tasks have been performed, section 6.5 explains how to list information from the Backup Database.

6.3.1. Determining Information About Tape Capacity and EOF Marker Size

The size of a tape determines the amount of data the Backup System can place on it. The tape size varies for different tapes and tape drives. In addition, the Backup System appends an end of file (EOF) marker after each volume it dumps to tape. (Note that an EOF marker is appended after each volume, not after each file.) The size of the marker also affects the amount of space available for data on a tape. You must specify these values in the */usr/afs/backup/tapeconfig* file once for each tape drive (a tape's size is also stored on its label).

If you do not know the tape size or EOF marker size for a tape drive, use the **fms** (file mark size) command to determine these figures. This command produces screen output and an *fms.log* file in the current directory. Both the screen output and the log file list the tape capacity and the size of the EOF markers for the tape and tape drive pair.

The **fms** command requires that a tape be inserted in the tape drive about which information is to be determined before the command is issued. Because the command inserts file marks onto the entire tape in the drive, it can take from several hours to more than a day to complete.

Note: You should not use the **fms** command on data compressing tape drives because the command will give misleading numbers. Instead, you should configure the tape size to a number substantially larger than the tape itself. The end-of-tape (EOT) handling capacity of the Tape Coordinator will prevent dumps from failing when they hit the EOT.

⇒ To determine tape capacity and EOF marker size

Step 1: If an *fms.log* 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.

Step 2: Insert a tape into the drive. The command causes the tape to be overwritten. Use a blank tape or one that can be recycled.

Step 3: Enter the **fms** command.

% **fms** <*tape special file*>

where

fms must be typed in full.

tape special file specifies the UNIX device name of the tape drive for which tape size and EOF marker size are to be determined.

An example of the screen output for this command follows. The command also writes similar information to the *fms.log* file. In the example, the tape size for the tape drive named */dev/rmt0h* is 2136604672 bytes, and the EOF marker size for the drive is 1910205 bytes.

```
% 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
```

6.3.2. Configuring Tape Coordinator Machines

Note: Section 6.3.3 at the end of this section includes instructions for adding or removing a Tape Coordinator for an existing Tape Coordinator machine. The instructions in that section for adding a Tape Coordinator assume the machine to which the Tape Coordinator is to be added is already configured as a Tape Coordinator machine.

To configure a Tape Coordinator machine, you must perform the following primary tasks. Perform each task once when you initially set up a Tape Coordinator machine.

- Install the tape drives.
- Create the */usr/afs/backup/tapeconfig* file; include a line defining the tape parameters for each drive.
- Create an entry in the Backup Database, using the **backup addhost** command, for each Tape Coordinator that you intend to use with the AFS Backup System.

The steps in this section detail the order in which you need to perform all tasks.

⇒ To configure Tape Coordinator machines

Step 1: Install one or more tape drives on the Tape Coordinator machine according to the manufacturer’s instructions. The Backup System can track a maximum of 58,511 tape drives per cell.

Step 2: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 3: Verify that the individuals who are to use the Backup System are included in the */usr/afs/etc/UserList* file (if necessary, issue **bos listusers** to check). It is assumed the same *UserList* file is propagated to all file server machines in the cell. To add someone to the file, you should issue the **bos adduser** command.

Step 4: Verify that the binary files for **backup**, **butc**, and **fms** are in */usr/afsws/etc*. These files should already be present as a result of installation. If they are not, consult the chapter in the *AFS Installation Guide* on installing client machines.

% **ls /usr/afsws/etc**

Step 5: Become "root" in the Tape Coordinator machine's UNIX file system by issuing the **su** command.

Step 6: Create the */usr/afs* directory. (If the Tape Coordinator machine is also a file server, this directory should already exist.)

mkdir /usr/afs

Step 7: Create the */usr/afs/backup* directory on the Tape Coordinator machine.

mkdir /usr/afs/backup

Step 8: Create the */usr/afs/backup/tapeconfig* file on the Tape Coordinator machine with a text editor. Include a single line in the file for each tape drive attached to the machine; each line should contain the following elements, separated by a space:

- **The default size of tapes** used in the tape drive. The Tape Coordinator uses this capacity as the default size of tapes used in the drive. It is recommended that you decrease the actual tape capacity by 10 to 15% to allow for tape variations. The following abbreviations can be used for the tape size unit of measurement (the default is kilobytes). Do not leave a space between the number and the letter.

  – Kilobytes: `k` or `K` (for example, `2k` or `2K`)

  – Megabytes: `m` or `M` (for example, `2m` or `2M`)

  – Gigabytes: `g` or `G` (for example, `2g` or `2G`)

- **The end-of-file (EOF) marker size** for the tape drive. The Backup System appends an EOF marker after each volume dumped to tape. The size of this marker affects the amount of space available for backup data. EOF marker size can vary from 2 kilobytes to more than 2 megabytes, depending on the type of tape drive used. It is recommended that you increase the actual file marker size by 10 to 15% to allow for tape variations.

  If you do not specify a unit of measurement, the default unit used for the EOF size is bytes (not kilobytes, as for tape capacity). To indicate other units, use the same abbreviations as used for tape capacity.

- **The UNIX device name** of the tape drive (for example, */dev/rmt1h*). The format of this name varies with each operating system.

- **The port offset number** of the Tape Coordinator associated with the tape drive. Because the Backup System can track a maximum of 58,511 drives per cell, legal values for port offset numbers are the integers *0* through *58510*. You need not assign the numbers in sequence and you can skip numbers. However, the port offset for any Tape Coordinator must be unique in the Backup Database. (The steps for adding the port offset number to the Backup Database follow later in this section.)

  Because the **backup** commands that require a port offset number always default to *0*, assign the port offset *0* to the Tape Coordinator for the drive you will use most often. This enables you to omit the port offset as often as possible.

If you do not know the tape size and EOF marker size for the tape drive, determine them using the **fms** command described in section 6.3.1.

Following is an example of the *tapeconfig* file for a machine with two drives. The tape size for each is 2 gigabytes; the EOF marker size is 1 megabyte. The respective port offset numbers of the two drives are *0* and *1*.

```
2g 1m /dev/rmt0h 0
2G 1M /dev/rmt1h 1
```

Step 9: Decide which user and group are to own the */usr/afs/backup* directory and */usr/afs/backup/tapeconfig* file. The files in the */usr/afs/backup* directory are protected in the standard UNIX fashion – with mode bits. Because a Tape Coordinator creates and writes to log and error files in that directory, a user must be able to write to files in that directory to start a Tape Coordinator (that is, he or she must have the UNIX write permission on the directory). Before configuring a Tape Coordinator machine, you must decide what user and group are to "own" the files in the directory. Several ownership options are available:

- Set the owner to the super-user "root." To issue the **butc** command to start a Tape Coordinator, a user must then log in as "root."
- Make a single administrator the owner. The administrator is then the only user who can start the Tape Coordinator.
- Create a special group and give that group the necessary permissions. Members of the group can then start the Tape Coordinator.
- Create a dummy administrative account. This allows several different administrators to start the Tape Coordinator.

In addition, all **backup** commands and the **butc** command require a user to be authenticated under an identity that appears in the */usr/afs/etc/UserList* file. This file should be the same on all AFS server machines.

Correct the UNIX mode bits on the directory and file, if necessary.

\# **chown** <*desired owner*> **/usr/afs/backup**
\# **chown** <*desired owner*> **/usr/afs/backup/tapeconfig**
\# **chgrp** <*desired group*> **/usr/afs/backup**
\# **chgrp** <*desired group*> **/usr/afs/backup/tapeconfig**
\# **chmod 774 /usr/afs/backup**
\# **chmod 664 /usr/afs/backup/tapeconfig**

Step 10: Log out as "root" from the machine's UNIX operating system by issuing the **exit** command or typing **^d** (Control-d).

Step 11: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 12: Enter the **backup addhost** command to create an entry in the Backup Database for each Tape Coordinator/tape drive pair. Repeat this command for each Tape Coordinator/tape drive pair.

% **backup addhost** **<***tape machine name***>** **[<***TC port offset***>]**

where

addh	is an acceptable abbreviation of **addhost**.
tape machine name	is the complete Internet host name of the Tape Coordinator machine.
TC port offset	is the port offset number of the tape drive on *tape machine name*; it must correspond to an entry in the *tapeconfig* file. You must provide this argument unless the default value of *0* is appropriate.

6.3.2.1. Using IBM RS/6000 Machines as Tape Coordinator Machines

To dump and restore data between a tape device on an IBM RS/6000 machine (system type *rs_aix32*) and tape drives on machines of other types, you must also ensure that the tape block size of the tape device on the IBM RS/6000 machine is set to *0*. A value of *0* for the tape block size of a drive on an IBM RS/6000 machine indicates a variable block size. The block size must be set to *0* to enable tape drives on machines other than RS/6000s to read tapes made on an RS/6000.

Use the **smit** program to verify or change the value of the tape block size for a tape device on an IBM RS/6000 machine. If necessary, perform the following steps to change the tape block size for a drive.

⇒ To change the tape block size on an IBM RS/6000 machine

Step 1: Become "root" in the Tape Coordinator machine's UNIX file system by issuing the **su** command.

Step 2: Issue the following command to change the tape block size for the appropriate tape drive to *0*.

chdev -l '*device_name***' -a block_size='0'**

where *device_name* is the name of the tape drive (for example, */dev/rmt0h*) on the IBM RS/6000 machine that is to be used with the AFS Backup System.

6.3.3. Adding and Removing Tape Coordinators

As explained in section 6.3.2, a separate Tape Coordinator is associated with each tape drive. You must assign each Tape Coordinator a port offset number and use that number in commands issued to the Tape Coordinator. All **backup** commands that use a tape drive have a **-portoffset** argument for that purpose.

The following sections detail the steps involved in adding or removing a Tape Coordinator. The steps for adding a Tape Coordinator assume you are adding the Tape Coordinator to a machine that is configured as a Tape Coordinator machine. If the machine is not already a Tape Coordinator machine, refer to section 6.3.2 for instructions on configuring the machine.

⇒ To add a Tape Coordinator

Step 1: Install the drive on the Tape Coordinator machine according to the manufacturer's instructions.

Step 2: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 3: Choose the port offset number for the drive. Enter **backup listhosts** to check previously assigned port offsets. See section 6.5 for more information about the output from the command.

`%` **backup listhosts**

where **listh** is an acceptable abbreviation of **listhosts**.

Step 4: Using a text editor, add a single line defining the new Tape Coordinator to the *tapeconfig* file. The line should contain the following information:

- **The default size of tapes** used in the tape drive. Use the abbreviations `k` or `K` to signify kilobytes, `m` or `M` to signify megabytes, or `g` or `G` to signify gigabytes. Do not leave a space between the number and the letter (for example, `20m` or `3G` are correct values). The default is kilobytes. Use a number 10 to 15% lower than the actual tape capacity to allow for tape variations.
- **The end-of-file (EOF) marker size** for the tape drive. Use the abbreviations `k` or `K` to indicate kilobytes, `m` or `M` to indicate megabytes, or `g` or `G` to indicate gigabytes. Do not leave a space between the number and the letter (for example, `4k` or `2M`). If you do not specify a number, the default is <u>bytes</u> (not kilobytes). Use a number 10 to 15% greater than the actual EOF size to allow for tape variations.
- **The UNIX device name** of the tape drive (for example, */dev/rmt1h*). The format of this name varies with each operating system.
- **The port offset number** of the Tape Coordinator associated with the tape drive. Because the Backup System can track a maximum of 58,511 tape drives per cell, valid port offset numbers are the integers *0* through *58510*. You need not assign the numbers in sequence and you can skip numbers. The drive with the port offset of *0* is used by default.

Refer to section 6.3.1 for information about the **fms** command, which you can use to determine the tape capacity and EOF marker size for a drive.

Step 5: Enter the **backup addhost** command to create an entry for the Tape Coordinator in the Backup Database.

% **backup addhost <***tape machine name***>** **[<***TC port offset***>]**

where

addh is an acceptable abbreviation of **addhost**.

tape machine name is the complete Internet host name of the Tape Coordinator machine.

TC port offset is the same port offset number specified for the Tape Coordinator in the *tapeconfig* file. You must provide this argument unless the default value of *0* is appropriate.

⇒ To remove a Tape Coordinator

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Using a text editor, remove the line defining the Tape Coordinator from the */usr/afs/backup/tapeconfig* file.

Step 3: Enter the **backup delhost** command to delete the entry for the Tape Coordinator from the Backup Database.

% **backup delhost <***tape machine name***>** **[<***TC port offset***>]**

where

delh is an 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* is appropriate.

6.3.4. Defining Volume Sets and Volume Entries

Before performing any backups, you must create one or more volume sets in the Backup Database. A volume set defines a group of one or more volumes that can be dumped. Volume set names can be no longer than 31 characters and they can include any characters except a period.

Once you define a volume set, you must then define the volume entries in the set. Each volume entry is defined in terms of one or more volumes and the location of each volume on a file server machine and partition. Each volume entry consists of three fields, with each field separated by a space.

- ***file server machine name***: The complete name of the file server machine that houses the volume. You can either specify a single file server machine name or use the .* metacharacter expression to specify all file server machines in the VLDB.
- ***partition name***: The complete name (for example, */vicepx*) of the partition on which the volume resides. You can either specify a single partition name or use the .* metacharacter expression to specify all partitions in the VLDB.
- ***volume name***: The name of the volume(s) to be backed up. You can use any combination of regular characters and metacharacters for this field. The following metacharacters can be used in the volume field:

The period (.)	matches any single character.
Square brackets ([ ])	when placed around a list of characters, match a single instance of any of the characters. For example, *[abc]* matches one *a*, one *b*, or one *c* (but not *d* or *A*).
The caret (^)	when the first character within square brackets, causes the expression to match any single character other than the characters following the caret. For example, [^*a*] matches any single character except lowercase *a*.
The asterisk (*)	matches any number of the preceding character or expression. It can be combined with any other expression.
The backslash (\\)	preceding or double quotes surrounding any of the other metacharacters makes them literal. For example, use *usr** or *"usr*"* to identify a volume named *usr**. (Using metacharacters in volume names is not recommended.)

When using the metacharacters in a regular expression outside interactive mode, you must surround the entire expression with double quotes; for example, *"user.*\.backup"* to indicate all volumes that begin with the prefix *user* and end with the extension *.backup*. Otherwise, the command shell attempts to resolve the metacharacters rather than passing them to the **backup** program for resolution.

All volume sets must be unique within the Backup Database of the local cell. Create and delete volume sets with the **backup addvolset** and **backup delvolset** commands. Create and delete volume entries with the **backup addvolentry** and **backup delvolentry** commands.

6.3.4.1. Suggestions for Creating Volume Entries

The **backup addvolentry** command has arguments that correspond to the three fields in a volume entry: **-server** for the file server machine name field, **-partition** for the partition name field, and **-volume** for the volume name field. By combining these arguments in different ways, you can create volume entries for different groupings of volumes. To create entries that include

- All volumes in the cell's file system, use the .* metacharacter expression for all three arguments (for example, *".*" ".*" ".*"*).
- Every volume on a specific file server machine, specify its name with **-server** and the .* metacharacter expression for **-partition** and **-volume** (for example, *fs1.transarc.com ".*" ".*"*).

- All volumes on partitions of the same name, specify the partition name with **-partition** and the .* metacharacter expression for **-server** and **-volume** (for example, *".*" /vicepa ".*"*).
- Every volume with a common string of letters (such as a *.backup* extension), specify the .* metacharacter expression for **-server** and **-partition**, and a regular character/metacharacter combination for **-volume** (for example, *".*" ".*" ".*\.backup"*).
- All volumes on a partition, specify the machine name with **-server**, the partition name with **-partition**, and the .* metacharacter expression for **-volume** (for example, *fs2.transarc.com /vicepb ".*"*).

A volume entry should contain only those volumes you wish to dump to the same tape at the same time (for example, weekly or daily) and in the same manner (fully or incrementally). The two most common types of volume sets include those based on a common volume location (file server machine and partition) and those based on similar contents (as reflected by a volume name). Because volumes can be moved between machines and partitions, it is better to use name-based volume sets rather than location-based ones. Therefore, you should usually specify the .* metacharacter expression for the **-server** and **-partition** arguments in the **backup addvolentry** command.

Do not include a large number of volumes in a volume entry. If the entries in a volume set contain a large number of volumes, dumps of the volume set can take a long time to complete, increasing the chance that the operation will fail due to a service interruption or outage while it is underway.

⇒ To add a volume set to the Backup Database

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Enter **backup addvolset** to create the volume set. The volume set remains empty until you use **backup addvolentry** to define entries in it.

% **backup addvolset** <*volume set name*>

where

addvols is an acceptable abbreviation of **addvolset**.

volume set name is the name of the volume set to be added. The name can include no more than 31 characters and it must be unique within the Backup Database. You should not include periods in the name.

⇒ To add a volume entry to a volume set

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following command assumes regular command mode is used.

Step 3: Enter **backup addvolentry** to define volume entries in a volume set previously created with **backup addvolset**. The Backup System automatically assigns an index number to each entry (starting with *1* for the first entry in a volume set). This number is used to remove a volume entry if necessary.

% **backup addvolentry** <*volume set name*> <*machine name*>
<*partition name*> <*volume name (regular expression)*>

where

addvole is an acceptable abbreviation of **addvolentry**.

volume set name specifies the volume set to which the entry is to be added.

machine name is the full Internet host name of the file server machine that houses the volume(s) in the entry. The ".*" metacharacter expression can be used to match the names of all file server machines.

partition name is the name of the partition that houses the volume(s) in the entry. The ".*" metacharacter expression can be used to match the names of all partitions.

volume name (regular expression)
specifies the volume(s) to be included in the volume entry. The ".*" metacharacter expression or any combination of regular characters and metacharacters described previously can be used to match the names of multiple volumes.

⇒ To delete a volume set from the Backup Database

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Issue **backup delvolset** to delete each volume set. It is not necessary to delete the volume entries in each volume set first; volume entries are automatically deleted when the volume set to which they belong is deleted.

% **backup delvolset** <*volume set name*>$^+$

where

delvols is an acceptable abbreviation of **delvolset**.

volume set name specifies each volume set to be deleted.

⇒ To delete a volume entry from a volume set

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Enter **backup listvolsets** to determine the index number of the entry you want to delete. This is necessary only if the volume set contains multiple entries, in which case this command is used to list the entries. If the set contains a single entry, the index is *1*. See section 6.5 for more information about the output from the command.

% **backup listvolsets** <*volume set name*>

where

listv	is an acceptable abbreviation of **listvolsets**.
volume set name	specifies the volume set whose entries are to be listed.

Step 3: Enter **backup delvolentry** to delete the volume entry.

% **backup delvolentry** <*volume set name*> <*volume entry index*>

where

delvole	is an acceptable abbreviation of **delvolentry**.
volume set name	specifies the volume set containing the entry to be deleted.
volume entry index	is the index number of the entry to be deleted.

6.3.5. Creating Dump Hierarchies

A dump hierarchy is defined by the dump levels that compose it. Following are some general issues to consider when defining dump levels:

- The name of a dump level can have any number of levels. The slash is used to separate different levels in the dump hierarchy. Regardless of its level in the dump hierarchy (full or incremental), each level in a dump level name must be preceded by a slash.
- You should not include a period in the name of a dump level. When a dump is transferred to tape, the dump level name becomes part of the dump name and the elements of the dump name (the volume set name and the last element of the dump level name) are joined by a period.
- Any characters other than a period can be included in a dump level name. Shell metacharacters included in a name must be properly escaped with a backslash or double quotes.
- The maximum length for any single level in a dump level name is 28 characters. This does not include the slash that precedes the level.
- The maximum length for the complete name of a dump level, full or incremental, is 256 characters. This includes any slashes that are part of the name.

6.3.5.1. Defining Expiration Dates for Dump Levels

Each dump level is associated with an expiration date that defines the date and time at which a dump created at that level expires. Expiration dates are initially defined for dump levels with the **backup adddump** command. They can be changed with the **backup setexp** command. Both commands have an **-expires** option used to set the expiration date.

Expiration dates can be specified in relative or absolute values.

- ***Absolute expiration dates*** use the optional keyword "**at**" to represent a specific date and, optionally time, to use as the expiration date. Absolute values are expressed as

  [**at**] <*mm/dd/yy*> [<*hh:mm*>]

  or

  [**at**] NEVER

  If you specify a time, you must use military time (24-hour format). For example, *at 11/22/94 18:00* specifies an expiration date and time of November 22, 1994 at 6:00 p.m. If no time is provided, a default time of 00:00 (12:00 a.m.) on the indicated date is used.

  Use NEVER to indicate that a dump backed up at the corresponding dump level should never expire.

- ***Relative expiration dates*** use the optional keyword "**in**" to indicate a number of years, months, and/or days to be added to the current date and time to calculate the expiration date. When the system creates a dump at this level, it calculates the date and time when the dump expires by adding the values to the start time of the operation. Relative values are expressed as

  [**in**] [<*integer*>**y**] [<*integer*>**m**] [<*integer*>**d**]

  At least one value must be provided. Multiple values must be listed in the order shown, with the appropriate unit abbreviation (**y**, **m**, or **d**) used with each value. For example, *in 1y 6m 2d* causes the system to add 1 year, 6 months, and 2 days to the current date and time to calculate the expiration date.

You can omit the **-expires** argument to specify that no expiration date is to be associated with a dump level; <u>however, using expiration dates is strongly recommended to prevent premature recycling of the dump set</u>. Dumps created at a dump level that has no expiration date are assigned an expiration date of 00:00 on 1 January 1970 (the UNIX time equivalent of 0). They are considered to be expired as soon as they are dumped.

⇒ To add a dump level

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following command assumes regular command mode is used.

Step 3: Issue the **backup adddump** command to define one or more dump levels. Remember that each parent dump level must already exist.

% **backup adddump -dump** <*dump level name*>$^+$
[-expires <*expiration date*>**]**

where

addd is an acceptable abbreviation of **adddump**.

dump level name names each new dump level to be added. If you specify more than one dump level name, you must include the **-dump** switch.

Provide the entire pathname of the dump level. Assign a meaningful name to the dump level. Precede each level in the pathname with a slash.

expiration date sets the expiration date associated with each *dump level name*. The expiration date is applied to all specified dump levels.

You can specify a relative or absolute expiration date; see section 6.3.5.1 for more information about the two types of dates. Omit this argument to assign no expiration date to each dump level.

Note: Although the usage message for this command indicates that multiple instances can be included for this argument, it is not possible to specify more than one expiration date. The plus sign follows the *expiration date* argument because a multi-word argument is required to define an expiration date, not because multiple dates can be specified.

⇒ To change a dump level’s expiration date

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following command assumes regular command mode is used.

Step 3: Issue the **backup setexp** command to change the expiration date(s) of one or more dump levels.

% **backup setexp -dump** <*dump level name*>$^+$
[-expires <*expiration date*>**]**

where

se is an acceptable abbreviation of **setexp**.

dump level name names each existing dump level whose associated expiration date is to be changed.

expiration date sets the expiration date associated with each *dump level name*. The expiration date is applied to all specified dump levels.

You can specify a relative or absolute expiration date; see section 6.3.5.1 for more information about the two types of dates. Omit this argument to assign no expiration date to each dump level. Dump levels with no expiration dates are treated as expired as soon as they are dumped.

Note: Although the usage message for this command indicates that multiple instances can be included for this argument, it is not possible to specify more than one expiration date. The plus sign follows the *expiration date* argument because a multi-word argument is required to define an expiration date, not because multiple dates can be specified.

⇒ To delete a dump level

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following command assumes regular command mode is used.

Step 3: Issue the **backup deldump** command to delete the dump level. <u>All dump levels for which this level serves as the parent, either directly or indirectly, are also deleted automatically.</u>

% **backup deldump** <*dump level name*>

where

deld is an acceptable abbreviation of **deldump**.

dump level name is the complete pathname of the dump level to be deleted.

6.3.6. Labeling Tapes

As mentioned in section 6.2.9, a tape's magnetic label can contain:

- The complete name of the tape.
- An indicator of "" or NULL, which can be created only when the **backup labeltape** command is issued without specifying a tape name. The Backup System replaces the empty indicator with the correct name when it puts a dump onto the tape.
- No label, indicating the tape is unused. Again, the Backup System generates the correct name when it transfers a dump onto the tape. (In this case, the tape contains no label.)

It is not essential to prelabel tapes before data is transferred to them; the Backup System can use unlabeled, partially labeled, or fully labeled tapes. However, you need to relabel a tape if

- You want to create a new dump set on the tape and the name of the initial dump for the new dump set is different from the name of the initial dump of the current dump set. The Backup

System will not allow the creation of a new dump set on a tape containing an old dump set if the names of the initial dumps do not match.

- You want to reuse a tape before the expiration date associated with its dump set has expired. The Backup System will not allow you to recycle a tape if it houses a dump set containing one or more unexpired dumps.
- The tape capacity of the tape is different from the default capacity for the tape drive that you intend to use. The default tape capacity for a tape drive is recorded in the */usr/afs/backup/tapeconfig* file.

⇒ To label a tape

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup labeltape** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: Issue the **backup labeltape** command to label the tape.

% **backup labeltape** **[**<*tape name, defaults to NULL*>**]**
[-size <*tape size in Kbytes, defaults to size in tapeconfig*>**]**
[-portoffset <*TC port offset*>**]**

where

la is an acceptable abbreviation of **labeltape**.

tape name, defaults to NULL
is the new name of the tape. It must have the form <*volume set name*>.<*dump level name*>.<*tape index*>. Omit this option to label the tape as empty with the NULL indicator.

tape size in Kbytes, defaults to size in tapeconfig
is the capacity of the tape. Specify a number and a letter. Add a `k` or `K` to the number to indicate kilobytes, an `m` or `M` to indicate megabytes, or a `g` or `G` to indicate gigabytes. The default unit of measure is kilobytes.

Specify a size only if the size of the tape is different from what is listed for the drive in the */usr/afs/backup/tapeconfig* file. If you omit this argument, the size specified for the drive in the *tapeconfig* file is recorded in the label as the size of the tape (even if the tape was previously labeled, in which case it already has a size recorded in its label).

TC port offset is the port offset number of the drive containing the tape to be labeled. You must provide this argument unless the default value of *0* is appropriate.

Step 5: Place the tape to be labeled in the drive, and press the **Return** key on the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, the Tape Coordinator will not prompt you to insert a tape.)

⇒ To read the label on a tape

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup readlabel** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: Issue the **backup readlabel** command to read the label on the tape.

% **backup readlabel [**<*TC port offset*>**]**

where

rea is an acceptable abbreviation of **readlabel**.

TC port offset is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

Step 5: Place the tape to be read in the drive, and press the **Return** key on the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, the Tape Coordinator will not prompt you to insert a tape.)

The following example reads the label on the tape in the drive with port offset *1*:

```
% backup readlabel 1
Tape read was labelled: guests.monthly.1 (697065340)
     size: 2150000 KBytes
```

The example displays the following in the Tape Coordinator window for the Tape Coordinator with port offset *1*:

```
Tape label
----------
name =           guests.monthly.1
creationTime =    Fri Feb  7 05:31:32 1992
cell =           transarc.com
size =           2150000 Kbytes
dump path = /monthly
dump id = 697065340
useCount = 44
-- End of tape label --
```

6.4. Interfaces to the Backup System

To run **backup** commands that interact with a Tape Coordinator, two windows must be open: a window to the Tape Coordinator and a command shell window. To run **backup** commands that do not interact with a Tape Coordinator, only the command shell window is required.

- Backup commands can be issued from any AFS client machine by opening a command shell window. You can run **backup** commands in either interactive or regular command mode; see section 6.4.2 for more information.
- A separate window to a Tape Coordinator machine must be opened for each Tape Coordinator/tape drive that you intend to use. The **butc** command is used to open this window. The Tape Coordinator window must remain open while the Tape Coordinator runs. See section 6.4.3 for information on starting a Tape Coordinator.

Note: The version of the **butc** and **buserver** processes must be the same for the Backup System to function properly.

6.4.1. A Note on Performing backup Commands in a Foreign Cell

It is possible to run the **butc** command and the **backup** commands in a foreign cell. When determining what cell to execute these commands, the Backup System will look first at the value of the environment variable AFSCELL. If this variable is undefined, it will look at the cell named in */usr/afs/etc/ThisCell*. (The **butc** command also has a **-cell** argument which will override the values specified by both AFSCELL and the contents of the */usr/afs/etc/ThisCell* file.)

The issuer of the **butc** command an the **backup** commands must be an authenticated user listed in the foreign cell's */usr/afs/etc/UserList*.

6.4.2. The Command Shell Window and Interactive Mode

You can issue **backup** commands in regular (non-interactive) command mode or in ***interactive mode***. Regular command mode is the manner in which most AFS commands are issued; in regular command mode, you type the command suite and operation code to execute a command (for example, **backup dump** to execute the **backup dump** command.)

When you enter interactive mode (by typing **backup** at the command prompt), you establish a temporary connection with the Backup Server and the VL Server. In interactive mode, the `backup>` prompt replaces the usual command shell prompt; to execute a **backup** command, you simply type the operation code of the command along with any arguments or flags. You cannot execute commands from other suites in interactive mode.

6.4.2.1. The Advantages of Using Interactive Mode

Interactive mode has several advantages over regular command mode. In interactive mode

- You do not need to enclose shell metacharacters in double quotes. In regular command mode, you must do so to prevent the command shell from resolving them.
- You can track current and pending operations with the **(backup) jobs** command. In regular command mode, you cannot use the **(backup) jobs** command.
- You can cancel current and pending operations with the **(backup) kill** command. In regular command mode, you cannot use the **(backup) kill** command.

Most of the operations described in this chapter are presented in regular command mode. Where appropriate, operations include steps introduced as "*Optional*" to indicate where working in interactive mode could be useful. The **(backup) jobs** and **(backup) kill** commands described in section 6.4.2.4 can be issued only in interactive mode.

6.4.2.2. Job IDs and Task IDs

Two significant types of ID numbers are associated with the Backup System's interactive mode:

- A ***job ID*** number is assigned to each **backup** operation when the operation is initiated. Job IDs are temporary numbers used to track and cancel current and pending operations. Job ID numbers can be viewed in interactive mode with the **(backup) jobs** command. They can be used to kill a current or pending operation in interactive mode with the **(backup) kill** command. See section 6.4.2.4 for more information about these two commands.
- A ***task ID*** is assigned to most tape operations. Each Tape Coordinator maintains its own task ID numbers and records the task ID number with the operation information it writes to its log and error files. To view the task ID numbers associated with a specific Tape Coordinator, run the **backup status** command.

6.4.2.3. Entering and Exiting Interactive Mode

Entering and exiting interactive mode are simple tasks.

⇒ To enter interactive mode

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Enter **backup** at the system prompt. The interactive prompt will appear.

```
% backup
backup>
```

⇒ To leave interactive mode

Step 1: Enter the **quit** command at the `backup>` prompt. The system prompt will appear.

```
backup> quit
%
```

Note: You can only exit interactive mode if all jobs issued during the session are complete.

6.4.2.4. Canceling Operations in Interactive Mode

While in interactive mode, you can use the **(backup) jobs** command to list job ID numbers and status information about executing or pending operations started during the current interactive session. You can use the job ID number of an operation (or its dump name, if it is a dump operation) to abort an executing or pending operation with the **(backup) kill** command.

- If you abort a dump operation while it is executing, all volumes written to tape *before* the kill signal is received are complete and usable on the tape. An entry will appear in the Backup Database for the aborted dump and for the volumes that were successfully dumped.

  There are three main ways to handle the remaining volumes of a volume set that are not backed up because of an aborted dump:

  – You can pick up the remaining volumes during the next regularly scheduled dump. This approach will create a temporary "hole" in the coverage provided by the Backup System because you will not pickup the missed data until the next regularly scheduled dump.

  – You can discard the entire dump set and run the dump again. To discard the dump set, relabel the tape using the **backup labeltape** command; see section 6.3.6 for more information on relabeling a tape.

  – You can create a temporary volume set for the missed volumes and dump those volumes in a temporary dump set. The missed volumes can be merged back into the original dump set during the next regularly scheduled dump.

- If you abort a restore operation while it is executing, all completely restored volumes are online and usable. However, because most restore operations require data from multiple tapes (a full dump tape and one or more incremental dump tapes), most volumes are usually not completely restored. If the kill signal occurs before the system accesses all of the necessary tapes, most volumes are not restored to the desired date or version and should not be used.

  If an interrupted restore operation is overwriting one or more existing volumes, the volumes can be lost entirely. However, the data being restored still exists on tape. To avoid inconsistencies that can result from an interrupted restore operation, reissue the restore command immediately.

⇒ To cancel operations in interactive mode

Step 1: Issue the **jobs** command to determine the job ID number of the operation to be killed.

`backup>` **jobs**

where **j** is an acceptable abbreviation of **jobs**.

If operations are pending or executing, the following information is displayed for each operation:

- The job ID number.
- One of the following names describing the operation:
  - `Dump (`*dump name*`)` for a dump operation initiated with the **backup dump** command. In the operation name, *dump name* is the name of the dump in the form <*volume set name*>.<*dump level name*>.
  - `Restore` for a restore operation initiated with the **backup volrestore** or **backup diskrestore** command.
  - `Labeltape (`*tape label*`)` for a tape labeling operation started with the **backup labeltape** command. In the operation name, *tape label* is the tape label specified with the **backup labeltape** command's arguments.
  - `Scantape` for a tape scanning operation initiated with the **backup scantape** command.
  - `SaveDb` for a **backup savedb** operation.
  - `RestoreDb` for a **backup restoredb** operation.
- For a dump operation, the number of kilobytes transferred from file system to tape. For a restore operation, the number of kilobytes transferred from tape to file system.
- For a dump operation, the string `volume` followed by the name of the volume currently being dumped. For a restore operation, the string `restore.volume`.
- One of the following messages indicating the status of the operation. No message is displayed if the operation is executing normally.
  - `[abort requested]` means the **(backup) kill** command was issued, but it has not been sent to the Tape Coordinator yet.
  - `[abort sent]` means the operation is canceled. Once the system removes it from the queue or stops its execution, the operation no longer appears in the listing from the **(backup) jobs** command.
  - `[butc contact lost]` means the **backup** program cannot access the Tape Coordinator. This can indicate that the Tape Coordinator executing the operation was killed or failed while the operation was executing or it can mean that the connection to the Tape Coordinator timed out. It can also mean simply that the Tape Coordinator is executing another operation. If this message continues to appear, determine whether the Tape Coordinator is

busy or idle, by running **backup status**, before deciding whether the Tape Coordinator needs to be restarted.

– `[drive wait]` means the operation is waiting for the specified tape drive to become free.

– `[operator wait]` means the Tape Coordinator is waiting for the operator monitoring the operation's execution to insert a tape in the drive.

The expiration date and time of the token used by the Backup Server process is also displayed. Without a valid token for the specified cell, no **backup** commands can be executed.

Any timed dumps along with the time that they are scheduled to execute are displayed. If a timed dump is scheduled to execute before the token expiration date, it will appear before the token expiration information; if a timed dump is scheduled to execute after the token expiration date, it will appear after the token expiration information. A timed dump that is scheduled to execute after the token expiration date will fail unless the token is renewed prior to the scheduled execution time.

Step 2: Issue the **(backup) kill** command to cancel the operation.

`backup>` **kill** <*job ID or dump set name*>

where

k is an acceptable abbreviation of **kill**.

job ID or dump set name
is either the job ID number of the operation to be canceled or the name of the dump being created in the form <*volume set name*>.<*dump level name*>. Specify a *dump set name* only if a dump operation is to be canceled.

6.4.3. The Tape Coordinator Window

Before performing an operation that requires a tape drive (such as a dump or restore), you must start the Tape Coordinator that is to execute the operation. This section explains how to start, stop, and check the status of a Tape Coordinator for a tape drive.

See section 6.3.2 for information on configuring Tape Coordinator machines.

⇒ To start a Tape Coordinator

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Verify that you can write to the Tape Coordinator's log (*TL_device_name*) and error (*TE_device_name*) files in the */usr/afs/backup* directory. 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.

Step 3: Open a window to the Tape Coordinator machine that contains the tape drive to be used (using **telnet**, **rlogin**, etc).

Step 4: Issue the **butc** command to start the Tape Coordinator window. This window must remain open while the Tape Coordinator runs.

% **butc [**<*port offset*>**] [-debuglevel** <*trace level*>**] [-cell** <*cellname*>**]**
[-aixscsi] [-noautoquery]

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* is appropriate.

trace level specifies the type of trace messages the Tape Coordinator is to produce on standard output (stdout). There are two valid values:

- *0* indicates the Tape Coordinator is only to prompt for new tapes as necessary. This is the default value.
- *1* indicates the Tape Coordinator is to report on the major activities it performs (for example, the volumes it accesses and dumps), in addition to prompting for new tapes as necessary.

cellname specifies the cell with respect to which the Tape Coordinator is to run. You must **klog** to the specified cell before using this argument. By default, the local cell of the Tape Coordinator machine is used.

The specified Tape Coordinator communicates with the Backup Server in the indicated cell. It can manipulate data only in the specified cell.

-aixscsi should be used only if the Tape Coordinator machine is running AIX 2.2.1 and has a driver that provides extended SCSI support, in which case its use prevents the Tape Coordinator from resetting before rewinding tapes.

-noautoquery disables the initial prompt for tape insertion. See section 6.7.1 for a complete description of this feature, its benefits, and the affected **backup** commands.

⇒ To stop a Tape Coordinator

Step 1: When you are finished using a Tape Coordinator, stop it by entering an interrupt signal such as **^c** (Control-c) in the Tape Coordinator window.

⇒ To check the status of a Tape Coordinator

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Issue the **backup status** command to check the status of a Tape Coordinator.

% **backup status [**<*TC port offset*>**]**

where

st is an 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* is appropriate.

If the Tape Coordinator is not currently performing an operation, the following message will appear:

```
Tape Coordinator is idle.
```

If no Tape Coordinator is running for the specified *TC port offset*, the following message will appear:

`Unable to connect to tape coordinator at port` <*TC port offset*>`.`

Otherwise, the output lists the following information:

- The task ID number associated with the operation.
- A name describing the operation the Tape Coordinator is performing.
- The number of kilobytes transferred from file system to tape (for a dump operation) or from tape to file system (for a restore operation).
- The string `volume` followed by the name of the volume currently being dumped (for a dump operation) or the string `restore.volume` (for a restore operation).
- A message reporting additional status information about the operation currently being performed. No message is displayed if the operation is proceeding normally.

6.5. Listing Backup Information

The following commands can be used to list information from or about the Backup Database:

- **backup dbverify** checks the status of the Backup Database.
- **backup listvolsets** lists volume sets and their volume entries.
- **backup listdumps** displays the dump hierarchy.
- **backup dumpinfo** displays information about dumps.
- **backup listhosts** lists the Tape Coordinators and their port offset numbers.
- **backup volinfo** displays the dump history for a volume.

The **backup scantape** command reads a tape and extracts information from its label and from the volume header of each volume it contains. The command can also detect damage to a tape. If it encounters damage, the command scans no more information from that tape; it continues with the next tape to be scanned. (The **backup scantape** command cannot be used to scan Backup Database dump tapes.)

Interactive mode can be entered prior to issuing any of the commands in this section. The advantages of interactive mode are described in section 6.4.2. The commands in this section assume regular command mode is used, not interactive mode.

Note: All of the commands in this section require that you be included in the */usr/afs/etc/UserList* file.

⇒ To verify the Backup Database

Step 1: Issue the **backup dbverify** command to check the status of the Backup Database.

% **backup dbverify [-detail]**

where

db is an acceptable abbreviation of **dbverify**.

-detail specifies that some additional information about the Backup Database is to be displayed. The extra information is not necessary to determine whether the database is damaged. See the *AFS Command Reference Manual* for more information about this flag.

The output from this command reports whether the Backup Database is damaged. If it is, refer to section 6.9 for information on restoring the database from tape. The following output reports that the database is undamaged:

```
% backup db
Database OK
```

⇒ To list volume sets and volume entries

Step 1: Issue the **backup listvolsets** command to view volume sets and their volume entries.

% **backup listvolsets [**<*volume set name*>**]**

where

listv is an acceptable abbreviation of **listvolsets**.

volume set name is the name of a volume set whose entries are to be displayed. Omit this argument to display the names of all volume sets and their entries currently defined in the Backup Database.

The output from this command displays the names of volume sets and their volume entries currently defined in the Backup Database. The following output shows all volume sets and volume entries:

```
% backup listv
Volume set user:
  Entry 1: server .*, partition .*, volumes:
           user.*\.backup
Volume set sun:
  Entry 1: server .*, partition .*, volumes: sun3_411\..*
  Entry 2: server .*, partition .*, volumes: sun4c_411\..*
Volume set rt:
  Entry 1: server .*, partition .*, volumes: rt_aos4\..*
```

⇒ To display the dump hierarchy

Step 1: Issue the **backup listdumps** command to display the dump hierarchy.

% **backup listdumps**

where **listd** is an acceptable abbreviation of **listdumps**.

The output from this command displays the dump hierarchy in the Backup Database and the expiration dates associated with each dump level:

```
% 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
```

⇒ To display backup information

Step 1: Issue the **backup dumpinfo** command to list information about dumps recorded in the Backup Database.

% **backup dumpinfo [**<*no. of dumps*>**] [-id** <*dump id*>**] [-verbose]**

where

dumpi is an acceptable abbreviation of **dumpinfo**.

no. of dumps specifies the number of dumps about which information is to be displayed. Information is displayed about the most recent *no. of dumps*. Use this argument or use *dump id*. Omit both arguments to display information about the last 10 dumps.

dump id specifies the unique dump ID number of a specific dump about which information is to be displayed. Use this argument or use *no. of dumps*. Omit both arguments to display information about the last 10 dumps.

-verbose includes more information about the dump specified with *dump id*. Use this flag only if a *dump id* is specified.

If *no. of dumps* is specified, the following information is provided about each displayed dump:

- `dumpid`: The dump ID number.
- `parentid`: The dump ID number of the dump's parent dump. A value of `0` indicates this is a full dump and therefore has no parent.
- `lv`: The depth in the dump hierarchy of the dump level used to create the dump. A value of `0` indicates a full dump, in which case `parentid` should also be `0`. A value of `1` or greater indicates an incremental dump made at the specified 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 required to record the information in the dump. A value of `0` indicates a dump operation that was aborted or that failed, but for which an entry still exists in the database.
- `nvols`: The number of volumes from which information was included in the dump. A value of `0` indicates a dump operation that was aborted or that failed, but for which an entry persists in the database; `nt` should also be `0`. If a volume spans tapes, it is counted twice.
- `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 dumped volume set and *dump level name* is the name of the last element in the dump level at which the volume set was dumped.

  The *initial dump ID*, if one is displayed, contains the dump ID of the initial dump in the dump set and tells you that that dump set contains two or more dumps. If there is no *initial dump ID*, the dump is the initial dump in a dump set that contains no appended dumps.

If just the **-id** is specified, the command displays the following information:

- The first line, labeled `Dump:`, displays the following information about the specified tape:
  - The dump ID number.
  - The depth in the dump hierarchy of the dump level used to create the dump. A value of `0` indicates a full dump, in which case `parentid` should also be `0`. A value of `1` or greater indicates an incremental dump made at the specified level in the dump hierarchy.

  – The number of volumes in the dump.

  – The creation date of the dump.

- The second line, labeled `Tape:`, displays the tape name containing the dump followed by the dump ID of the initial dump of the dump set (in parentheses).
- The third line displays the number of volumes in the dump and the creation date of the dump.
- The fourth line contains header labels for the remaining lines in the output. The data in these columns include the following information about the volumes in this dump for the tape listed in the second line.
  - `Pos`: The relative position of each volume on the corresponding tape beginning with position *2*. (Position *1* is used by the tape label.)
  - `Clone time`: For a Backup or ReadOnly volume, this is the time at which it was cloned from its ReadWrite source. For a ReadWrite volume, this is the same as the creation date for the corresponding dump.
  - `Nbytes`: The size of each volume in bytes.
  - `Volume`: The name of each volume.

If both **-id** and **-verbose** are specified, the command displays the following information:

- The heading `Dump` followed by a number of lines displaying detailed information about the specified dump.
- The heading `Tape` followed by a number of lines displaying detailed information about the tape containing the specified dump.
- Zero or more occurrences of the heading `Volume`, each of which is followed by detailed information about one of the volumes included in the specified dump.

The following command displays information about the last five dump sets recorded in the Backup Database:

```
% backup dumpi 5
   dumpid   parentid lv created        nt nvols dump name
690169399          0 0  02/02/92 04:12  1    22 usr.sun (690169399)
690428867  690169399 1  02/05/92 05:29  1    62 usr.wed (690169399)
690514342  690169399 1  02/06/92 05:21  1    46 usr.thu (690169399)
690600291  690169399 1  02/07/92 05:23  1    58 usr.fri (690169399)
690874954          0 0  02/09/92 05:31  2    73 sys.week
```

⇒ To list information about Tape Coordinators

Step 1: Issue the **backup listhosts** command to list the Tape Coordinators and their port offset numbers currently defined in the Backup Database.

% **backup listhosts**

where **listh** is an acceptable abbreviation of **listhosts**.

The output from this command lists each Tape Coordinator machine and the port offsets of its Tape Coordinators currently defined in the Backup Database. The appearance of a port offset number does not imply that the associated Tape Coordinator is actually running. Machine names are displayed in the format in which they were specified with **backup addhost**.

The following output shows the current Tape Coordinators:

```
% backup listhosts
Tape hosts:
    Host backup1.transarc.com, port offset 0
    Host backup1.transarc.com, port offset 2
    Host backup2.transarc.com, port offset 1
    Host backup2.transarc.com, port offset 3
```

⇒ To display a volume's dump history

Step 1: Issue the **backup volinfo** command to display the dump history of a volume.

`%` **backup volinfo** <*volume name*>

where

voli is an acceptable abbreviation of **volinfo**.

volume name is the name of the volume whose dump history is to be displayed. If the Backup or ReadOnly version rather than the ReadWrite version of the volume was dumped, include the `.backup` or `.readonly` extension with the name.

The output contains a line of information for each dump of the specified *volume name* contained in the Backup Database. The output is displayed in six columns:

- `dumpID`: The dump ID containing *volume name*.
- `lvl`: The depth in the dump hierarchy of the dump level at which *volume name* was dumped. A value of `0` indicates a full dump set. A value of `1` or greater indicates an incremental dump set made at the specified level in the dump hierarchy.
- `parentid`: The dump ID of the dump's parent dump. A value of `0` indicates a full dump set, which has no parent; in this case, `lvl` should also be `0`.
- `creation date`: The date and time at which the Backup System started the dump operation that created the dump set.
- `clone date`: For a Backup or ReadOnly volume, this is the time at which it was cloned from its ReadWrite source. For a ReadWrite volume, this is the same as the `creation date`.
- `tape name`: The name of the tape containing the dump set in the form <*volume set name*>.<*dump level name*>.<*tape index*> (*dump ID*) where *volume set name* is the name of the volume set associated with the initial dump in the dump set; *dump level name* is the name of the final element in the dump level at which the initial dump was backed up; *tape index* is the numerical position of the tape in the dump set; and *dump ID* is the ID of the initial dump in the dump set of which this tape is a part.

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
654998415 1   654925629 02/04/92 05:26 02/04/92 05:01 user.tue.1 (654912115)
654912115 1   654925629 02/03/92 05:25 02/03/92 05:05 user.mon.1 (654912115)
654825629 0           0 02/02/92 05:38 02/02/92 04:58 user.sun.1 (654825629)
    .     .      .         .       .       .      .         .
    .     .      .         .       .       .      .         .
```

⇒ To scan the contents of a tape

Step 1: If it is not already running, start the Tape Coordinator for the tape drive you want to use. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup scantape** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 2: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 3: Issue the **backup scantape** command to read information from the dump label(s) and volume header(s) on the tape to verify that the tape is undamaged. When using this command, you must insert tapes sequentially by index number.

% **backup scantape [-dbadd] [-portoffset <***TC port offset***>]**

where

sc	is an acceptable abbreviation of **scantape**.
-dbadd	specifies that information from the tape is to be added to the Backup Database. (Information on using this flag appears in section 6.9.2.)
TC port offset	is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

Step 4: Place the tape to be scanned in the drive and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape into the tape drive prior to executing the **backup scantape** command.)

The output from this command first lists information from the dump label and the volume headers of the tape. See the *AFS Command Reference Manual* for a detailed explanation of the command's output.

The following output, displayed in the Tape Coordinator window associated with port offset *2*, provides information on the tape in the drive with port offset *2*:

% **backup scantape 2**

```
Dump label
----------
name =           guests.monthly.1
creationTime =    Fri Feb  7 05:31:32 1992
cell =           transarc.com
size =           2150000 Kbytes
dump path = /monthly
dump id = 697065340
useCount = 44
-- End of dump label --

-- volume --
volume name: user.guest10.backup
volume ID 1937573829
dumpSetName: guests.monthly
dumpID 697065340
level 0
parentID 0
endTime 0
clonedate Fri Feb  7 05:03:23 1992
```

6.6. Backing Up Data

The **backup dump** command is used to back up AFS data to tape. When you enter the command, specify the volume set to be dumped and the dump level at which it is to be dumped. All entries in the specified volume set are dumped at the dump level you specify. If you specify a full dump level, all files in all volumes in the specified volume set are dumped; if you specify an incremental dump level, only those files that have changed since the volumes were dumped at the corresponding parent level are dumped.

When a volume set is dumped to tape, the resulting dump can either create a new dump set or be appended to an existing dump set. A new dump set is created by default when using the **backup dump** command. To append a dump to an existing dump set, you must use the **-append** flag with the **backup dump** command. (Appending dumps to existing dump sets reduces the amount of unutilized tape and the number of tape changes.)

When a new dump set is created, the tapes that make up the dump set are named using the following parameters of the initial dump: the volume set name, the last element of the dump level, and the dump ID. In addition, the index number identifying the position of the tape in the dump set is also used. Each tape name has the following format:

<*volume set name*>.<*dump level name*>.<*tape index*> (*dump ID*)

If you want to overwrite an existing dump set with a new dump set using the **backup dump** command, two preconditions must be met:

- The name of the initial dump of the new dump set must match the name of the initial dump of the old dump set.
- All of the dumps associated with the old dump set must be expired.

You can still use a tape that does not meet the above conditions; however, the tape must be relabeled (using the **backup labeltape** command) prior to dumping the new dump set. See section 6.3.6 for more information on labeling a tape.

6.6.1. Some Warnings about Making Backups

Dumping a volume makes it inaccessible to other file system users for the duration of the dump process. To reduce inconvenience, create a Backup version of a volume (a version with a `.backup` extension) and dump the Backup version rather than the ReadWrite or ReadOnly version; this does not interrupt users' work. Creating a Backup version of a volume (using **vos backup** or **vos backupsys**) does make its ReadWrite or ReadOnly source volume unavailable for a short period of time (user processes will block but will not fail). Therefore, you may wish to create Backup versions during low-usage periods, possibly using **bos** commands to create a cron process to automate the procedure.

Sometimes the Backup System cannot access a volume (perhaps because of a file server machine or Volume Server outage). When this happens, the Backup System skips the volume until after it has successfully dumped all of the volumes it can. It then attempts to dump each volume it skipped on the first pass, asking the operator monitoring the backup operation whether it should again attempt to dump each volume it was not able to successfully dump.

In the interim, the operator can attempt to rectify whatever problem prevented the volumes from being accessed on the initial pass. When prompted, the operator can instruct the Backup System to again attempt to access a volume, ignore the volume and proceed with the backup operation, or abort the **backup** operation altogether.

If a dump operation is interrupted or fails for any reason, all volumes written to tape *before* the interrupt are complete and usable on the tape. An entry will appear in the Backup Database for the aborted dump and for the volumes that were successfully dumped.

There are three main ways to handle the remaining volumes of a volume set that are not backed up because of an interrupted dump:

- You can pick up the remaining volumes during the next regularly scheduled dump. This approach will create a temporary "hole" in the coverage provided by the Backup System because you will not pickup the missed data until the next regularly scheduled dump.
- You can discard the entire dump set and run the dump again. To discard the dump set, relabel the tape using the **backup labeltape** command; see section 6.3.6 for more information on relabeling a tape.
- You can create a temporary volume set for the missed volumes and dump those volumes in a temporary dump set. The missed volumes can be merged back into the original dump set during the next regularly scheduled dump.

6.6.2. The Backup Process: an Overview

This section provides an overview of the Backup process. In this overview, the volume set *usersys* is dumped at the dump level */sunday1/wednesday*.

1. The Backup System reads the volume set *usersys* in the Backup Database to learn which volume entries it contains. The volume set and its entries were created beforehand with the **backup addvolset** and **backup addvolentry** commands. In this example, the volume entries in *usersys* are

   .* .* user.*

   .* .* sys.*
2. The Backup System scans the VLDB to match the metacharacters from each volume entry and generates a complete list of the volumes to be included in the dump. If multiple volume entries reference the same volume, the duplicates are ignored; only one occurrence of any volume is included.
3. The Backup System reads the dump hierarchy in the Backup Database to check if `/sunday1/wednesday` is an incremental dump and, if so, to determine which preceding level is its parent. In this example, the `sunday1/wednesday` level is incremental and `/sunday1` is its parent.

```
sunday1   expires in  27d
      /monday  expires in  13d
      /tuesday  expires in  13d
      /wednesday  expires in  13d
      /thursday  expires in  13d
      /friday  expires in  13d
sunday2   expires in  27d
      /monday  expires in  13d
      /tuesday  expires in  13d
      /wednesday  expires in  13d
      /thursday  expires in  13d
      /friday  expires in  13d
```

If `/sunday1` were specified, this would be a full dump; the system would copy the complete contents of each volume in *usersys*. Because `/sunday1/wednesday` is an incremental dump level, the dump set includes only those files that have changed since *usersys* was dumped at the `/sunday1` level.

4. Because `/sunday1` is the parent for `/sunday1/wednesday`, the Backup System checks the Backup Database for the date and time of the last dump of *usersys* at the `/sunday1` level.
5. If this dump is the initial dump in a new dump set and the tape contains a previous dump set that is being overwritten, the Backup System checks to make sure that
   - Each dump in the current dump set has expired. If one or more of the expiration dates associated with the dump set has not expired, the operator will be prompted for another tape.
   - The name of the initial dump in the new dump set matches the name of the initial dump in the old dump set. If a mismatch occurs, the operator will be prompted for another tape.
6. The Backup System transfers the list of volumes to be backed up to the appropriate Volume Servers and Tape Coordinator, which determine which data in the volumes was modified after the date and time of the last dump at the `/sunday1` level.
7. The designated Tape Coordinator puts the gathered data onto tape. The expiration date associated with the dump is stored with the data. The Backup System assigns a unique dump ID number to the dump and records the number in the Backup Database. If one tape is not large enough to hold the entire dump, the Backup System prompts the operator to place additional tapes in the drive as necessary.

6.6.3. Using Tapes with a Backup Operation

Before performing a backup, make sure the tapes are at least as large as the tape size listed in the */usr/afs/backup/tapeconfig* file for the tape drive to be used in the operation. The Backup System fills a labeled tape with the amount of data listed as the capacity of the tape in its label. It fills an unlabeled tape with the amount of data listed as the capacity of the drive in the */usr/afs/backup/tapeconfig* file.

If an unlabeled tape is larger than the tape size listed in the *tapeconfig* file, it simply is not filled to capacity when the backup is performed. If an unlabeled tape is smaller than the size listed in the *tapeconfig* file, any volumes which do not completely fit on the tape will be removed from the tape after partial dump and will be dumped on the next tape in the dump set. If any volume in a dump is larger than the actual tape capacity, you will be unable to backup that volume on that tape. Therefore, if you want to ensure maximum use of your tapes, be sure to run the **fms** command to determine the proper size of each type of tape you use.

Note: You should not use the **fms** command on data compressing tape drives because the command will give misleading numbers. Instead, you should configure the tape size to a number larger than the tape itself. The end-of-tape (EOT) handling capacity of the Tape Coordinator will prevent dumps from failing when they hit the EOT.

A volume does not have to fit entirely on a single tape (assuming that the tape capacity matches the size listed in the *tapeconfig* file). If the Backup System reaches the end of a tape while dumping a volume, it puts the remaining data on another tape. The Backup Database automatically records that the volume resides on multiple tapes.

Prior to performing a backup, you can preview the effects of your command without having the system perform the actual dump. Simply include the **-n** flag with the **backup dump** command, specifying the remaining arguments as you would to really execute the dump. This lets you check a volume set's size before actually dumping it so that you can calculate the correct number of tapes needed.

After completing a **backup** operation that involves the creation of a new dump set, you may want to record the dump ID of the initial dump on the paper label of each tape in the dump set. This makes it easier to determine which tapes are required to restore information from the dump set.

⇒ To back up a volume set (create a new dump set)

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup dump** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: Decide which volume set and dump level to use. If necessary, issue the **backup listvolsets** and **backup listdumps** commands to display information about existing volume sets and dump levels. See section 6.5 for more information about the output from these commands.

Step 5: Check that you have a sufficient number of tapes (if you do not have enough tapes, you will not be able to complete the backup).

You must also check that the tapes are properly prelabeled (if necessary, use the **backup readlabel** command to check the labels). You must relabel any tape that is part of a dump set containing one or more unexpired dumps or that contains an initial dump that is different than the initial dump of the new dump set. If the label is incorrect, use the **backup labeltape** command to label the tape correctly. See section 6.3.6 for information about using **backup readlabel** and **backup labeltape**.

Step 6: Issue the **backup dump** command to dump the volume set to tape.

% **backup dump** <*volume set name*> <*dump level name*>
[<*TC port offset*>**] [-n]**

where

dump	must be typed in full.
volume set name	is the name of the volume set to be dumped.
dump level name	is the complete pathname of the dump level at which the volume set is to be dumped.
TC port offset	is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.
-n	specifies that the names of all volumes that would be included in the indicated dump be displayed without the dump actually being performed. Specify all other arguments as you would to actually perform the operation.

Step 7: Place the correct tape in the drive, and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape into the tape drive prior to executing the **backup scantape** command.)

If more than one tape is required, you must remain at the console to respond to prompts for subsequent tapes (if you do not respond immediately, a bell rings periodically to draw your attention). You are prompted for tapes by name; therefore, you should uniquely identify each tape by recording the dump name and dump ID on each tape's paper label.

6.6.4. Placing Multiple Dumps on a Dump Set

By default, only a single volume set at a particular dump level can be dumped onto a single dump set (that is, one dump per dump set). However, by using the **-append** flag with the **backup dump** command, you can append a dump to a dump set that already contains one or more other dumps.

This feature may speed the backup process (because fewer tape changes are required), reduce the number of tapes required to store backup data, and improve tape utilization. When used in combination with the automatic scheduling of dumps feature and the **-noautoquery** feature, this feature can significantly increase the level of backup automation. See section 6.7.1 for more information on the **-noautoquery** feature; see section 6.7.2 for more information on the automatic scheduling of dumps feature.

Two conditions must be met before a dump can be appended to an existing dump set:

- The tape must contain an initial dump created with the AFS 3.3 **backup dump** command.
- The initial dump must be recorded in the Backup Database.

If a tape does not contain an initial dump, the dump operation will continue as an initial dump. If the initial dump does not exist within the Backup Database (according to the tape label), the operator will be prompted to put a valid tape in the tape drive. In the latter case, the operator can either switch to a correct tape or kill the dump job and scan the tape into the Backup Database.

By default, a dump set inherits the most future expiration date associated with the dumps in the dump set. For example, appending a dump with no expiration date to a dump that will expire on 1/1/94 will cause the expiration date of the entire dump set to be 1/1/94.

Note: With the ability to append dumps, the number of tapes in a dump set is theoretically unlimited; however, there are two good reasons to keep your dump sets small (that is, less than five tapes to a dump set for an average cell):

- If a bad spot develops on one of the tapes in a dump set, the subsequent data on that tape and the subsequent tapes in the dump set cannot be used to reconstruct the information in the Backup Database (using the **backup scantape** command). Data can still be recovered from these tapes if the information is properly stored in the database. (See section 6.9.2 for more information on using tapes to reconstruct the information in the Backup Database.)
- If you archive backup data on an as-needed basis (rather than on a regular basis), the entire dump set containing the relevant data must be archived rather than just the tapes containing that data. If your dump sets are large, you will be forced to archive a large number of tapes containing irrelevant data in order to properly archive the relevant data. This wastes both tape and archive storage space. (See section 6.2.7 for more information on archiving.)

⇒ To back up a volume set (append a dump to an existing dump set)

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup dump** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following command assumes regular command mode is used, not interactive mode.

Step 4: Issue the **backup dump** command using the **-append** flag to append a dump to an existing dump set:

% **backup dump** <*volume set name*> <*dump level name*>
[-portoffset <*TC port offset*> **-append**

where

volume set name is the name of the volume set to be dumped.

dump level name is the complete pathname of the dump level at which the volume set is to be dumped.

TC port offset is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

6.7. Procedures for Automating the Backup System

This section provides information on procedures that increase the level of automation of the Backup System. The procedures are completely optional; you may incorporate them into your backup procedures if desired. There are two main ways to improve the level of automation in your backup system:

- Disable unnecessary operator queries from the Tape Coordinator window. This is accomplished by using the **-noautoquery** flag with the **butc** command.
- Set up **backup dump** commands to be invoked automatically at specified times. This is accomplished by specifying an execution time with the **-at** argument of the **backup dump** command or creating a file containing **backup dump** command line instructions and specifying that file with the **-file** argument.

Note: Appending multiple dumps to a single dump set may also increase the level of system automation because it can reduce the number of required tape changes. See section 6.6.4 for more information on placing multiple dumps on a single dump set.

The sections below discuss these automation options.

6.7.1. Disabling Initial Prompts for Tape Insertion

When you execute a **backup** command involving a tape, the appropriate Tape Coordinator prompts you, by default, to insert the correct tape in the appropriate drive and press the **Return** key. If the Tape Coordinator could assume that the drive already contained the correct tape, it could immediately perform the requested action without having to prompt the operator. By default, the Tape Coordinator will not make this assumption; however, if the **butc** program is started with the **-noautoquery** flag, the Tape Coordinator will make this assumption and immediately begin the requested operation.

This feature is useful if you want to run unattended backups; however, if multiple tapes are required to complete an operation, an operator must be present to change tapes. Furthermore, if the Tape Coordinator encounters a error condition, it will write an error message to the Tape Coordinator window and operator intervention will be necessary to continue the operation. Some of the more common error conditions include:

- No tape in the drive.
- Unexpired tape in the drive.
- Write-protected tape in the drive.

This feature affects all **backup** commands that interact with a Tape Coordinator/tape drive, including:

- **backup diskrestore**
- **backup dump**

- **backup labeltape**
- **backup readlabel**
- **backup restoredb**
- **backup savedb**
- **backup scantape**
- **backup volrestore**

To take advantage of this feature, you must insert the appropriate tape in the drive prior to executing any of the above commands. If you fail to insert the appropriate tape, an error message will appear on the Tape Coordinator window and you will need to insert an appropriate tape and press **Return**, effectively nullifying the purpose of the feature.

When used in combination with the automatic scheduling of dumps feature and the multiple dumps to a dump set feature, this feature can increase the level of backup automation. See section 6.6.4 for more information on the multiple dumps to a dump set feature; see section 6.7.2 for more information on the automatic scheduling of dumps feature.

6.7.2. Automatic Scheduling of Dumps

It is possible to direct one or more **backup dump** commands to execute at specified times in the future. These types of dumps are referred to as ***timed*** or ***scheduled dumps***; there are two ways to set up timed dumps:

- You can instruct a single **backup dump** command to execute at a specified time by executing the **backup dump** command with the **-at** argument.
- You can instruct one or more **backup dump** commands to execute at one or more specified times by executing the **backup dump** command with the **-file** argument and specifying the name of a file containing **backup dump** command instructions.

When a timed dump is executed, it is first checked for validity. The following validity checks are made:

- The volume set must exist within the Backup Database.
- The dump level must exist within the Backup Database.
- The port offset must be valid.
- The specified date and time for command execution must be in the future.
- Valid tokens must be available for use by a timed dump. If no valid tokens are available at the time the command is issued or executed, the operation will fail.

If the command is valid, it is assigned a job number and added to the jobs list. These jobs may be listed with the **(backup) jobs** command and killed with the **(backup) kill** command.

Note: Because this feature requires unexpired tokens and backup tapes to be left unattended, you may want to institute additional physical security measures to protect against improper access.

When used in combination with the **-noautoquery** feature and the multiple dumps to a dump set feature, this feature can increase the level of backup automation. See section 6.6.4 for more information on the no multiple dumps to a dump set feature; see section 6.7.1 for more information on the **-noautoquery** feature.

⇒ To specify a single timed dump (backup dump with -at argument)

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to the execution of the **backup dump** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. If executed in interactive mode, the **backup dump** command will be submitted as a background job and the `backup>` prompt will be returned to the operator. If executed in non-interactive mode, the **backup dump** command will remain in the foreground until the specified execution time; only then will the command line prompt be returned to the operator. The other advantages of interactive mode are described in section 6.4.2. The following command is shown in non-interactive mode.

Step 4: Issue the **backup dump** command using the **-at** argument to create a single timed dump:

% **backup dump** <*volume set name*> <*dump level name*>
[-portoffset <*TC port offset*>**] -at** <*Date/time to start dump*>

where

volume set name is the name of the volume set to be dumped.

dump level name is the complete pathname of the dump level at which the volume set is to be dumped.

TC port offset is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

Date/time to start dump
specifies the date and time at which the **backup dump** command should execute. The date is required if this argument is used. The time is optional; if it is excluded, the time defaults to *12:00 am*. The instance must be specified using the following format: *mm/dd/yy hh:MM*, where *mm/dd/yy* indicates the month (mm), day (dd), and year (yy) and *hh:MM* indicates the hour (hh) and minutes (MM) in numerical form (use a 24-hour format).

For example, *12/23/93 20:20* indicates that the process should execute at 8:20 pm on December 23, 1993.

Note: Although the usage message for this command indicates that multiple instances can be included for this argument, it is not possible to specify more than one start date/time. The plus sign follows the *expiration date* argument because a multi-word argument is required to define a start date/time, not because multiple dates/times can be specified.

Note: If you intend to run unattended backups via this command you must:

- Ensure that the Tape Coordinator (**butc**) is running with the **-noautoquery** flag. See section 6.7.1 for information on this flag.
- Place the appropriate tape in the specified drive prior to the scheduled execution of the program.

You cannot run unattended backups if a dump cannot be completely stored on one tape because you must remain at the console to respond to prompts for subsequent tapes.

⇒ To specify one or more timed dumps (backup dump with -file argument)

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to the execution of the **backup dump** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. If executed in interactive mode, the **backup dump** command will be submitted as a background job and the `backup>` prompt will be returned to the operator. If executed in non-interactive mode, the **backup dump** command will remain in the foreground until the last specified execution time; only then will the `backup>` prompt be returned to the operator. The other advantages of interactive mode are described in section 6.4.2. The following command is shown in non-interactive mode.

Step 4: Issue the **backup dump** command using the **-file** argument to create one or more timed dumps:

% **backup dump -file** <*load file*>

where

-file specifies the pathname of the file containing **backup dump** commands that will be executed at the specified time(s). You can specify the full pathname of the file or a partial pathname if the path includes the current directory.

The format of the individual lines in the *load file* are the same as the syntax of the **backup dump** command except the word **backup** should not be included. This format is shown below. (**Note:** Each command must be placed on a single line; the syntax example below is shown on two lines because of space restrictions.)

dump <*volume set name*> <*dump level name*> [**-port** <*TC port offset*>]
-at <*Date/time to start dump*> [**-append**]

The *volume set name* and *dump level name* are mandatory; if desired, you may include the **-vol** and **-dump** switches in the file, but they are not required. You must provide a value for the *TC port offset* unless the default value of *0* is appropriate.

The *Date/time to start dump* is also mandatory; see the instructions on page 6-52 for information on date/time format. You can include the **-at** switch if desired; however, if you did not specify a value for *TC port offset*, you must include the switch.

Include the **-append** flag if you want a specific dump to be appended onto an existing dump set.

Below is an example of a **backup dump** file:

```
dump user /sunday1/wednesday 1 02/05/94
dump rt /sunday1/wednesday 2 02/05/94
dump sun /sunday1/wednesday 2 02/05/94 02:00 -append
```

Note: If you intend to run unattended backups via this command you must:

- Ensure that each appropriate Tape Coordinator (**butc**) is running with the **-noautoquery** flag. See section 6.7.1 for information on this flag.
- Place the appropriate tape in the specified drive prior to the scheduled execution of the program.

You will not be able to run unattended backups if a particular backup requires more than one tape because you must remain at the console to respond to prompts for subsequent tapes.

6.8. Restoring and Recovering Data

When you restore data to the file system, you can restore individual volumes with the **backup volrestore** command or you can restore an entire disk partition with the **backup diskrestore** command. Using **backup volrestore**, you can perform a full restore of the most recently dumped version of a volume or you can perform a date-specific restore, which restores the volume to the state it was in at its last dump before the indicated date. With either command, the creation date of each restored volume is set to the date and time at which it is restored.

Before performing a restore, you can direct the Backup System to list the tapes needed to complete the operation. This allows you to locate the proper tapes before actually issuing the command. To view the list, include the **-n** flag with **backup volrestore** or **backup diskrestore**, along with any other arguments you intend to use with the command.

Note: If a restore operation is interrupted or fails for any reason, you cannot be sure that any of the restored volumes are complete in the file system. Immediately restart the restore. If you do not, the file system may contain inconsistent information, which can create problems.

6.8.1. Specifying the Type and Destination of a Restore Operation

The Backup System performs a ***current restore*** by default, recreating a volume as it existed when last dumped (including data from the last full dump and any subsequent incremental dumps). With the **backup volrestore** command, you can use the **-date** argument to direct the system to perform a ***date-specific restore***. This recreates the volume as it was when last dumped before the indicated date. It includes the last full dump and any incremental dumps done before the indicated date.

Using **backup volrestore**, you can restore data with the same name it had previously, in which case it replaces the existing information in the file system, or you can restore it with a new name, in which case the original data is maintained in the file system. Either way, you can restore it to the same or a different location.

- To overwrite the current contents of a volume, use the **-server**, **-partition**, and **-volume** arguments. If **-server** and **-partition** specify the location at which the volume currently resides, the current volume is overwritten with the restored volume; the volume's VLDB entry remains unchanged. If the arguments specify a new location, the current volume is deleted from the file system, and its VLDB entry is updated accordingly.
- To preserve the current contents of a volume in the file system, use the **-extension** argument to restore the data to a new volume with the same name as the existing volume with the addition of a distinguishing extension, such as `.restored`. A new VLDB entry is created for it, and it is assigned its own volume ID number. You can place the restored volume at the same partition as the existing volume or at a different partition. You must mount the newly restored volume before it can be used.
- To restore a volume that no longer exists in the file system, simply provide the **-server**, **-partition**, and **-volume** arguments. A new VLDB entry is created for the volume and a volume ID is assigned. You must mount the newly restored volume before it can be used.

Using **backup diskrestore**, the following specifications are possible:

- To restore the contents of the partition to the same location, use the **-server** and **-partition** arguments. All volumes listed in the VLDB as residing at the specified site are restored to the site. The VLDB entries of the volumes remain unchanged.
- To move the contents of the partition to a new location, include the **-newserver** and **-newpartition** arguments. All volumes listed in the VLDB as residing at the specified site are restored to the new location. The VLDB entries of the volumes are updated accordingly.

⇒ To restore individual volumes

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If they are not already running, start the Tape Coordinator(s) for the tape drive(s) you want to use with the operation. If a **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to the execution of the **backup volrestore** command. Refer to section 6.4.3 for information on using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: 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 an acceptable abbreviation of **volrestore**.

destination machine specifies the name of the file server machine to which each specified volume is to be restored. This can be the volumes' current file server machine or a different machine.

destination partition specifies the name of the partition to which each specified volume is to be restored. This can be the volume's current partition or a different partition.

volume(s) to restore specifies the name of each volume to be restored. If you specify more than one volume, you must include the **-volume** switch.

If you specify the name of the ReadWrite version of a volume, the Backup System automatically appends a `.backup` extension to the name. It then determines which version of the volume was dumped most recently, the Backup or the ReadWrite, and restores that version. If you include a `.backup` or `.readonly` extension with the name of the volume, the Backup System restores only that version of the volume.

new volume name extension

specifies an extension to be added to the name of each restored volume. This preserves the existing volume with that name in the file system. If you specified more than one volume to be restored and you want to include this argument, you must use the **-extension** switch. If you want a period to separate the extension from the name, specify the period as the first character of *new volume name extension* (for example, `.restored`).

date from which to restore

specifies the date and time to use for a date-specific restore. Only information from dump sets of the indicated volumes dated before the specified date is restored. Omit this argument to perform a full restore.

You can specify a date or a date and time combination: *mm/dd/yy* specifies 00:00 (12:00 a.m.) on month/day/year (for example, *02/08/92*); *mm/dd/yy hh:mm* specifies a time (hours:minutes) on month/day/year (for example, *"02/08/92 06:00"*). As in the example, the time must be in military time (24-hour format), and the entire instance must be surrounded with double quotes because it contains a space.

TC port offsets specifies the port offset number(s) of the tape drive(s) to be used in the operation. If different tape drives were used to create the tapes that contain the dump from which data is to be restored, specify the port offset numbers of all of the drives required for the operation, in the order in which they were used to create the dumps (that is, full dump first, next lower incremental, etc.,).

Port offsets are paired with dump levels in a one to one correspondence. Note that a dump needs only to be restored from the same type of drive, not the exact same physical drive.

You must provide this argument unless the default value of *0* is appropriate for all required dump. However, the port offset of *0* must be provided if it is preceded or followed by a different port offset required for another dump.

Note: If you specify more than one volume to restore, the portoffset order must be the same for both volumes.

-n specifies that the names of all tapes needed to perform the operation are to be listed, without the operation actually being performed. Specify all other arguments as you would to actually perform the operation.

Step 5: Place the correct tape in the drive, and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape(s) into the tape drive(s) prior to executing the **backup volrestore** command.) If more than one tape is required for any tape drive, you must remain at the console to respond to prompts for subsequent tapes.

⇒ To restore a partition

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup diskrestore** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: Issue the **backup diskrestore** command with the desired arguments.

% **backup diskrestore** <*machine to restore*> <*partition to restore*>
[<*TC port offset*>**] [-newserver** <*destination machine*>**]**
[-newpartition <*destination partition*>**] [-n]**

where

di is an acceptable abbreviation of **diskrestore**.

machine to restore specifies the name of the file server machine that houses the partition whose contents are to be restored.

partition to restore specifies the name of the partition on *machine to restore* whose contents are to be restored.

TC port offset is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate. Note that a partition can be restored from a single drive only.

destination machine specifies the name of an alternate file server machine to which the data is to be restored. Omit this option to restore the data to the file server machine on which it formerly resided (*machine to restore*).

destination partition
specifies the name of an alternate partition to which the data is to be restored. Omit this option to restore the data to the partition with the same name as the one on which it formerly resided (*partition to restore*).

-n specifies that the names of all tapes needed to perform the operation are to be listed, without the operation actually being performed. Specify all other arguments as you would to actually perform the operation.

Step 5: Place the correct tape in the drive, and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape into the tape drive prior to executing the **backup diskrestore** command.) If more than one tape is required, you must remain at the console to respond to prompts for subsequent tapes.

6.9. The Backup Database

The Backup Database stores information used to create and track all backups done in a cell. While it is relatively easy to recreate volume set, Tape Coordinator host, and dump hierarchy information in the Backup Database, information about dumps is difficult to recreate. Therefore, it is important that you periodically copy the Backup Database to tape with the **backup savedb** command.

Should the Backup Database become corrupted, you can restore it to its former condition using two commands:

- The **backup restoredb** command is used to restore the latest copy of the Backup Database created with the **backup savedb** command.
- The **backup scantape** command (with the **-dbadd** flag) is used to reconstruct Backup Database data from dump set tapes created after the latest backup of the Backup Database.

You will need to manage the size of the Backup Database. If it gets too large, its performance degrades. The **backup savedb** and the **backup deletedump** commands contain features that will help you manage the size of the Backup Database.

6.9.1. About Backing Up the Backup Database

You should set up a backup schedule for the Backup Database. You should base the schedule on the amount of data that you are willing to manually scan in from tape (using the **backup scantape** command) should a database failure occur.

When you issue the **backup savedb** command, the entire Backup Database is copied to tape. When the command is issued, the tape is labeled with the name *Ubik_db_dump.1* (*dump ID*). If the database does not fit on a single tape, subsequent tapes are labeled *Ubik_db_dump.2* (*dump ID*) and so on.

When the Backup Database is saved to tape, a dump record is created in the Backup Database. The expiration date of this dump is automatically set to NEVER. The only way to remove a Backup Database dump record is to use the **backup deletedump** command with the **-dumpid** argument; see section 6.9.3 for information on this command.

⇒ To back up the Backup Database

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup savedb** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: Issue the **backup savedb** command to save the Backup Database to tape.

% **backup savedb [**<*TC port offset*>**]**

where

sa is an acceptable abbreviation of **savedb**.

TC port offset is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

Step 5: Place an unlabeled tape, a tape labeled with NULL, or a tape labeled with the name *Ubik_db_dump.1* in the drive, and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape into the tape drive prior to executing the **backup savedb** command.)

6.9.2. About Restoring the Backup Database

If the Backup Database becomes corrupted (for instance, if the disk housing the database becomes damaged), you must delete the old database and restore an entirely new version from its backup tape. You can use the **backup dbverify** command to determine whether the Backup Database is damaged. Section 6.5 provides information about using **backup dbverify**.

Do not attempt to recover information from a damaged Backup Database. Instead, follow the steps described on the following pages to remove the damaged database and restore a backup version from tape with the **backup restoredb** command. Once the database is restored, volume set, volume entry, dump level, and Tape Coordinator information modified since the last backup of the database must be recreated manually; see section 6.3 for instructions.

Database records for dumps and dump sets created since the last backup of the database can be recreated from the tapes housing the new and modified dump sets by using the **backup scantape** command with the **-dbadd** flag. This command can also be used to restore information about dump sets from tape should such information ever be accidentally removed from the database. (The **backup scantape** command cannot be used to scan Backup Database dump tapes.)

When it transfers information from a tape to the Backup Database, **backup scantape** creates an entry in the Backup Database for each dump on the tape. Database entries are identified by tape name, dump level pathname, and dump ID number, which is unique for every dump. Because the database can contain only one entry per dump ID number, the command will not create an entry for a dump with the same dump ID as an existing entry in the database nor will it overwrite an existing entry.

⇒ To restore the Backup Database from a Database backup

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: Use the **bos shutdown** command to stop all Backup Server (*buserver*) processes on all database server machines. Repeat the command for each database server machine.

% **bos shutdown** <*machine name*> **buserver [-wait]**

where

sh	is an acceptable abbreviation of **shutdown**.
machine name	specifies the name of the database server machine on which the Backup Server is to be stopped.
buserver	specifies the standard name assigned to the Backup Server process. Use the name assigned to the Backup Server in the */usr/afs/local/BosConfig* file at the time of its creation.
-wait	indicates that the command shell prompt is not to return until the Backup Server process has stopped. If you omit this flag, the prompt returns almost immediately, even if the process is not yet stopped.

Step 3: Choose one of the database server machines (it does not matter which), and become "root" in the machine's UNIX file system by issuing the **su** command.

Step 4: Remove the old Backup Database by deleting the */usr/afs/db/bdb.DB0* and */usr/afs/db/bdb.DBSYS1* files from the database server machine.

\# **rm /usr/afs/db/bdb.DB0**
\# **rm /usr/afs/db/bdb.DBSYS1**

Step 5: Log out as "root" from the database server machine's UNIX operating system by issuing the **exit** command or typing **^d** (Control-d).

Step 6: Repeat the previous three steps on the remaining database server machines. You must delete the old Backup Database from all database server machines.

Step 7: Use the **bos startup** command to restart all Backup Server (*buserver*) processes on all database server machines. When you restart the Backup Server on a machine, the process creates an empty Backup Database by creating the */usr/afs/db/bdb.DB0* and */usr/afs/db/bdb.DBSYS1* files on the machine. Repeat the command for each database server machine.

% **bos startup** <*machine name*> **buserver**

where

startu	is an acceptable abbreviation of **startup**.
machine name	specifies the name of the database server machine on which the Backup Server is to be started.
buserver	specifies the standard name assigned to the Backup Server process. Use the name assigned to the Backup Server in the */usr/afs/local/BosConfig* file at the time of its creation.

Step 8: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 9: Issue the **backup addhost** command to create an entry in the empty Backup Database for the Tape Coordinator from which you will restore the database.

If you have multiple database server machines, wait approximately five minutes after starting the last Backup Server before you issue this command. This allows Ubik to elect a synchronization site from among the database server machines.

% **backup addhost** **<***tape machine name***>** **[<***TC port offset***>]**

where

addh	is an acceptable abbreviation of **addhost**.
tape machine name	is the complete Internet host name of the Tape Coordinator machine on which the Tape Coordinator is to run.
TC port offset	is the port offset number of the drive to be used in the operation. You must use the port offset number specified for the Tape Coordinator in the */usr/afs/backup/tapeconfig* file. You must provide this argument unless the default value of *0* is appropriate.

Step 10: Start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup savedb** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 11: Issue the **backup restoredb** command to restore the Backup Database from tape.

% **backup restoredb** **[<***TC port offset***>]**

where

res	is an acceptable abbreviation of **restoredb**.
TC port offset	is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

Step 12: Place the tape labeled with the name *Ubik_db_dump.1* in the drive, and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape into the tape drive prior to executing the **backup restoredb** command.)

⇒ To recover Backup Database data from dump set tapes

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup scantape** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: Issue the **backup scantape** command. Information from the tape is displayed in the Tape Coordinator window.

% **backup scantape [-dbadd] [**<*TC port offset*>**]**

where

sc is an acceptable abbreviation of **scantape**.

-dbadd specifies that information from the tape is to be added to the Backup Database. See section 6.5 for more information about the output from the command.

TC port offset is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

Step 5: Place the tape to be scanned in the drive, and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape into the tape drive prior to executing the **backup scantape** command.) If multiple tapes are to be scanned, they must be inserted into the drive in sequential order.

6.9.3. About Managing the Backup Database

The Backup System automatically removes the record of a dump set, and its constituent dumps, from the Backup Database when any tape in a dump set is overwritten or relabeled. However, as time goes on, irrelevant or outdated records may accumulate in the Backup Database.

There are two commands that can help you to maintain your Backup Database:

- The **backup savedb** command contains the **-archive** argument that allows you to specify a deletion date for the dump records in the Backup Database. After the Backup Database is dumped, the Backup System will delete all dump records created prior to the specified **-archive** date; dump records associated with backups of the Backup Database will <u>not</u> be deleted during this archive. System administrators should incorporate this functionality by creating an archiving strategy.
- The **backup deletedump** command can be used to manually remove information about a dump set (and all dumps appended to it) from the Backup Database. This command may be used to remove dump records that
  - Contain incorrect information (possibly because a dump operation was interrupted or failed).
  - Contain information associated with a tape that is being discarded.
  - Correspond to a Backup Database dump. Using the **backup deletedump** command (with the **-dumpid** argument) is the only way to remove a record of a Backup Database dump.

  The **backup deletedump** command cannot be used to delete an appended dump record independently of the initial dump record in a dump set. An appended dump record can only be deleted indirectly as a result of the deletion of the initial dump record in a dump set. When an initial dump record is deleted using the **backup deletedump**, the output of the command lists the dump IDs of all dump records deleted as a result.

After you use **backup deletedump** to delete the record of a dump set from the Backup Database, any dump sets for which it serves as the parent, either directly or indirectly, are unusable. You can reissue the command to delete those dump sets from the database; however, leaving them in the database, while possibly confusing, causes no problems. Also, as long as the tape that contains the parent dump set remains available, you can always use the **backup scantape** command to restore dump set information about the parent from that tape to the database, making the dump sets that rely on the parent dump set usable again (see section 6.9.2).

⇒ To archive the Backup Database

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: If it is not already running, start the Tape Coordinator for the tape drive you want to use with the operation. If the **butc** process is run with the **-noautoquery** flag, you should insert the appropriate tape prior to executing the **backup savedb** command. Refer to section 6.4.3 for information about using the **butc** command to start a Tape Coordinator.

Step 3: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 4: Issue the **backup savedb** command to save the Backup Database to tape.

% **backup savedb [**<*TC port offset*>**] -archive** <*date time*>

where

sa is an acceptable abbreviation of **savedb**.

TC port offset is the port offset number of the drive to be used in the operation. You must provide this argument unless the default value of *0* is appropriate.

-archive specifies a deletion date and time cut off. After the Backup Database is dumped, all dump records created before the specified date and time are deleted from the Backup Database (except for records of Backup Database dumps). If a dump set contains one or more dumps whose creation date falls after the specified date, none of the dumps in the dump set are deleted.

The instance must have the following format: *mm/dd/yy hh:MM*, where *mm/dd/yy* indicates the month (mm), day (dd), and year (yy) and *hh:MM* indicates the hour (hh) and minutes (MM) in 24-hour format. The date is required. The time is optional; if it is excluded, the time defaults to *12:00:59 am*.

You can also use **NOW** for this instance to specify the current date and time. Using **NOW** will delete all database records in the Backup Database except for records of Backup Database dumps.

Note: Although the usage message for this command indicates that multiple instances can be included for this argument, it is not possible to specify more than one archive date and time. The plus sign follows the argument because a multi-word argument is required to define an archive date, not because multiple dates can be specified.

Step 5: Place an unlabeled tape, a tape labeled with NULL, or a tape labeled with the name *Ubik_db_dump.1* in the drive, and press the **Return** key in the Tape Coordinator window. (If you are running the **butc** process with the **-noautoquery** flag, you will not be prompted in this manner unless you failed to insert the appropriate tape into the tape drive prior to executing the **backup savedb** command.)

⇒ To delete dump records from the Backup Database

Step 1: Verify that you are included in the */usr/afs/etc/UserList* file. If necessary, issue **bos listusers** to check.

Step 2: *Optional*. Issue the **backup** command to enter interactive mode. The advantages of interactive mode are described in section 6.4.2. The following commands assume regular command mode is used.

Step 3: If necessary, issue the **backup dumpinfo** command to list information, including dump IDs, about dump sets recorded in the Backup Database. See section 6.5 for more information on this command.

Step 4: Issue the **backup deletedump** command to delete the dump set.

% **backup deletedump [-dumpid** <*dump id*>$^+$**] [-from** <*date time*>**]**
[-to <*date time*>**]**

where

dele is an acceptable abbreviation of **deletedump**.

-dumpid provides the dump ID of each dump to be deleted from the Backup Database. Each dump ID must be associated with an initial dump; appended dumps cannot be directly deleted. Using this argument is the only way to delete Backup Database dump records.

-from specifies the bottom end of the creation date range for the group of dump IDs that you want to delete from the Backup Database. If you use the **-to** argument without this argument, the from date and time will default to *0* (the UNIX equivalent of January 1, 1970).

The instance must have the following format: *mm/dd/yy hh:MM*, where *mm/dd/yy* indicates the month (mm), day (dd), and year (yy) and *hh:MM* indicates the hour (hh) and minutes (MM) in 24-hour format. The date is required. The time is optional; if it is excluded, the time defaults to *12:00:59 am*.

-to specifies the top end of the creation date range for the group of dump IDs that you want to delete from the Backup Database. If you use the **-to** argument without the **-from** argument, the from date and time will default to *0* (the UNIX equivalent of January 1, 1970).

The instance must have the following format: *mm/dd/yy hh:MM*, where *mm/dd/yy* indicates the month (mm), day (dd), and year (yy) and *hh:MM* indicates the hour (hh) and minutes (MM) in 24-hour format. The date is required. The time is optional; if it is excluded, the time defaults to *12:00:59 am*.

Note: You can also use **NOW** for this instance to specify the current date and time. Using **NOW** with the **-to** argument will delete all database records in the Backup Database from the date specified with the **-from** argument up to the current date and time.

7. Monitoring and Auditing Tools

AFS comes with two main monitoring tools:

- ***Scout*** – monitors the status of file server processes and gathers data on those processes.
- ***AFSMonitor*** – monitors the status of both file server and cache manager processes and gathers data on those processes.

AFS also contains a mechanism to allow the auditing of AFS events on file server machines running AIX.

The chapter includes the following sections:

7.1 Summary of Commands 7-1
7.2 Scout 7-2
7.3 AFSMonitor 7-14
7.4 The xstat Data Collection Facility 7-25
7.5 Auditing AFS Events on AIX File Servers 7-29

7.1. Summary of Commands

Initialize Scout . . . . **scout**
Initialize AFSMonitor . . . . **afsmonitor**

7.2. Scout

7.2.1. About Scout

Scout (process name: **scout**) makes it easy to monitor 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 those statistics in an understandable screen format, and alerts you if any of the statistics exceed a configurable threshold.

Scout allows you to

- Monitor, from a single location, any number of File Server processes (the only limit is the window size which must be large enough to display the specified processes). The monitored File Server processes can reside on machines in multiple cells.
- Set thresholds for many of the statistics. When the value of a statistic exceeds the threshold, **scout** 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 section 7.2.1.4.
- Respond more quickly to a File Server outage by highlighting the name of any file server which does not respond to a **scout** probe. A possible cause for response failure is that the File Server machine or process or a part of the network is down. When the File Server machine or process or the network comes back up, the machine name and statistics appear normally again.
- Specify how often **scout** collects statistics from the File Server processes.

7.2.1.1. Requirements for Running Scout

You can run the **scout** program on any machine configured as an AFS client. In the standard client configuration, the binary file for **scout** resides in */usr/afsws/bin*. The machine also must be able to access the **curses** graphics package, since **scout**'s display abilities rely on it. Most UNIX distributions include **curses** as a standard utility.

Both "dumb" terminals and windowing systems that emulate terminals can run **scout**. The **scout** screen will look the best on machines that support reverse video and cursor addressing. To let **scout** know what type of environment is in use, 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 setting for `TERM` is `vt100`, as long as 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 **scout**, so its presence in */usr/afsws/bin* on a machine means that any user of that machine may monitor File Server status, as long as */usr/afsws/bin* appears in the user's path. The probes through which **scout** collects statistics do not constitute a significant burden on the File Server 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.

Any number of instances of **scout** may run on a single machine. Each instance runs in a separate dedicated window. The **scout** process must run in the foreground, so once you initialize it in a window, that window does not accept further input (except a kill signal).

It is also possible to run **scout** on several machines and view its output on a single machine. To achieve this, open multiple windows on the single machine, then telnet to a different machine in each window in order to run **scout** there. In this case, you might find it useful to mark each window with the name of the "host" machine running **scout**, by including the **-host** flag on the **scout** command line. See section 7.2.1.3 for more details.

7.2.1.2. Using a "Basename" When Monitoring Multiple File Server Processes

As mentioned in the introduction, a single instance of **scout** can monitor the File Server process running on any number of file server machines. Most of the time, you will monitor only your cell's file server machines, which usually implies that all of the machine names in the window will share the same ***basename*** (end in the same string). If your cell's machines have Internet-style host names, then their basename is the same as your cell name. For example, all machine names in the Transarc Corporation cell end in the basename *transarc.com*, which is also the cell name.

If all machines in a **scout** window have the same basename, you can use the **-basename** argument on the **scout** command to specify it. This has several advantages:

- You have to type the basename only once, rather than on every machine name. When typing machine names on the command line, specify only the initial, distinctive part.
- As **scout** establishes a connection with the file server machines you wish to monitor, it automatically appends the basename to the abbreviated names you provide. It is "legal" to use abbreviated names even without **-basename**, but in that case correct resolution of the names (necessary for establishing the connection) depends on the state of your cell's naming service at the time of connection.
- The machine names are more likely to fit in their column in the **scout** window without having to be truncated (see section 7.2.1.3 for more on truncating names in the display column).
- The basename appears at the top of the **scout** window, displaying the name of the cell you are monitoring.

7.2.1.3. The Layout and Contents of the Scout Screen

The **scout** screen can display statistics either in a dedicated window or on a plain screen if a windowing environment is not available. For best results, the window or screen should have the ability to print in reverse video and do cursor addressing.

The **scout** screen has three main regions: the banner line, the statistics display region and the probe/message line. This section describes the contents of the three regions. Section 7.2.3 provides illustrated examples.

The Banner Line

By default, the word "Scout" appears in the banner line at the top of the window or screen, to indicate that **scout** is running. Two additional types of information may appear in the banner line if you include the appropriate flag or argument on the command line:

- The **-host** flag causes **scout** to display this machine name in the banner line. As mentioned previously, this is useful when you are running **scout** on several machines but displaying the results on a single machine.

  For example, if you run **scout** on the machine *client1.transarc.com* and use the **-host** flag, the banner line reads:

  ```
  [client1.transarc.com] Scout
  ```

- The **-basename** argument causes **scout** to display the indicated basename on the banner line. Section 7.2.1.2 explains what the basename is and why using it is advantageous.

  For example, if you specify a value of *transarc.com* for **-basename**, the banner line reads:

  ```
  Scout for transarc.com
  ```

The Statistics Display Region

In this region, which occupies the majority of the window, **scout** displays the statistics gathered for each File Server process. Each process appears on its own line.

The region is divided into six columns, labeled as indicated and displaying the following information:

- `Conn`: The first column displays the number of RPC (remote procedure call) connections open between the File Server process and client machines. This number should equal or exceed the number in the `Ws` column (see the fourth entry below). It can exceed the `Ws` number because each user on the machine can have several separate connections open at once, and one client machine can handle several users.
- `Fetch`: The second column displays the number of fetch-type RPCs (fetch data, fetch access list, and fetch status) that client machines have made to the File Server process since the latter started. This number is reset to zero each time the File Server process restarts.
- `Store`: The third column displays the number of store-type RPCs (store data, store access list, and store status) that client machines have made to the File Server process since the latter started. This number is reset to zero each time the File Server process restarts.
- `Ws`: The fourth column displays the number of client machines (workstations) that have communicated with the File Server process within the last 15 minutes (such machines are defined as "active"). This number is likely to be smaller than the number in the first (`Conn`) column because a single client machine can have several connections open to one File Server.
- The fifth, unlabeled, column displays the name of the file server machine on which the File Server process is running. Names of 12 characters or less are displayed in full; longer names are truncated and an asterisk (`*`) appears as the last character in the name. Using the **-basename** argument is a good way to avoid truncation, but only if all machine names end in a common string; see section 7.2.1.2.

- `Disk attn`: The sixth column displays the number of available kilobyte blocks on each AFS disk partition on the file server machine. The display for each partition has the form

  `x:`*free_blocks*

  where `x` indicates the partition name. For example, `a:8949` specifies that partition */vicepa* has 8,949 1Kbyte blocks free. Available space may be displayed for up to 26 partitions. If the window is not wide enough for all partition entries to appear on a single line, **scout** automatically creates multiple lines, "stacking" 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 label is

  ```
  Disk attn: > 95% used
  ```

  because by default **scout** highlights the entry for any partition that is over 95% full. See section 7.2.1.4 for more on this threshold and its effect on highlighting.

For all columns except the fifth (file server machine name), you can use the **-attention** argument to set the value at which entries in the column should be highlighted to notify you that a certain value has been exceeded. Only the fifth and `Disk attn` columns ever become highlighted by default. See section 7.2.1.4 for details on how to use the **-attention** argument.

The Message/Probe Line

The bottom line of the **scout** screen indicates how many times **scout** has "probed" the File Server processes for statistics. The statistics gathered in the latest probe appear in the statistics display region. By default, **scout** probes the File Servers every 60 seconds, but you can use the **-frequency** argument to specify a different probe frequency.

7.2.1.4. Highlighting of Fields

The **scout** screen uses highlighting (reverse video) to draw your attention to the entries in a column that exceed a threshold value. Because you set the threshold value for most statistics, you determine which values are worthy of special attention and which are normal.

Highlighting for Extreme Values

You can use the **-attention** argument on the **scout** command line to set the threshold value for five of the six columns in the display region. (Highlighting in the display’s fifth column is not under your control.) The threshold you set for a column applies to all of the entries in it. It is not possible to set thresholds on a per-machine basis.

The **-attention** argument accepts one or more instances, each of which pairs the statistic type and a threshold value. The statistic/threshold pairs may appear in any order. The legal values for the pairs are as follows.

- **conn** <*connections*>. Sets the threshold in the `Conn` column, determining the maximum number of connections that the File Server may have open to client machines before the value is highlighted. Highlighting deactivates when the value goes back below the threshold. By default, no threshold is set.
- **fetch** <*number of fetch RPCs*>. Sets the threshold in the `Fetch` column. Indicates the maximum number of fetch RPCs that can be made to the File Server before the value is highlighted. Highlighting deactivates when the value goes below the threshold; this is possible only if the File Server is restarted, at which time the value returns to zero. By default, no threshold is set.
- **store** <*number of store RPCs*>. Sets the threshold in the `Store` column, determining the maximum number of store RPCs that can be made to the File Server before the value is highlighted. Highlighting deactivates when the value goes below the threshold; this is possible only if the File Server is restarted, at which time the value returns to zero. By default, no threshold is set.
- **ws** <*number of active client machines*>. Sets the threshold in the `Ws` column, determining the maximum number of "active" client machines the File Server can serve before the value is highlighted. "Active" machines are defined as those that communicated with the File Server in the last 15 minutes. Highlighting deactivates when the value goes below the threshold. By default, no threshold is set.
- **disk** <*percent full*>**%**. Sets the threshold in the `Disk attn` column, determining the maximum percentage of the disk that can be full before the value is highlighted; legal values are integers from *0%* to *99%* (you must type the percent sign).

  <u>or</u>

  disk <*minimum blocks free*>. Sets threshold in the `Disk attn` column, determining the minimum number of kilobyte blocks that must be available on the partition if the value is not to be highlighted.

  As the definitions indicate, you must include the percent sign when you want the threshold to be a percent—**scout** interprets the number as a number of disk blocks if a percent sign does not follow it. If there is a percent sign, then **scout** highlights the partition's entry when a higher percentage of its blocks are full than the threshold *percent full*. If there is no percent sign, then **scout** highlights the partition's entry when fewer blocks are free than the *minimum blocks free* threshold.

  The column label displays the threshold value following the string "`Disk attn:`". Highlighting disappears when an entry returns below the percent threshold or above the blocks-free threshold.

  This statistic is the only one with a default value; that value is 95% full.

Note: You must stop **scout** and restart it to change the attention settings. Also, **scout** does not store the settings from the previous instance, so you need to specify all of the desired settings each time you set them.

The following example command line fragment causes **scout** to highlight any entries in the `Conn` column that exceed 100, any entries in the `Ws` column that exceed 50, and any entries in the `Disk attn` column that reflect partition usage greater than 75%. The `Fetch` and `Store` columns retain their default thresholds (i.e., none):

-attention conn 100 ws 50 disk 75%

This example is identical to the previous one, except that it causes **scout** to highlight any entries in the `Disk attn` column that fall below 5000 free blocks:

-attention disk 5000 ws 50 conn 100

The next example causes **scout** to highlight any entries in the `Fetch` column that exceed 6,000,000. The other columns retain their default values (95% full for the `Disk attn` column, no threshold for the other columns):

-attention fetch 6000000

Displaying Server Outages

You cannot control highlighting in the fifth column, where file server machine names appear. When the File Server process on a machine fails to respond to **scout**'s probe for statistics, **scout** automatically highlights the machine's name and removes the values in the other columns. Failure to respond to the probe may indicate a File Server process or file server machine crash, or a network outage. This highlighting helps you notice more quickly a situation that is probably interrupting service to users. The **scout** screen continues to display statistics for the machines that respond to its probes.

When the machine once again responds to **scout**'s probes, its name appears normally and values reappear in the other columns on its line. 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 **scout**.

7.2.1.5. About Resizing the Scout Screen

Do not resize the window while **scout** is running. Making the window larger than it originally was does no harm, but **scout** will not necessarily adjust to take advantage of the new dimensions. Making the window narrower can cause columns to line up incorrectly, making the display harder to read. With any type of resizing, **scout** does not adjust the display in any way until it is time to display results of the next probe.

If you wish to resize the window, stop **scout**, resize the window and then restart the program. Even in this case **scout**'s response to the resizing depends on the accuracy of information provided by 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, you may find that executing the following sequence of commands before starting **scout** will help it to adjust properly to resizing. You could also create an alias for the sequence and/or place it in a login or shell initialization file.

% **set noglob**
% **eval '/usr/bin/X11/resize'**
% **unset noglob**

7.2.2. Starting and Stopping scout

Before running **scout**, you may wish to check that you have fulfilled the requirements discussed in section 7.2.1.1.

⇒ To start scout

Step 1: Open a separate command shell window for each instance of **scout** you want to run.

Step 2: In each window, initialize scout. (You will not be able to issue any further commands in the window.)

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. Use full Internet-style host names, unless the **-basename** argument is used. In that case, specify only the unique initial part of each machine name, omitting the domain/cell name suffix (or "basename") common to all the machine names.

-basename specifies the basename (domain/cell name) suffix common to all of the file server machine names specified as *File Server name(s) to monitor*. See section 7.2.1.2 to learn the advantages of using this argument.

Do not include the period that separates the basename from the distinguishing part of each machine name, but do include any periods that occur within the suffix itself. (For example, in the Transarc Corporation cell, the proper value would be *transarc.com*, not *.transarc.com*.) If you provide this argument, omit the basename from the machine names named with *File Server name(s) to monitor*.

-frequency indicates how often **scout** should probe the File Server processes. Specify an integer number of seconds greater than *0*. The default is 60 seconds.

-host causes the banner line to display the name of the machine that is running **scout**. The default is no machine name displayed.

-attention defines a list of one or more entries, each of which pairs a type of statistic and a threshold value. When the value of the statistic exceeds the indicated threshold value, **scout** highlights the value. The pairs may appear in any order. Legal values for the pairs are as follows; section 7.2.1.4 explains how to use them.

- **conn** <*number of connections*>. Example: *conn 300*
- **fetch** <*number of fetch RPCs*>. Example: *fetch 6000000*
- **store** <*number of store RPCs*>. Example: *store 300000*
- **ws** <*number of active client machines*>. Example: *ws 65*
- **disk** <*percent full*>**%**. Examples: *disk 90%* and *disk 72%*

  or

  disk <*minimum blocks free*>. Examples: *disk 1500* and *disk 5000*

-debug enables debugging output and directs it into the specified file. Providing a complete pathname is best; otherwise the pathname is interpreted relative to the current working directory. By default, no debugging output is produced.

⇒ To stop scout

Type **^C** (Control-c) to exit **scout**. This is the proper interrupt signal to use even if the general interrupt signal in your environment is different.

7.2.3. Examples of scout Commands and Displays

This section presents examples of **scout** commands with different combinations of arguments and illustrations of the screen displays that result from them.

7.2.4. Example 1

The most basic use of Scout is to monitor the File Server processes running on one cell's file server machines. In this example (see Figure 7-1), an administrator in the Transarc Corporation issues the **scout** command without providing any optional arguments or flags. She includes the **-server** switch (abbreviated to **-s**), which is required because she is providing multiple instances of that argument. Even though she has not used the **-basename** argument, the administrator has chosen to provide only the first part of each server machine's name (for example, *ernie* rather than *ernie.transarc.com*). This is valid, but means that successful expansion to the complete machine name (necessary for **scout** to establish a connection with each machine) depends on the cell's name service being available when she issues the **scout** command.

% scout -s ernie bigbird

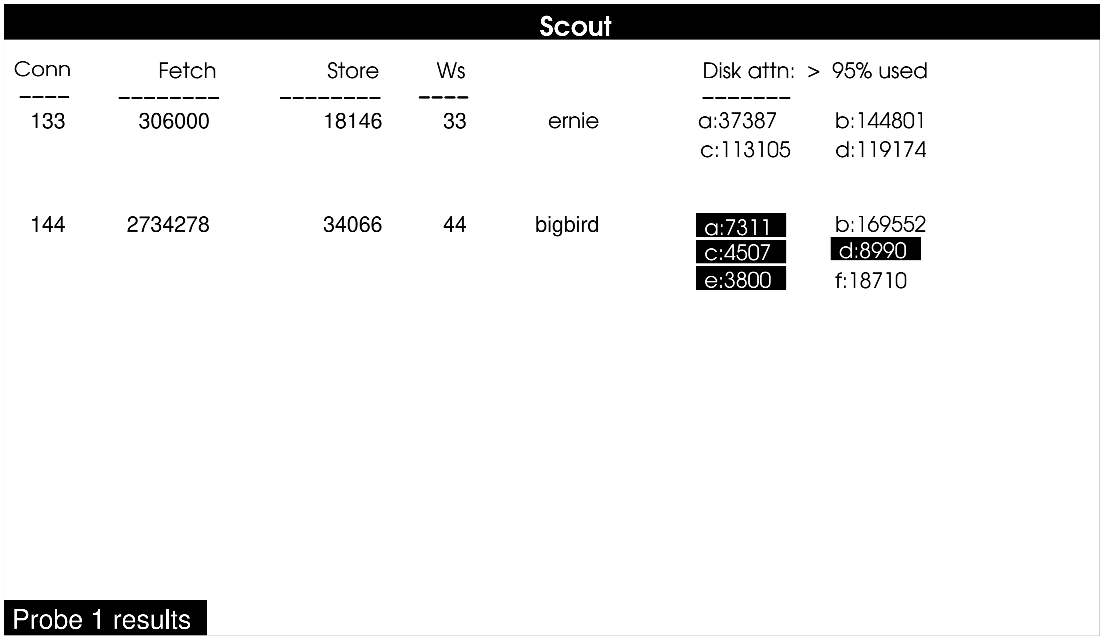

Figure 7-1. Scout Screen for Example 1

The first thing to notice about the screen is that the machine names in the fifth (unlabeled) column appear in the form the administrator provided them on the command line. As an example of how to interpret the other columns, consider the second line in the display region, where the machine name "`bigbird`" appears in the fifth column. The `Conn` and `Ws` columns together show that *bigbird* 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 `2734278` fetch RPCs to *bigbird* since the File Server process last started and the `Store` column shows that they have made `34066` 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 were wider, there would be 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).

7.2.4.1. Example 2

In this example, the Transarc cell administrator takes advantage of more of **scout**'s optional arguments (see Figure 7-2).

- She provides the machine names in the same form as in Example 1, but this time she also indicates their basename (*transarc.com*) with the **-basename** argument, abbreviating its switch to **-b**. This implies that expansion of the names to full form does not require the name service (though that service is still needed to convert the full names into network addresses).

- She uses the **-host** flag, abbreviated to its shortest unambiguous form **-ho**, so that the name of the client machine where this **scout** is running will appear in the banner line.
- She changes the probing frequency from its default of once per minute to once every five seconds, by specifying **-f 5** (the switch is short for **-frequency**).
- She changes the highlighting threshold for partitions to a 5000 block minimum rather than its default of 95% full, by specifying **-a disk 5000** (the switch is short for **-attention**).

% **scout -s ernie bigbird -b transarc.com -f 5 -ho -a disk 5000**

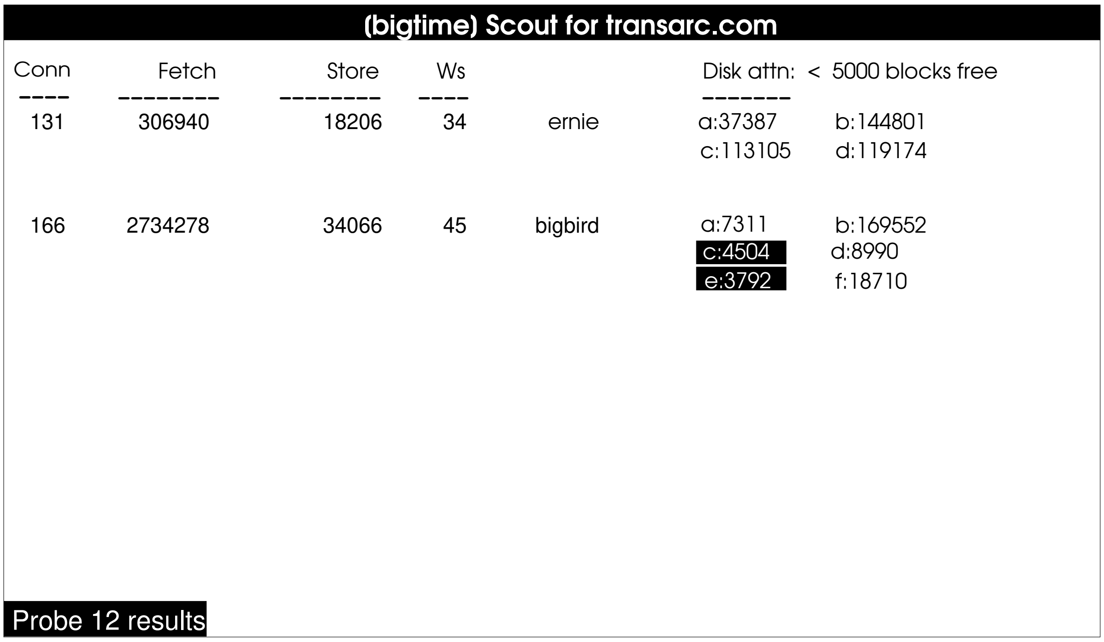

Figure 7-2. Scout Screen for Example 2

The use of optional arguments results in several differences between this screen and Example 1. First, because the **-host** flag is set, the banner line displays the name of the machine running **scout** as `[bigtime]` as well as the basename `transarc.com` specified with the **-basename** argument.

Another difference is that two, not four of `bigbird`'s partition entries appear in reverse video, even though their values are almost the same as in Example 1. This is because the administrator changed the highlight threshold to a 5000 block minimum with **-a disk 5000**, as reflected in the `Disk attn` column's header. And while `bigbird`'s partitions `a` and `d` are still 95% full, they have more than `5000` free blocks left; partitions `c` and `e` have fewer than `5000` free blocks left.

The final difference to note is the result of the change in probe frequency, as reflected in the probe/message line at the bottom left corner of the screen. Both this screen example and that of Example 1 represent a time lapse of one minute after the administrator issues the **scout** command. In Example 2, however, **scout** probed the File Server processes 12 times as opposed to once in Example 1.

7.2.4.2. Example 3

In this example, an administrator in the Andrew cell at Carnegie Mellon University monitors three of that cell's file server machines in one screen (see Figure 7-3). All of the machines have names that begin with "*vice*," so he uses the command shell's curly brace (**{ }**) notation to reduce the typing. (This notation is a feature of the command shell, not **scout** per se). He could also have typed each name individually, separating them with spaces (i.e., **vice2 vice3 vice4 . . .**). He uses the **-basename** argument to specify the *andrew.cmu.edu* basename.

% scout -s vice{2,3,4} -b andrew.cmu.edu

```
                         Scout for andrew.cmu.edu
Conn     Fetch      Store    Ws                 Disk attn:  > 95% used
----   ---------  --------- ----                -------
131     306940      18206    34     vice2       a:37387     b:144801
                                                c:113105    d:119174
166    2734278      34066    45     vice3       a:132456    b:169552
                                                c:99098     d:8990
                                                e:18710     f:18710
 86     906235      27945    37     vice4       a:65789     b:167856
                                                c:87690     d:5648

Probe 1 results
```

Figure 7-3. Scout Screen for Example 3

7.2.4.3. Example 4

The following example illustrates three concepts (see Figure 7-4). First, that it is possible to monitor file server machines from multiple cells in a single **scout** display. Here, an administrator in the Transarc Corporation cell chooses to monitor the machines

- *ernie.transarc.com* in the Transarc Corporation cell in Pittsburgh.
- *soup.mit.edu* at the Massachusetts Institute of Technology in Cambridge.
- *vice3.alw.nih.gov* at the National Institutes of Health in Bethesda.

Because the machines belong to different cells, it is not possible to provide the **-basename** argument. The administrator also chooses to specify the complete name for each machine to avoid potential problems with the name service in her cell.

Second, when machine names are too long to fit in the fifth column, **scout** truncates them, placing an asterisk in the final slot to indicate that it has done so.

And third, when **scout** cannot reach a file server machine, the machine name is highlighted to indicate the connection problem. (This can result from the File Server process or machine crashing, or from a network failure.) In this case, **scout** cannot reach *vice3.alw.nih.gov*. The machine name appears in reverse video to reflect its status, and all the other fields are blank. Because the outage does not affect the other two machines that **scout** is monitoring, their entries remain complete.

% scout -s vice1.transarc.com soup.mit.edu vice3.alw.nih.gov

Figure 7-4. Scout Screen for Example 4

7.3. AFSMonitor

7.3.1. About AFSMonitor

AFSMonitor (command/process name: **afsmonitor**) allows you to monitor the status and performance of specified File Server and Cache Manager processes by gathering statistical information on those processes. Among its other uses, **AFSMonitor** can be used to fine-tune Cache Manager configuration and load balance File Servers.

Using **AFSMonitor**, you can

- 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, **AFSMonitor** highlights it to draw your attention. Threshold levels can be set globally and customized for specific processes.
- Set up programs/scripts to run automatically when a statistic exceeds its threshold.

7.3.1.1. Requirements for Running AFSMonitor

The **AFSMonitor** program can only be run on systems that

- Support the AFS xstat libraries (see section 7.4) which **AFSMonitor** relies upon to gather its data.
- Includes the "curses" graphics package which is provided as a standard utility on most UNIX distributions.

The **AFSMonitor** screens can be displayed on both "dumb" terminals and windowing systems that emulate terminals. The **AFSMonitor** screens make use of reverse video and cursor addressing so for the screen to look its best, the display environment should support those features. You should set the environment variable `TERM` to the correct terminal type or one with characteristics similar to the type of terminal you are using. The display window or terminal should be at least 80 columns wide and 12 lines long.

No privilege is required to run **AFSMonitor**; its presence in */usr/afsws/bin* on a machine means that any user of that machine may monitor File Server and Cache Manager processes as long as */usr/afsws/bin* appears in his or her path. The probes through which **AFSMonitor** collects statistics do not constitute a significant burden on the File Server 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.

Any number of instances of **AFSMonitor** may run on a single machine; however, each instance runs in a separate dedicated window or on a dedicated terminal. Once the process is initiated, you will be unable to perform other activities in the dedicated window or on the dedicated terminal until the session is exited. The **AFSMonitor** program must run in the foreground.

7.3.1.2. The Layout and Contents of the AFSMonitor Screens

The **AFSMonitor** program displays its data on three screens:

- **System Overview screen**: The System Overview screen is the first screen displayed when **AFSMonitor** is executed. It provides summary information on the monitored processes. The screen provides information on the number of processes monitored, the host associated with each process, and the alert status of each process (that is, the number of statistics above a threshold value and whether or not a process failed to respond to the last probe). The information on this screen is categorized into Cache Manager and File Server groupings.
- **File Servers screen**: This screen displays detailed statistical information on each File Server process being monitored. Processes that failed to respond to the last probe are identified; statistics that exceed threshold values are highlighted.
- **Cache Managers screen**: This screen displays detailed statistical information on each Cache Manager process being monitored. Processes that failed to respond to the last probe are identified; statistics that exceed threshold values are highlighted.

Standard Screen Features

Each screen displays four standard blocks of information, one in each corner:

- In the top left corner, the program name and version number are displayed.
- In the top right corner, the screen title, current and total page numbers, and current and total column numbers are displayed. The page numbers (for example, p. 1 of 3) indicate the current page of information that is displayed followed by the total number of pages over which all available data is vertically displayed. The column numbers (e.g., c. 1 of 235) indicate the current left-most column displayed followed by the total number of columns over which all available data is horizontally displayed. (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 available commands is displayed. Typing the first letter in the command name will execute that command. Only those commands that can be executed will be displayed; for example, if there is only one page of data, there is no reason to scroll up or down, so the **n**ext and **p**rev commands will not appear. See section 7.3.1.4 for more information on these commands.
- In the bottom right corner, the probe count and probe frequency are displayed. The probe counts indicate the number of times that the File Servers (fs) and/or Cache Managers (cm) have been probed for statistics. These numbers can be different depending upon the number of Cache Managers and File Servers being monitored and the probe frequency. The frequency indicates how often (in seconds) probes are initiated.

System Overview Screen

When **AFSMonitor** is initialized, the System Overview screen is displayed. This screen shows the status of as many File Server and Cache Manager processes as can be initially displayed; additional data can be accessed by scrolling up and down.

The information on this screen is categorized into Cache Manager and File Server groupings. The left half of the screen displays File Server information and the right half displays Cache Manager information. At the top of each grouping there are two pieces of information:

- The number of processes (whether Cache Manager or File Server) monitored by this **afsmonitor** process.
- The number of alerts and the number of processes (machines) affected by those alerts. (Each alert represents a single instance of a statistic exceeding its threshold or of a process failing to respond to the last probe.)

Under this status information is a list of the hosts containing the monitored processes. If one or more of a process's statistics have exceeded a threshold, the number of above-threshold statistics are listed in square brackets to the left of that process. If a process failed to respond to the last probe, the letters "PF" (probe failure) appear in square brackets to the left of that process.

Figure 7-5 gives an example of a System Overview screen. In this example, **AFSMonitor** is monitoring six File Server and seven Cache Manager processes. The Cache Manager process on host *grungetta* and the File Server process on host *colby* have each been highlighted for exceeding one threshold value. No response was received from the File Server process on host *henson* during 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] grungetta.transarc.com                 grungetta.transarc.com
         junior.transarc.com                    bigbird.transarc.com
         bigbird.transarc.com                   junior.transarc.com
         ernie.transarc.com                     oscar.transarc.com
         oscar.transarc.com                     ernie.transarc.com
  [ PF] henson.transarc.com                [   1] colby.transarc.com
                                                drseuss.transarc.com






Command [fs, cm]?                            [probes 9(fs) 8(cm), freq=60 sec]
```

Figure 7-5. System Overview Screen

File Servers Screen

Executing the **fs** command option from either the System Overview or Cache Managers screen displays the File Server screen. This screen provides detailed statistical information collected from the File Servers during the last probe.

The File Servers screen is laid out as follows:

- A summary line is displayed at the top of the screen just below the standard program version and screen title blocks. It specifies the number of monitored processes, the number of alerts, and the number of processes (machines) affected by the alerts.
- The names of the hosts containing the monitored File Server processes are displayed in separate rows down the left side of the screen. These host names are always displayed, even when the screen is scrolled left and right.
- As many columns of data as can fit within the confines of the window are displayed to the right of the host names. Each column is labeled with the name of its corresponding statistic. The order and content of the statistical display area is configurable. As the screen is scrolled left and right, these columns change to display new data. If the last probe failed to contact a File Server, the fields in that row will contain dashes (--); otherwise, some value will be displayed. If a value exceeds its configured threshold, it will be highlighted in reverse video. The columns are 10 characters wide; if a particular value is too large to fit into one column, it overflows into the next row in the same column. (See section 7.3.1.5 for information on configuring the statistical display area and the threshold values for statistics. See Appendix A for information on the available statistics.)

Figure 7-6 depicts a File Servers screen that continues the example begun in Figure 7-5; however, one additional server probe has been completed. In this example, the File Server process on *grungetta* has exceeded the configured threshold for the number of performance calls received (**numPerfCalls**) so that field is displayed in reverse video. Probe 10 failed to contact the File Server process on *henson*; therefore, dashes appear in the statistics fields for *henson*.

AFSMonitor [Version 1.0] [File Servers, p. 1 of 1, c. 1 of 235]

6 File Servers monitored, 2 alerts on 2 machines >>>

	num PerfCalls	vcache L_Entries	vcache L_Allocs	vcache L_Gets	vcache L_Reads	vcache L_Writes
grungetta	236	400	1715	1415067	33149	95620
junior	174	400	14466	14177920	408586	921456
bigbird	123	400	22300	11766145	430462	1138833
ernie	188	400	22318	12256574	458441	1145729
oscar	210	400	4059	1256726	29334	149236
henson	--	--	--	--	--	--

Command [oview, cm, right]? [FS probes 10, freq=60 sec]

Figure 7-6. File Servers Screen

Both the File Servers and Cache Managers screen can display hundreds of columns of data and are therefore designed to scroll left and right. In Figure 7-6, 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. (See Appendix A for information on the available statistics.)

If the **right** command is executed, the screen might look something like the example pictured in Figure 7-7. Note that the horizontal scroll symbols now point to both the left (<<<) and 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 S_Entries	vcache S_Allocs	vcache S_Gets	vcache S_Reads	vcache S_Writes	vcache H_Entries
grungetta	400	49338	1757652	390217	471959	200
junior	400	0	0	0	0	200
bigbird	400	379851	25189529	5777415	12499650	200
ernie	400	330685	19103031	4109113	9845424	200
oscar	400	75035	2240386	572887	678820	200
henson	--	--	--	--	--	--

Command [oview, cm, left, right]? [FS probes 10, freq=60 sec]

Figure 7-7. File Servers Screen Shifted One Page to the Right

Cache Managers Screen

Executing the **cm** command from either the System Overview or File Servers screen displays the Cache Managers screen. This screen provides detailed information collected from the Cache Managers for the latest probe.

The Cache Managers screen is laid out as follows:

- A summary line is displayed at the top of the screen just below the standard program version and screen title blocks. It specifies the number of monitored processes, the number of alerts, and the number of processes (machines) affected by the alerts.
- The names of the hosts containing the monitored Cache Manager processes are displayed in separate rows down the left side of the screen. These host names are always displayed, even when the screen is scrolled left and right.

- As many columns of data as can fit within the confines of the window are displayed to the right of the host names. Each column is labeled with the name of its corresponding statistic. The order and content of the statistical display area is configurable. As the screen is scrolled left and right, these columns change to display new data. If the last probe failed to contact a Cache Manager, the fields in that row will contain dashes (--); otherwise, some value will be displayed. If a value exceeds its configured threshold, it will be highlighted in reverse video. The columns are 10 characters wide; if a particular value is too large to fit into one column, it overflows into the next row in the same column. (See section 7.3.1.5 for information on configuring the statistical display area and the threshold values for statistics. See Appendix A for information on the available statistics.)

Figure 7-8 depicts a Cache Managers screen that continues the example begun in Figure 7-5. In the example, the Cache Manager process on *colby* has exceeded the configured threshold for the number of cells it can contact (**numCellsContacted**) and so that field is displayed 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 PerfCalls	epoch	numCells Visible	numCells Contacted	dlocal Accesses	vlocal Accesses
grungetta	830	725547565	92	1	33980	49653
bigbird	1959	717359638	96	1	219700	254185
junior	207	731803376	94	3	58239	84194
oscar	829	727131965	92	1	32567	35478
ernie	527	729182137	92	1	17926	17661
colby	437	731371085	84	2	2566159	2162551

Command [oview, fs, next, right]? [CM probes 7, freq=60 sec]

Figure 7-8. Cache Managers Screen

7.3.1.3. Starting and Stopping AFSMonitor

Before running **AFSMonitor**, you may wish to check that you have fulfilled the requirements discussed in section 7.3.1.1.

⇒ To start AFSMonitor

Step 1: Open a separate command shell window or use a dedicated terminal for each instance of **AFSMonitor**. This window or terminal must be devoted to the exclusive use of the **afsmonitor** process because the command cannot be run in the background.

Step 2: Initialize **AFSMonitor**:

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*>$^{+}$**]**
[-buffers <*number of buffer slots*>**] [-help]**

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 a configuration file to be used by **afsmonitor**; the configuration file contains a list of the processes to monitor, the threshold values to alert on, and the statistics to display. If a complete pathname is not provided the file name is interpreted relative to the current directory. Although this argument is optional, you must use either this argument or the **-fshosts** and/or **-cmhosts** arguments. Without a configuration file, you cannot set thresholds or customize the screen display. See 7.3.1.5 for more information on creating this file.

-frequency specifies the frequency in seconds with which File Server and Cache Manager processes are probed. You may use a value between 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 **afsmonitor** will write all of the statistical data that it collects. By default, no output file is created. See section 7.3.1.6 for information on this file.

-detailed indicates that the output file (specified with the **-output** argument) should be laid out in a more easily readable format. This flag should only be used with the **-output** argument.

-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. Do not use this argument with the **-config** argument; it may be used with the **-cmhosts** 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. Do not use this argument with the **-config** argument; it may be used with the **-fshosts** argument.

-buffers specifies the number of circular buffer slots that **afsmonitor** uses to store the data it collects. This facility is provided for future enhancements to the **AFSMonitor** and provides no functionality at this time.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

Step 3: A successful startup of the **afsmonitor** will cause the message `AFSMonitor Collecting Statistics ...` to appear followed by the System Overview screen.

⇒ To stop AFSMonitor

To exit an **AFSMonitor** session, type **^C** (Control-c) or an uppercase **Q**.

7.3.1.4. Navigating the AFSMonitor Screens

There are several commands available within the **AFSMonitor** screens to help you move within a screen and from screen to screen. The commands currently available are always displayed in the lower left hand corner of the screen. To execute a command, type its initial lowercase letter; the **left**, **right**, **next**, and **prev** commands offer slightly different functionality when the corresponding uppercase letter is typed. Below is a list of all the commands and a description of their use:

cm access the Cache Managers screen. This command is available from the System Overview and File Servers screen.

fs access the File Servers screen. This command is available from the System Overview and the Cache Managers screen.

left access the next set of data columns situated to the left of the current set (i.e., scroll the screen horizontally to the left). This command is available on the File Servers and Cache Managers screens when the <<< symbol appears in the upper left portion of the screen. Typing an uppercase **L** will display the leftmost column of a screen.

next scroll the screen vertically to the next page. This command is available on a screen if the screen contains at least two pages and you are not currently viewing the last page. Typing an uppercase **N** will display the last page of a screen.

oview access the System Overview screen. This command is available from the Cache Managers and File Servers screen.

prev scroll the screen vertically to the previous page. This command is available on a screen if the screen contains at least two pages and you are not currently viewing the first page. Typing an uppercase **P** will display the first page of a screen.

right access the next set of data columns situated to the right of the current set (i.e., scroll the screen horizontally to the right). This command is available on the File Servers and Cache Managers screens when the >>> symbol appears in the upper right portion of the screen. Typing an uppercase **R** will display the rightmost column of a screen.

7.3.1.5. Configuring AFSMonitor

You can configure **AFSMonitor** by creating a simple ASCII file and specifying that file upon startup with the **-config** argument. With a configuration file, you can

- Specify the File Servers and/or Cache Managers to monitor.
- Specify the statistics to display. By default, all of the statistics that are gathered are displayed; there are 271 statistics for the File Servers screen and 570 statistics for the Cache Managers screen. See Appendix A for information on the available statistics.
- Assign threshold values to statistics and specify 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.

Each line in an **AFSMonitor** configuration file must begin with one of the following words/symbol and use the specified format:

- **cm** <*host_name*>

  This line indicates that the Cache Manager process on *host_name* is to be monitored. It will be added to the list of such processes and will appear on the System Overview and Cache Managers screens in the order specified in this file.

- **fs** <*host_name*>

  This line indicates that the File Server on *host_name* is to be monitored. It will be added to the list of such processes and will appear on the System Overview and File Servers screens in the order specified in this file.

- **thresh** <**fs**|**cm**> <*field_name*> <*thresh_val*> [<*cmd_to_execute*>] [<arg_1>] . . . [<arg_n>]

  This line assigns the threshold value <*thresh_val*> to <*field_name*> (i.e., a statistic field) for either File Servers or Cache Managers (<**fs**|**cm**>). If the <*cmd_to_execute*> option appears, the specified script/program (with the its full pathname) will be executed each time the corresponding threshold value is exceeded; though it is not executed if both the new value and the previous value are above the threshold.

  When the *cmd_to_execute* is executed, the following variables will be passed to it as arguments:

  <*host_name*> <**fs**|**cm**> <*field_name*> <*threshold_val*> <*actual_val*> [<arg_1>] . . . [<arg_n>]

  The variables correspond to the **thresh** line and the affected host machine; the *actual_val* represents the actual value of *field_name* that forced it over the threshold value. The values arg_1 through arg_n represent additional optional arguments that can be used as needed; if any of these arguments contain spaces, enclose the argument in double quotes.

  A **thresh** line can be used to set either a global threshold value or a process-specific threshold value. To define a global threshold, place the **thresh** line before any **fs** or **cm** lines. A global **thresh** line applies to all monitored File Server or Cache Manager processes. To define a process-specific threshold value, place the **thresh fs** or **thresh cm** line after the appropriate **fs** or **cm** line, respectively and prior to any other **fs** or **cm** lines. A process-specific **thresh** line applies only to the preceding File Server or Cache Manager process. A process-specific threshold value always overrides a corresponding global value.

- **show** <**fs**|**cm**> <*field/group/section*>

  The **show** line is used to customize the File Servers and/or Cache Managers screen so that the screen provides only the statistics of interest, listed in the specified order. A **show** line instructs **AFSMonitor** to display only the statistics fields identified by that and other **show** lines for the corresponding Cache Managers (**cm**) or File Servers (**fs**) screen. A **show** line can appear anywhere in the configuration file and you can include as many as necessary to customize the displays to your liking. The order of the statistics field on the corresponding screen (from left to right) will follow the order established by the **show** lines in the configuration file (from top to bottom).

If there are no **show** lines, all statistics fields for both Cache Managers and File Servers are displayed. If only **show fs** lines are included, the specified File Server statistics fields are displayed and all Cache Manager statistics fields are displayed. If only **show cm** lines are included, the specified Cache Manager statistics fields are displayed and all File Server statistics are displayed.

Each statistic monitored by **AFSMonitor** is part of a statistics group and each group is a part of a statistics section. The *field/group/section* variable is used to specify a single field, group, or section to be displayed. If a field is specified, only that statistic field is displayed; if a group or section is specified, all of the statistics fields that are members of that group or section are displayed. (See Appendix A for information on the available statistics.)

- **#** <*comments*>

  Begin comment lines with a pound symbol (#) in the very first column of the line.

Consider the following example configuration file:

```
thresh cm dlocalAccesses  1000000
thresh cm dremoteAccesses  500000 handleDRemote
thresh fs rx_maxRtt_Usec     1000
cm bigtime
cm drama
cm brick
thresh cm dlocalAccesses  2000000
thresh cm vcacheMisses      10000
cm apollo
fs ernie
fs oscar
fs bigbird
fs fozzie
show cm numCellsContacted
show cm dlocalAccesses
show cm dremoteAccesses
show cm vcacheMisses
show cm Auth_Stats_group
```

Since the first three **thresh** lines appear before any **fs** or **cm** lines, they set global threshold values:

- All Cache Manager process in this file use the global threshold value of 1000000 for **dlocalAccesses** (with the exception of *brick* where the global value is overridden by a process-specific value of 2000000.)
- All Cache Manager processes in this file use the global threshold value of 500000 for **dremoteAccesses**; if that value is exceeded, the script **handleDRemote** will be invoked.
- All File Server processes in this file use the global threshold value of 1000 for **rx_maxRtt_Usec**.

The Cache Managers *bigtime*, *drama*, *brick*, and *apollo* will be monitored and all but *brick* inherit all of the global threshold values. Cache Manager *brick* uses the global threshold value for **dremoteAccesses**, but it uses a process-specific threshold value for **dlocalAccesses**. Furthermore, *brick* is the only Cache Manager for which a threshold has been set for the **vcacheMisses** field.

The File Servers *ernie*, *oscar*, *bigbird*, and *fozzie* will also be monitored and they all inherit the global **rx_maxRtt_Usec** threshold value.

All statistics fields will be displayed on the File Servers screen because there are no **show fs** lines in the configuration file. On the Cache Managers screen, the following statistics fields only will be listed (left to right) based on the statistics fields specified by the **show cm** lines: **numCellsContacted**, **dlocalAccesses**, **dremoteAccesses**, and **vcacheMisses**; and the following, which belong to the **Auth_Stats_group**: **curr_PAGs**, **curr_Records**, **curr_AuthRecords**, **curr_UnauthRecords**, **curr_MaxRecordsInPAG**, **curr_LongestChain**, **PAGCreations**, **TicketUpdates**, **HWM_PAGS**, **HWM_Records**, **HWM_MaxRecordsInPAG**, and **HWM_LongestChain**.

7.3.1.6. Creating an Output File for AFSMonitor Statistics

All of the statistical information collected and displayed by **AFSMonitor** may 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. The ability to record this information allows you to analyze process performance over long periods of time and to apply post-processing techniques to further analyze system trends.

The **AFSMonitor** 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*>

The line 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, a single *-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, you may want to use the **-detailed** flag with the **-output** argument. This flag will cause the output file to be formatted in a more user-readable manner.
- 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 **AFSMonitor** output.

7.4. The xstat Data Collection Facility

The **AFSMonitor** program uses the ***xstat*** data collection facility to gather and calculate the data that it (**AFSMonitor**) then uses to perform its function. You can also use **xstat** 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.

7.4.1. The *xstat* 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.

Detailed specifications providing much additional usage information about *xstat*, its libraries, and the routines in the libraries can be accessed as follows:

- ***via AFS***, in the directory */afs/grand.central.org/pub/doc/afs/specs/xstat*
- ***via anonymous ftp*** to *grand.central.org*, IP address *192.54.226.100*, in the directory *pub/doc/afs/specs/xstat*

If you do not have network access, contact your AFS Product Support representative to arrange to have copies of the *xstat* document sent to you.

7.4.2. 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 collect data about file servers

Step 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.

% **/usr/afsws/bin/xstat_fs_test [initcmd]**
-fsname <*File Server name(s) to monitor*>$^+$
-collID <*Collection(s) to fetch*>$^+$ **[-onceonly]**
[-debug] [-help]

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 whose File Server process is to be monitored.

-collID specifies each data collection to be returned by the command. 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).

Three legal arguments exist for this argument:

- *0* provides profiling information about the numbers of times different internal File Server routines were called since the File Server was started. This value is not currently implemented; it returns no data.
- *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 with the value of *1*, 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 type **^c** (Control-c) to stop execution of the command whenever you want.

-debug causes the command to display additional information as it executes.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

The command produces a fair amount of output. You may want to direct the output to a file that you can examine later.

⇒ To collect data about Cache Managers

Step 1: Enter 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.

% /usr/afsws/bin/xstat_cm_test [initcmd]
-cmname <*Cache Manager name(s) to monitor*>+
-collID <*Collection(s) to fetch*>+ **[-onceonly]**
[-debug] [-help]

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 Internet host name of each client machine whose Cache Manager is to be monitored.

-collID specifies each data collection to be returned by the command. 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).

Three legal arguments exist for this argument:

- *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 with the value of *1*, 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 type **^c** (Control-c) to stop execution of the command whenever you want.

-debug causes the command to display additional information as it executes.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

As with the **xstat_fs_test** command, this command produces a fair amount of output. You may want to direct the output to a file that you can examine later.

7.5. 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. Appendix B contains a complete list of specific AFS audit events.

- 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 How to Set Up an Auditing System in the *AIX 3.2 System Management Guide*.

7.5.1. Enabling and Disabling AFS Auditing

Enabling and disabling AFS auditing on an AIX File Server are relatively simple tasks. However, both tasks requires restarting the *fileserver* process, so you should perform them during periods of low File Server usage, if possible.

⇒ To enable AFS audit

Step 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
```

Step 2: Issue the **bos restart** command (with the **-all** flag) to stop and restart all server processes on each File Server. See section 4.8 for instructions on using this command.

⇒ To disable AFS auditing

Step 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.

Step 2: Issue the **bos restart** command (with the **-all** flag) to stop and restart all server processes on each File Server. See section 4.8 for instructions on using this command.

7.5.2. Configuring AFS Auditing on AIX File Servers

The directory */usr/afs/local/audit* contains three files that contain the information you will need to configure AIX File Servers to audit AFS events:

- The *events.sample* file contains information on auditable AFS events. The contents of this file should be 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 that will be audited for the File Server user "root". The contents of this file should 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 should only audit files in the local file space. The contents of this file should 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.

8. Managing Authentication and Authorization Requirements

This chapter discusses the means through which server processes guarantee that only properly authorized users perform privileged commands: authorization checking and mutual authentication. The chapter further explains how system administrators control authorization checking and how the issuer of a command may disable mutual authentication in certain cases.

It has the following sections:

8.1 Summary of Commands 8-1
8.2 Authentication versus Authorization 8-2
8.3 Controlling Authorization Checking for a File Server Machine 8-3
8.4 Preventing Mutual Authentication for an Individual Command 8-5

8.1. Summary of Commands

Set authorization checking	**bos setauth**
Prevent authentication with bos commands	add **-noauth** flag
Prevent authentication with fs commands	not possible
Prevent authentication with kas commands	**(kas) noauthentication** or add **-noauth** flag on some commands
Prevent authentication with pts commands.	add **-noauth** flag
Prevent authentication with vos commands	add **-noauth** flag

8.2. Authentication versus Authorization

Many commands in the AFS command suites are "privileged"—the AFS server process invoked by the command will execute them only for properly authorized users. To determine if someone is properly authorized, the server process performs two tests before executing a command: authentication and authorization checking.

- In the ***authentication*** test, the server mutually authenticates with the issuer of the command (or rather, with the process representing him or her). The goal of this test is to determine who is issuing the command. Mutual authentication allows the server process to verify that the issuer really is who he or she claims to be, by examining the ticket and other elements in 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 assigns him or her the identity *anonymous*, a completely unprivileged user. See section 1.5 for a more complete explanation of mutual authentication.

  Many individual commands allow the issuer to disable the authentication test. The server process assigns the identity *anonymous* rather than attempting to mutually authenticate with him or her. Disabling mutual authentication is useless if the server process is still checking for authorization (see just below), because the server will refuse to perform privileged commands for *anonymous*.

- 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 may do. Different server processes consult different lists, and you as a system administrator determine who appears in them; see chapter 20. The server process refuses to execute any privileged command for an unauthorized issuer. If the command has no privilege requirements (that is, anyone is allowed to perform it), the server process skips this step and performs the command immediately.

  As a system administrator, you control whether the server processes on a certain file server machine check for authorization, using the **bos setauth** command. If you turn off authorization checking, the server processes on that machine perform any action for any user. The command affects only the machine on which it is issued; the other machines retain their current status.

Note: You should never place the inherently unprivileged user *anonymous* or the group *system:anyuser* on a privilege list, or authorization checking becomes worthless.

8.3. Controlling Authorization Checking for a File Server Machine

8.3.1. When to Turn Off Authorization Checking

Turning off authorization checking is a serious breach of security because it means that the processes on a file server machine will perform any action for any user, even *anonymous*.

The only time it is common to turn off authorization checking is when you are installing a new file server machine (see the *AFS Installation Guide*). It is necessary then because the machine is not yet connected into AFS and so processes on the machine cannot check the lists of privileged users (in the case of installing the cell's first file server machine, the lists do not even exist yet). For greatest safety, the installation instructions have you work at the console of the machine you are installing and then turn on authorization checking as soon as is viable.

During normal operation, the only reason you should turn off authorization checking is if an error occurs with the server encryption keys, leaving the servers unable to authenticate users properly. Section 10.6 explains how to handle key-related emergencies.

8.3.2. How the Server Processes Know When to Check for Authorization

The server processes constantly monitor the directory */usr/afs/local* on their local disks to determine if they should 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 do check for authorization.

You as a system administrator control the presence of *NoAuth* through the BOS Server. When you turn off 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 *NoAuth* as a 0-length file in */usr/afs/local*. When you turn on authorization checking, the BOS Server removes the file.

8.3.3. Controlling Authorization Checking for the Entire Cell

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 will 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.

8.3.4. The Need to Restart Processes

As mentioned, the server processes constantly monitor for the presence of *NoAuth*. When you change authorization checking requirements, they immediately obey the new requirements for their connections with clients. However, the server processes also have established connections among themselves. For instance, the File Server keeps open a connection with the Protection Server so that it can request CPSs for new users without having to establish a new connection each time. The Ubik lightweight processes for the database server processes all have connections open to one another as well. You need to restart the servers to make sure that they destroy these connections and open new ones that obey the current authorization requirements. It is easiest to restart all the processes.

⇒ To turn off authorization checking

Step 1: Verify that you have the necessary privilege (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Turn off authorization checking.

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

where

seta is an acceptable abbreviation of **setauth**.

machine name specifies the file server machine on which server processes should not check for authorization.

Step 3: Stop and restart all the processes on the file server machine except the BOS Server, which does not maintain connections with other processes. This will cause an outage; the *fs* process lasts the longest because it depends on the number of volumes on the machine. The **-noauth** flag is necessary to prevent an authentication attempt while authorization checking is turned off. Section 4.8 describes the **bos restart** command fully.

% **bos restart** <*machine name*> **-noauth**

⇒ To turn on authorization checking

Step 1: Turn on authorization checking. (You need not be privileged because the machine is not currently checking for authorization.)

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

where

seta is an acceptable abbreviation of **setauth**.

machine name specifies the file server machine on which authorization checking should return to normal (server processes check for authorization).

Step 2: Stop and restart all the processes on the file server machine except the BOS Server, which does not maintain connections with other machines. This will cause an outage; the *fs* process' lasts the longest because it depends on the number of volumes on the machine. Section 4.8 describes the **bos restart** command fully.

% **bos restart** <*machine name*>

8.4. Preventing 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*.

8.4.1. When to Prevent Mutual Authentication

The facilities for preventing mutual authentication are provided for use in emergencies (such as the key emergency discussed in section 10.6). During normal circumstances, authorization checking should be turned on, making it useless to prevent authentication: the server processes will refuse to perform privileged commands for *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 prevent mutual authentication with bos commands

Include the **-noauth** flag on the command line. If you wish to abbreviate the flag, consult the *AFS Command Reference Manual* to learn which shortened form is acceptable for a given command. The **bos help** and **bos apropos** commands do not accept this flag.

⇒ To prevent mutual authentication with fs commands

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

⇒ To prevent mutual authentication with kas commands

The majority of the **kas** commands allow you to include a **-noauth** flag on the command line, including **kas create**, **kas debuginfo**, **kas delete**, **kas examine**, **kas getrandomkey**, **kas interactive**, **kas list**, **kas setfields**, **kas setkey**, **kas setpassword**, and **kas statistics**.

The availability of **-noauth** on the **kas interactive** command means that you can prevent mutual authentication for any **kas** command by entering interactive mode before you issue it. If you have entered interactive mode with an authenticated identity, issuing the **noauthentication** command allows you to discard that identity and become *anonymous*.

⇒ To prevent mutual authentication with pts commands

Include the **-noauth** flag on the command line. If you wish to abbreviate the flag, consult the *AFS Command Reference Manual* to learn which shortened form is acceptable for a given command. The **pts help** and **pts apropos** commands do not accept this flag.

⇒ To prevent mutual authentication with vos commands

Include the **-noauth** flag on the command line. If you wish to abbreviate the flag, consult the *AFS Command Reference Manual* to learn which shortened form is acceptable for a given command. The **vos help** and **vos apropos** commands do not accept this flag.

9. Maintaining *CellServDB* on File Server Machines

Every file server machine in a cell maintains a list of the cell's database server machines in the file */usr/afs/etc/CellServDB* on its local disk. Section 1.4.4 explains database server machines.

As a system administrator, you must make sure that the information in *CellServDB* is accurate at all times. You will modify the contents of this file whenever you add or delete a database server machine from your cell.

This chapter explains completely the function of *CellServDB* and describes how to maintain it. It has the following sections:

9.1 Summary of Commands 9-1
9.2 About *CellServDB* 9-2
9.3 Listing the Contents of *CellServDB* 9-5
9.4 Adding Database Server Machines to *CellServDB* 9-6
9.5 Removing Database Server Machines from *CellServDB* 9-8

9.1. Summary of Commands

List the contents of *CellServDB* . . . . **bos listhosts**
Add a database server machine to *CellServDB*. . . . . **bos addhost**
Remove a database server machine from *CellServDB* . . . . **bos removehost**

9.2. About *CellServDB*

The processes running on every file server machine in a cell use the listing of database server machines kept in the file */usr/afs/etc/CellServDB* on the local disk, for the following reasons:

- The database server processes (Authentication, Backup, Protection, and Volume Location Servers) themselves must know about their peers in order to keep their database copies identical.

  Recall from section 2.13 that each database server process runs a "Ubik" lightweight process responsible for synchronizing the local database with the corresponding database on other database server machines. For each database, one of the Ubik processes is elected as the "coordinator." It alone accepts changes to the information in the database and distributes them to each "secondary" Ubik process for incorporation into its copy of the database. The coordinator consults *CellServDB* to learn where the secondary database copies reside. *CellServDB* also tells the coordinator how many other Ubik processes there are; it must remain in contact with a majority of them to remain the coordinator.

  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 *CellServDB* 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. One common example: when a File Server process needs to determine if a user possesses the privileges necessary to access a file in the requested manner, it contacts a Protection Server (which consults the Protection Database) to learn which groups a user belongs to (for more on this, see section 18.2.5.1).

9.2.1. Negative Consequences of an Incorrect *CellServDB*

Negative consequences result when *CellServDB* does not accurately list the cell’s database server machines. The exact consequence depends on whether the error is one of omission or inclusion. (The following discussion assumes familiarity with the Ubik concepts presented in section 2.13.)

- If a database server machine does not appear in any copy of *CellServDB* (server version or client version), then the machine effectively is not a database server machine. The Ubik coordinator will not send it updates, nor include it in its count that establishes a majority. It will not be included in the election of a new coordinator when that becomes necessary. The non-database server processes will never contact the database server processes on the machine, leaving the processes under-utilized. Client machines will never contact it either. None of these effects are particularly serious, but running an unused server process is inefficient.

  If you include a database server machine in the client version but not the server version of *CellServDB*, the consequences could be more serious. Clients will contact the database server processes running on it, but the processes will refuse to give out data because they are not in touch with their respective coordinators. The clients generally wait for one server to time out before contacting another, so users are likely to experience a delay whenever they contact this machine.

- If *CellServDB* includes a machine that is not actually a database server machine, the consequences can be serious. The Ubik coordinator will be unable to send it updates, but will include it in the count it uses to determine if it is receiving votes from the majority. If other Ubik sites go down, then the coordinator may believe it has lost the necessary majority, even though it is still in contact with a majority of the actual database server machines. This will force election of a new coordinator, causing a service outage, during which the database is completely unavailable. The election may not proceed smoothly, either, if the lack of a vote from the nonexistent site makes the other sites believe that a majority of sites are not voting for the new coordinator.

  A more minor consequence of wrongly including a machine is that non-database server processes will attempt to contact the database server processes on the machine. Since the processes do not actually exist, there may be a delay when the non-database process waits for a timeout before contacting another database server machine.

9.2.2. When to Update *CellServDB*

You must update */usr/afs/etc/CellServDB* whenever you add or remove a database server machine in your cell. Failure to do so can result in the negative consequences discussed previously.

The *AFS Installation Guide* provides complete instructions for installing or removing a database server machine and for updating *CellServDB* in that context. This chapter is mainly for reference.

9.2.3. How to Maintain *CellServDB*

In cells using the United States edition of AFS, the recommended way to guarantee that every file server machine's copy of */usr/afs/etc/CellServDB* is correct is to keep a central copy on the cell's system control machine. This arrangement has several implications for administrators in such cells:

- The system control machine must run the server portion of the Update Server. All the other file server machines in the cell should run a client portion of the Update Server that points to the system control machine. The *AFS Installation Guide* explains how to install server and client portions of the Update Server on the appropriate machines.
- You should change *CellServDB* only on the system control machine, not on any other machines. If you do accidentally change another machine's copy and it is running a client portion of the Update Server, then the changes you made will be overwritten by the system control machine's copy within a short time (by default, within 5 minutes).
- **Do not edit *CellServDB* directly, even on the system control machine.** Instead, direct the **bos** commands described in this chapter to the system control machine. These commands make the necessary changes automatically.

Cells using the international edition of AFS should not use the Update Server to distribute any files from */usr/afs/etc*, including *CellServDB*. Instead, you must update *CellServDB* on each file server machine individually, using the appropriate **bos** commands (as with the United States edition, you should not edit *CellServDB* directly).

The *AFS Installation Guide* provides instructions for both editions on how to install a new database server machine and verify that it is added to *CellServDB*.

9.2.4. Clients Consult */usr/vice/etc/CellServDB*

Client machines keep a similar file, */usr/vice/etc/CellServDB* on their local disks. This "client" version is not the same as the "server" version discussed here: in addition to the local database server machines, the client version lists database server machines for every cell to which you want to enable access from the machine.

Remember to update */usr/vice/etc/CellServDB* on every client machine whenever you modify the server version. See chapter 15.

9.2.5. Informing Other Cells of Changes to Your *CellServDB*

If you participate in the AFS global name space, you will probably want to inform other cells when you change the identity of your cell's database server machines. As mentioned in section 2.3.2, Transarc Corporation maintains a version of *CellServDB* listing all cells that have agreed to participate in the AFS global name space. The path to this file is */afs/transarc.com/service/etc/CellServDB.export*. Your site contact should inform Transarc Corporation of any changes in your cell's database server set up that would require updates to this file.

You may also wish to maintain a */afs/**cell**/service/etc/CellServDB.local* file in which you make a copy of your local *CellServDB* available to other cells. See section 2.4.3, which discusses the proposed standard which cells should use to advertise information to other cells in the *service/etc* directory in their file tree. The *CellServDB.local* file is a component of that standard.

9.3. Listing the Contents of *CellServDB*

There are two methods for listing the contents of */usr/afs/etc/CellServDB*:

1. Use the **bos listhosts** command, described below.
2. Telnet to the desired machine and **cat** the file. This method displays the Internet host addresses of the database server machines, whereas **bos listhosts** normally reports them by name.

The contents of *CellServDB* will be the same on every file server machine if you used the appropriate **bos** commands when making alterations, and set up the required server and client portions of the Update Server. Nevertheless, both of the methods above allow you to specify a particular machine, which is useful if you suspect an error in that machine's copy.

⇒ To list the database server machines in *CellServDB*

Step 1: At the command shell prompt (%), type

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

where

listh is an acceptable abbreviation of **listhosts**.

machine name specifies the file server machine whose *CellServDB* you wish to check.

cell name is optional, and specifies a complete cell name (such as *transarc.com*). If you provide it, the output displays the *CellServDB* for the specified foreign cell rather than the local cell. Note that you must already know the name of at least one file server machine in the cell, to provide as *machine name*.

For example, issuing this command in the Transarc Corporation cell would produce something like the following:

% **bos listhosts fs1.transarc.com**

```
Cell name is transarc.com
    Host 1 is fs1.transarc.com
    Host 2 is fs7.transarc.com
    Host 3 is fs4.transarc.com
```

The machines are listed in the order they appear in the *CellServDB* on *machine name*. The "Host" numbers shown (here, 1 through 3) are for purposes of listing only. There is no necessary relationship between these numbers and the names or Internet addresses of the machines.

9.4. Adding Database Server Machines to *CellServDB*

Whenever you add a database server machine to your cell, you must add it to each file server machine's copy of */usr/afs/etc/CellServDB,* for the reasons outlined in section 9.2.

If using the United States edition of AFS, make changes to *CellServDB* only on your cell's system control machine. Section 4.4 explains how to determine which machine is the system control machine, if you are unsure. If using the international edition of AFS, make the change on all of your cell's file server and database server machines individually.

When you add a machine to *CellServDB,* you must restart the database server processes (Authentication Server, Backup Server, Protection Server and VL Server) on each database server machine so that they include their new peer process in the election of a new Ubik coordinator. If you do not restart the processes, they will not notice the new machine, with the consequences described in section 9.2.1. Restarting these processes will cause an outage while a new Ubik coordinator is being elected for each.

⇒ To add a database server machine to *CellServDB*

Step 1: Verify that you have the necessary privilege (inclusion in the */usr/afs/etc/UserList*). If necessary, issue the **bos listusers** command, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: If using the United States edition of AFS, use the **bos addhost** command to add each new database server machine to *CellServDB* on the system control machine. (If you have forgotten which machine is the system control machine, see section 4.4.)

If using the international edition of AFS, repeat the **bos addhost** command for each of your cell's file server and database server machines in turn, substituting the name of each for *machine name*.

% **bos addhost -server** <*machine name*>$^+$ **-host** <*host name*>$^+$

where

addh is an acceptable abbreviation of **addhost**.

machine name should name the cell's system control machine, if you are using the United States edition of AFS. If *machine name* is not the system control machine but is running a client portion of the Update Server, then the changes you make will be overwritten the next time the client portion copies the system control machine *CellServDB* (by default, within 5 minutes).

If using the international edition of AFS, this argument should name each of your file server machines in turn.

host name is the full Internet-style name of each new database server machine to be listed in *CellServDB* (example: *fs4.transarc.com*). The BOS Server consults your cell's name service to learn each machine's Internet address and records both the name and address automatically.

Step 3: Restart the Authentication Server, Backup Server, Protection Server, and VL Server on every database server machine, so that the new machine is included in the election of a new Ubik coordinator. If you did not follow the AFS naming recommendations for process names, then substitute your choices for **buserver**, **kaserver**, **ptserver** and **vlserver**.

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

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

where

res is an acceptable abbreviation of **restart**.

machine name specifies a database server machine.

Step 4: Add each machine to the client version of *CellServDB*. See chapter 15.

Step 5: **(Optional.)** If you participate in the AFS global name space, the designated site contact should inform Transarc Corporation of the addition. Include the machine's name and Internet address. Transarc will add the machine to your entry in

/afs/transarc.com/service/etc/CellServDB.export

where it will be visible to other cells.

If you maintain a */afs/***cell***/service/etc/CellServDB.local*, make the same change there.

9.5. Removing Database Server Machines from *CellServDB*

Whenever you remove a database server machine from your cell, you must remove it from the list in */usr/afs/etc/CellServDB*, for the reasons outlined in section 9.2.

If using the United States edition of AFS, make changes to *CellServDB* only on your cell's system control machine. Section 4.4 explains how to determine which machine is a cell's system control machine, if you are unsure. If using the international edition of AFS, make the change on all of your cell's file server and database server machines individually.

When you remove a machine from *CellServDB*, you must restart the database server processes (Authentication Server, Backup Server, Protection Server and VL Server) on each database server machine so they no longer include their former peer process in the election of a new Ubik coordinator. If you do not restart the processes, they will not notice the removal of the machine, with the serious consequences described in section 9.2.1. Restarting these processes will cause an outage while a new Ubik coordinator is being elected for each.

⇒ To remove a database server machine from *CellServDB*

Step 1: Verify that you have the necessary privilege (inclusion in */usr/afs/etc/UserList*). If necessary, issue the **bos listusers** command, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: If using the United States edition of AFS, use the **bos removehost** command to remove the former database server machine(s) from *CellServDB* on the system control machine. (If you have forgotten which machine is the system control machine, see section 4.4.)

If using the international edition of AFS, repeat the **bos removehost** command for each of your cell's file server and database server machines in turn, substituting the name of each for *machine name*.

% **bos removehost** <*machine name*> <*host name*>$^{+}$

where

removeh is an acceptable abbreviation of **removehost**.

machine name should name the cell's system control machine, if you are using the United States edition of AFS. If *machine name* is not the system control machine but is running a client portion of the Update Server, then the changes you make will be overwritten the next time the client portion brings over the system control machine's copy of *CellServDB* (by default, within 5 minutes).

If using the international edition of AFS, this argument should name each of your file server machines in turn.

host name is the full Internet-style name of each former database server machine to be removed from the *CellServDB* (example: *fs2.transarc.com*). If you are not sure of the full name, it is available in the output from **bos listhosts**; see section 9.3.

Step 3: Restart the Authentication Server, Backup Server, Protection Server and VL Server on every database server machine, so that the former machine is not included in the election of a new Ubik coordinator. If you did not follow AFS recommendations for process names, then substitute your choices for **buserver**, **kaserver**, **ptserver**, and **vlserver**.

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

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

where

res is an acceptable abbreviation of **restart**.

machine name names a database server machine.

Step 4: Remove each machine from the client version of *CellServDB*. See chapter 15.

Step 5: **(Optional.)** If you participate in the AFS global name space, send mail to **afshelp@transarc.com** to inform AFS Product Support at Transarc of the removal. Include the machine's name and Internet address. Transarc will make remove the machine from your entry in

/afs/transarc.com/service/etc/CellServDB.export

where it will be visible to other cells.

If you maintain a */afs/**cell**/service/etc/CellServDB.local*, make the same change there.

10. Managing Server Encryption Keys

This chapter explains how to maintain your cell's ***server encryption keys***, which are vital to secure communication in AFS.

It has the following sections:

10.1 Summary of Commands 10-1
10.2 About Server Encryption Keys 10-2
10.3 Listing Server Encryption Keys 10-6
10.4 Adding Server Encryption Keys 10-9
10.5 Removing Server Encryption Keys 10-12
10.6 Handling Server Encryption Key Emergencies 10-14

10.1. Summary of 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 *KeyFile* . . . . **bos listkeys**
Remove an old server encryption key . . . . **bos removekey**

10.2. 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 provide security for information being transferred between AFS server processes and their clients. Server encryption keys may be thought of as a server process' password; like user passwords, they are stored in the Authentication Database.

Maintaining your cell's server encryption keys properly is the most important part of protecting the information in your AFS file tree. The next section explains how these keys protect data transferred between clients and servers.

10.2.1. The Role of Server Encryption Keys in Mutual Authentication: A Review

Server encryption keys play a central role in the mutual authentication between client and server processes in AFS. See section 1.5 for a more complete explanation of mutual authentication.

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 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 indicates 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 makes the client believe that the server is legitimate.

10.2.2. Maintaining AFS Server Encryption Keys

There are several points to keep in mind as you manage your cell's server encryption keys.

10.2.2.1. The AFS Servers Can Only Use One Server Encryption Key at a Time

The AFS servers can only use one server encryption key at a time (that is, the version of the server encryption key in the *afs* Authentication Database entry). When a token is issued to a user, it is sealed with that version of the server encryption key.

10.2.2.2. You Must Maintain Older Keys for a Specified Amount of Time

Any time you change the AFS server encryption key, it is probable that some clients have unexpired tickets sealed with the previous key. The more frequently you change the server encryption key, the more such tickets there are likely to be.

If a token is to remain valid throughout its lifetime, the AFS servers must have continued access to the server encryption key used to seal that token and the version number associated with that key (that is, the ***key version number*** or ***kvno***). Therefore, you must maintain a list of server encryption keys, rather than storing only the one key currently in use. This list must be maintained in the file */usr/afs/etc/KeyFile* on the local disk of every file server machine in the cell.

You should delete an old server encryption key only when a new server encryption key has been available for an amount of time equal to the maximum token lifetime in your cell. By default, the maximum token lifetime is 25 hours (except for users whose Authentication Database entries were created under AFS 3.0, for whom the default is 100 hours); this default can be changed using the **kas setfields** command.

10.2.2.3. You Change Server Encryption Keys Just Like Passwords

You invent a new AFS server encryption key in much the same way regular users change their passwords, by providing a character string. The string is converted into an encryption key automatically. See section 10.4.

10.2.2.4. The Server Encryption Keys are Stored in Two Places

There are two separate storage sites for the AFS server encryption key. You must make changes in both places whenever you change the key, and make them in the correct order (as specified in section 10.4). If you make changes in one place but not the other or make changes in the wrong order, clients and servers may become unable to communicate, potentially shutting down your entire cell.

The two storage sites for keys are

- The file */usr/afs/etc/KeyFile* on the local disk of every file server machine in the cell. The AFS servers consult *KeyFile* to learn which encryption key they should use to decrypt the tickets that clients present to them. *KeyFile* records a list of AFS server encryption keys, each marked with a different ***key version number***.

- The *afs* entry in the Authentication Database. This entry is similar to an entry for an individual user—the scrambled "password" in the key field is the AFS server encryption key. The Ticket Granting Service uses this key to encrypt the server tickets it gives to clients. The *afs* entry in the Authentication Database only contains one key at a time, rather than a list, because the TGS never needs to read old tickets—it only needs to make new tickets using the newest key.

Note: The *AFS Installation Guide* contains instructions on installing your cell's first file server machine and creating the initial *afs* entry and *KeyFile*.

10.2.2.5. All File Server Machines Must Have the Same *KeyFile*

All file server machines in your cell must have the same version of *KeyFile*. In cells using the United States edition of AFS, the easiest way to guarantee consistency is to use the Update Server to distribute the system control machine's copy of *KeyFile* to all of the other file server and database server machines. This implies that administrators in cells that use the United States edition should

- Change only the system control machine's *KeyFile* and only using the appropriate **bos** commands (see sections 10.4 and 10.5). Never edit *KeyFile* directly with a text editor, and do not change it anywhere other than the system control machine, except when dealing with the emergencies discussed in section 10.6. Any changes you make on non-system control machines will be overwritten the next time the machine's Update Server client portion copies the system control machine's copy of the *KeyFile*. By default, this happens within five minutes.
- Run an Update Server server portion on the system control machine and an Update Server client portion on every other file server machine to point to the system control machine. The *AFS Installation Guide* explains how to install both portions of the Update Server on the proper machines. Section 4.3 explains how to check if Update Server server and client portions are running.

Cells running the international edition of AFS should not use the Update Server to distribute the contents of */usr/afs/etc*, particularly the *KeyFile*. The data in the *KeyFile* is too sensitive to cross the network in unencrypted form, and the international edition of AFS does not include the necessary encryption routines in a form that the Update Server can use. (United States government regulations forbid export of these routines in certain forms to countries other than Canada).

Administrators in cells outside the United States and Canada should change the *KeyFile* on every file server and database server machine individually, using the appropriate **bos** commands (see sections 10.4 and 10.5). Never edit *KeyFile* directly with a text editor. Because you are typing the **bos** commands multiple times, you must take extra care to type exactly the same key on each and every file server machine.

10.2.2.6. You Should Remove "Old" Keys from *KeyFile*

You may want to remove old server encryption keys from *KeyFile* periodically, to keep the file from being cluttered. It is safe to remove old keys from *KeyFile* if a new server encryption key has been in the Authentication Database for at least as much time as it takes for a token with the maximum lifetime to expire. See section 10.5 to learn how to remove keys from *KeyFile*.

10.2.2.7. You Should Change the AFS Server Encryption Key Frequently

You can improve system security by changing the AFS server encryption key frequently. Most cells should change the key at least once a month; changing it more often enhances security.

10.3. Listing Server Encryption Keys

You can list the server encryption keys in */usr/afs/etc/KeyFile* on any file server machine using **bos listkeys**. You can examine the key in the Authentication Database's *afs* entry using **kas examine**.

There are two modes in which to examine the keys. If you want to see the actual string of octal digits that make up the key, you must first use **bos setauth** to turn off authorization checking on the file server machine to which you direct the **bos listkeys** or **kas examine** command. The reasoning behind this requirement is two-fold. First, it implies that only someone authorized to issue the **bos setauth** command (presumably an administrator) is able to view keys from the Authentication Database. Second, it makes it evident that the system is in a compromised state of security while keys are being examined (both turning off authorization checking and displaying actual keys on a screen are inherent security risks). Because you turn off authorization checking for such a short time, you do not need to restart the server processes on the machine in this case. In other situations, such as fixing a server encryption key emergency, it is important to restart processes after changing authorization checking requirements. See section 10.6.

The other mode for inspecting keys is both simpler and safer, because it does not involve disabling authorization checking. You use the same **bos listkeys** and **kas examine** commands, but instead of seeing the actual octal digits, you see a ***checksum***, which is a decimal number derived by encrypting a constant with the key.

In most cases, seeing a checksum rather than the actual key is adequate. The checksums you see for a given key are the same with both **bos listkeys** and **kas examine**, so comparing them reveals whether the keys match just as well as comparing the actual keys. Seeing checksums is adequate when examining keys for all the following common reasons:

- When choosing the key version number for a new key, to learn which key version numbers are already in use. Here the key itself is irrelevant.
- To verify that the keys with the same key version number are the same in both the *afs* entry and *KeyFile*. As mentioned, the checksums match if the keys are the same. If you suspect that they are different, you are probably experiencing authentication problems throughout your cell. The easiest solution in this case is to replace the keys in both places; then you can be certain the new keys match. See section 10.4 or 10.6.
- Before removing old keys from *KeyFile*, a previous key needs to have been in *KeyFile* for as long as the maximum lifetime you allow the Authentication Server to grant on a token (the default is 25 hours; for users created under AFS 3.0, 100 hours). Removing old keys prevents *KeyFile* from growing excessively large. See section 10.5 to learn how to remove keys.

The following instructions use the method in which you do not turn off authorization checking and so see checksums rather than actual keys. In order to see actual keys, first use the **off** option on **bos setauth** to disable authorization checking on the file server machine to which you will direct the examination command (for **kas examine**, this must be a database server machine). Then issue the **bos listkeys** and/or **kas examine** command, adding the **-noauth** flag to prevent mutual authentication attempts; with **kas examine**, also use the **-servers** argument to direct the command to the database server machine on which you disabled authorization checking. When finished, remember to use **bos setauth** again to turn authorization checking back on.

⇒ To list the contents of *KeyFile*

Step 1: Verify that you have the privilege necessary to look at the contents of *KeyFile* (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: List the contents of one machine's *Keyfile*.

% **bos listkeys** <*machine name*>

where

listk is an acceptable abbreviation of **listkeys**.

machine name specifies a file server machine. Normally, any machine is acceptable, because *KeyFile* should be the same on all of them. Being able to specify a certain machine is handy if you suspect that its *KeyFile* disagrees with the system control machine's.

Since you have not disabled authorization checking, the output from **bos listkeys** specifies the key version number and the checksum for each valid AFS server encryption key, and the time at which *KeyFile* last changed. The line "`All done.`" indicates the end of the output.

Following is an example output:

```
key 0 has cksum 972037177
key 1 has cksum 2825165022
Keys last changed on Fri Oct 1 11:20:29 1993.
All done.
```

⇒ To see the key in the Authentication Database's *afs* entry

Step 1: Verify that you have the privilege necessary to examine any Authentication Database entry (the `ADMIN` flag in your Database entry). If necessary, issue **kas examine** on your own entry.

% **kas examine** <*name of user*>
Password for *name of user***:** <*your password*>

Step 2: Inspect the *afs* key.

% **kas examine afs**

where

e is an acceptable abbreviation of **examine**.

The output should be similar to the following.

```
User data for afs
 key (1) cksum is 2825165022, last cpw: no date
 entry expires on never. Max ticket lifetime 100.00 hours.
 last mod on Fri Oct 1 11:23:36 1993 by admin
```

The second line shows the key version number in parentheses and the key's checksum. The last line reports when the *afs* key was last changed and by whom. There is no necessary relationship between the date shown in this entry and the date reported from **bos listkeys** (as shown in the example above). However, if the administrator who last changed the keys followed the instructions in section 10.4 in an uninterrupted pass, then the time stamps should be fairly close to one another. The *AFS Command Reference Manual* entry for **kas examine** explains the other fields in this output.

10.4. Adding Server Encryption Keys

Section 10.2 explained that there are two separate storage sites for the AFS server encryption key: the file */usr/afs/etc/KeyFile* (consulted by server processes) and the Authentication Database *afs* entry (consulted by the TGS). To ensure that server processes and the TGS always share the same AFS server encryption key, execute all the steps without interruption.

⇒ To add a new server key

Step 1: Verify that you have the privilege necessary to alter *KeyFile* (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that you have the privilege necessary to alter entries in the Authentication Database (the `ADMIN` flag appears in your own Authentication Database entry). If necessary, issue **kas examine**, which is fully described in section 20.4.

% **kas examine** <*name of user*>
Password for *name of user***:** <*your password*>

where **e** is an acceptable abbreviation of **examine**.

Step 3: Learn which key version numbers are already in use in *KeyFile*, as a first step in choosing key version number for the new key.

% **bos listkeys** <*machine name*>

where

listk is an acceptable abbreviation of **listkeys**.

machine name names any file server machine.

Step 4: Choose a key version number for the new key, based on what you learned in step 3, and the following constraints:

- Key version numbers must be integers between *0* 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 *KeyFile*, particularly if it is the one also currently in the Authentication Database. If you do, the new key overwrites the key that currently has that number in *KeyFile*. Some clients may still have tickets sealed with the current key, but the server processes use the new key to decrypt tickets marked with that key version number. Believing the ticket to be expired or forged, the server process refuses to communicate with the client.

Step 5: If your cell runs the United States edition of AFS, create a new AFS server encryption key in *KeyFile* on the system control machine. (If you have forgotten which machine is the system control machine, see section 4.4.)

If your cell runs the international edition of AFS, repeat the **bos addkey** command for each file server and database server machine in your cell, specifying each as *machine name* in turn.

The following instruction includes command-line switches to eliminate any possible confusion about which argument is the key and which is the key version number.

% **bos addkey -s** <*machine name*> **-key** <*key*> **-kvno** <*key version number*>

where

addk is an acceptable abbreviation of **addkeys**.

machine name names the cell's system control machine if you are using the United States edition of AFS.

If you are using the international edition of AFS, *machine name* specifies the name of each file server machine in turn.

key is a character string like a password, not a string of octal numbers. It may be of any length greater than 0 and less than about 1000 characters; the BOS Server uses a scrambling algorithm to convert it into an octal string. A longer string is better because it makes the password harder to break.

Important: Remember the string. You will use it again in step 7. If you are using the international edition of AFS, be sure to type the same string each time you issue this command.

key version number
is the desired key version number (a decimal integer). Legal values are integers between *0* and *255*. Do not use *999* or a number currently used in *KeyFile,* for the reasons outlined in step 4.

Important: Remember the number. You will use it again in step 7. If you are using the international edition of AFS, be sure to type the same number each time you issue this command.

Step 6: If you are using the United States edition of AFS, wait for a few minutes to ensure that the Update Server client portion running on each file server machine has a chance to copy the new *KeyFile* from the system control machine. The maximum amount of time you must wait is equivalent to whatever time is specified on the command parameter line of the *upclientetc* instance; the default time is five minutes. (If necessary, you can check the *upclientetc* instance's command parameter line by using the **-long** flag on **bos status**; see section 4.3.)

If you are using the international edition of AFS, try to complete step 5 within the maximum time period it takes the Update Server to distribute the *KeyFile* in the United States edition (by default, five minutes).

Do not continue until all file server machines have the same version of *KeyFile*. If the system control machine is running the server portion of the Update Server and the other machines are running the client portion, then waiting the specified time suffices. If you wish to be certain that all machines have the same *KeyFile,* issue **bos listkeys** (described fully in section 10.3) for every file server machine and compare the output with that for the system control machine.

% **bos listkeys** <*machine name*>

Step 7: Put the same key in the Authentication Database *afs* entry.

% **kas setpassword -name afs -kvno** <*kvno*>
`Password for` *name of user*: <*your password*>
`new_password:` <*afs password*>
`Verifying, please re-enter new_password:` <*afs password*>

where

sp	is an acceptable abbreviation of **setpassword**
kvno	is the same key version number you specified in step 5.
your password	is your password.
afs password	is the same character string you specified as *key* in step 5. It will not echo to the screen.

Step 8: **(Optional.)** If you want to verify that the keys you just created in *KeyFile* and in the *afs* entry are identical and have the same key version number, follow the procedures for listing keys described in section 10.3.

Step 9: **(Optional.)** After waiting 25 hours (or however long server tickets and tokens last in your cell), remove the former server encryption key from *KeyFile*. This will prevent *KeyFile* from becoming cluttered. Use **bos removekey**, which section 10.5 describes fully.

% **bos removekey** <*machine name*> <*key version number*>+

10.5. Removing Server Encryption Keys

As mentioned in section 10.2, keys must remain in */usr/afs/etc/KeyFile* no longer than the longest token lifetime in your cell (the default is 25 hours for users whose Authentication Database entries were created under AFS 3.1 or 3.2, 100 hours for those whose entries were created under AFS 3.0). Removing old keys from *KeyFile* keeps it to a reasonable size.

There is no command for removing the key from the *afs* entry in the Authentication Database, because that field should never be empty. You use **kas setpassword** to replace the *afs* key, but you should not do so without following all of the steps outlined in section 10.4 for a normal key change.

Do not remove from *KeyFile* the key that is currently in the *afs* field. Doing so will make the AFS server processes unable to decrypt the tickets that clients present to them.

⇒ To remove key(s) from *KeyFile*

Step 1: Verify that you have the privilege necessary to alter *KeyFile* (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

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

Step 2: Verify that it has been at least 25 (100) hours since a new key was placed in *KeyFile*. If necessary, use **bos listkeys** (described fully in section 10.3) to check the date the key last changed.

`%` **bos listkeys** <*machine name*>

The line beginning "`Keys last changed`" tells you the needed date.

Step 3: Verify that each key you wish to remove does not have the same key version number as the key currently in the Authentication Database *afs* entry. If necessary, use **kas examine** (fully described in section 10.3 to learn the key version number of the current key.

`%` **kas examine afs**
`Password for` *name of user*`:` <*your password*>

The second line of the output tells you the key version number.

Step 4: If your cell runs the United States edition of AFS, remove each old key from *KeyFile* on the system control machine. (If you have forgotten which machine is the system control machine, see section 4.4.)

If your cell runs the international edition of AFS, repeat the **bos removekey** command for each file server and database server machine in your cell, specifying each as *machine name* in turn.

`%` **bos removekey** <*machine name*> <*key version number*>$^+$

where

removek	is and acceptable abbreviation of **removekey**.
machine name	names the cell's system control machine if you are using the United States edition of AFS.
	If you are using the international edition of AFS, *machine name* specifies the name of each file server machine in turn.
key version number	specifies the key version number of each key you wish to remove from *KeyFile*.

10.6. Handling Server Encryption Key Emergencies

Under 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 being 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 process' machine does not include the key that the Ticket Granting Service (TGS) is using to encrypt tickets (the key from the *afs* Authentication Database entry).

Another possibility is that the *KeyFile*s on different machines do not contain the same keys. In this case, communication among server processes themselves becomes 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 using different server encryption keys.

One symptom of server encryption key mismatch is receiving the following error message when you direct a **bos** command to a file server machine in the local cell. (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* on each file server machine, so that the TGS and all server processes again share the same key.

Handling key emergencies is not difficult, but it does require some unusual actions. Sections 10.6.1 through 10.6.6 explain the reasons for these actions; the actual procedures appear in the subsequent instructions.

10.6.1. Preventing 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 probably cannot understand any ticket you might present them. When you do not mutually authenticate, the server processes assign you the identity *anonymous*. To prevent mutual authentication, the following instructions have you discard any tokens you may already have (using the **unlog** command) and use the **-noauth** flag where available.

10.6.2. Disabling Authorization Checking by Hand

Because the server processes consider you *anonymous* when you do not mutually authenticate, you must turn off authorization checking. Only with authorization checking disabled do the server processes allow *anonymous* to perform privileged actions such as key creation.

In an emergency, you disable authorization checking by creating the file */usr/afs/local/NoAuth* by hand. In normal circumstances, you should use the **bos setauth** command instead.

10.6.3. Restarting Processes

As soon as you change authorization checking requirements (either enabling or disabling them), the server processes obey the requirements in making connections with clients, but not in existing connections they may have among themselves. The File Server, for example, maintains an open connection with the Protection Server, so that it can request current protection subgroups (CPSs) for new users without having to establish a new connection each time (for an explanation of CPSs, see section 18.2.5.1). The instances of a database server process on different machines have connections open to one another as well.

When correcting a server encryption key mismatch, it is important that the server processes create new connections that obey the changed authorization requirements. You must restart the processes to force them to open new connections. It is simplest to restart all the processes on the machine, except the BOS Server, which does not maintain connections with any other processes. Restarting causes a service outage, which only lasts a few minutes for most processes. The length of the *fs* process' outage may be longer, depending on how many volumes reside on the machine (it must bring the volumes online one by one).

10.6.4. Working Quickly on Each Machine

Disabling authorization checking is a severe security breach, because it means that all server processes on that machine perform any action for anyone. You should disable authorization checking only for as long as necessary, completing all steps in an uninterrupted session and as quickly as possible.

10.6.5. Working at the Console

Because disabling authorization checking is dangerous, you may wish to work at the console of each file server machine rather than telnetting to it. This is not a requirement, but it does ensure that no one else logs onto the console while you are working there. It does not prevent others from telnetting to the machine or doing a remote login, 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. As recommended in section 2.12.4, you can improve security in general by limiting the number of people who can telnet to your file server machines at any time.

10.6.6. Changing Individual *KeyFile*s

If your cell uses the United States edition of AFS, an emergency is the only time you should change the *KeyFile* on any machine other than the system control machine. In a key emergency, it is necessary to treat each machine individually because mismatched keys mean that the server portion of the Update Server (*upserver* process) running on the system control machine may not be able to mutually authenticate with the client portions of the Update Server (*upclientetc* processes) on all the other file server machines.

If they cannot mutually authenticate, then the *upserver* refuses to release the new *KeyFile* to the *upclientetc* processes. Changing individual *KeyFile*s in this case is the only possibility, because the normal distribution mechanism is not working.

If the *upserver* and *upclient*s can mutually authenticate, then the normal distribution mechanism does work: the *upserver* distributes the system control machine's copy of *KeyFile* to overwrite the individual copies maintained by the *upclientetc*s. However, you must change the individuals anyway, since you cannot predict if the distribution is working correctly. To avoid creating inconsistency, the following instructions have you change the copy of *KeyFile* on the system control machine first. That way, its *upserver* process is distributing the same new key that you are placing on each file server machine by hand.

If your cell uses the international edition of AFS, you standardly change keys on all of your file server and database server machines individually. Because you are not using the Update Server to distribute the *KeyFile*, you need not worry about the inability of the server and client portions to mutually authenticate. Nevertheless, the following instructions are also appropriate for you when dealing with key emergencies.

10.6.7. Two Component Procedures

There are two subprocedures used frequently in the following instructions: disabling authorization checking and re-enabling it. For the sake of clarity, the procedures are detailed here; the instructions refer to them as necessary.

10.6.7.1. Disabling Authorization Checking in an Emergency

Substep a: Log onto the console of the file server machine (or telnet) as "root."

Substep b: Turn off authorization checking by creating the file */usr/afs/local/NoAuth*.

touch /usr/afs/local/NoAuth

Substep c: Discard any tokens you may have. They might have been created with an incompatible key, which can prevent some commands from executing.

unlog

Substep d: Stop and restart all the processes on the file server machine, except the BOS Server, which does not maintain connections with other processes. This causes an outage, as described in section 10.6.3.

bos restart <*machine name*> **-noauth**

where

res is an acceptable abbreviation of **restart**.

machine name specifies the file server machine on which to restart processes.

-noauth means the BOS Server assigns you the identity *anonymous* rather than trying to authenticate you. You may shorten it to **-n**.

10.6.7.2. Re-enabling Authorization Checking in an Emergency

Substep a: Obtain a token under a system administrator identity (member of *system:administrators* and */usr/afs/etc/UserList*).

klog <*administrator username*>
`Password:` <*administrator password*>

Substep b: Remove the *NoAuth* file from the file server machine's */usr/afs/local* directory.

Substep c: Stop and restart all the processes on the file server machine, except the BOS Server, which does not maintain connections with other processes. This causes an outage, as described in section 10.6.3.

bos restart <*machine name*>

where

res is an acceptable abbreviation of **restart**.

machine name specifies the file server machine on which to restart processes.

Substep d: If appropriate, log out from the console of the machine (or close the telnet connection), after issuing **unlog** to destroy your tokens.

⇒ To create a new server encryption key in emergencies

Step 1: **On the system control machine**, perform the procedure for disabling authorization checking described in section 10.6.7.1.

Step 2: Find out which key version numbers are already in use in *KeyFile*, as a first step in choosing the new key's key version number. You may wish to direct this command to several file server machines, to see if they are all aware of the same keys.

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

where

listk is an acceptable abbreviation of **listkeys**.

machine name specifies a file server machine.

-noauth means the BOS Server assigns you the identity *anonymous* rather than trying to authenticate you.

Step 3: Choose a key version number for the new key, based on what you learned in step 2 plus the following constraints:

- 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 *KeyFile*. If you do, the new key overwrites the key that currently has that number in *KeyFile*. Some clients may still have tickets sealed with the current key, but the server processes use the new key to decrypt tickets marked with that key version number. Believing the ticket to be expired or forged, the server process refuses to communicate with the client.

Step 4: **On the system control machine**, create a new AFS server encryption key in *KeyFile*. (The following instruction includes command-line switches to eliminate any possible confusion about which argument is the key and which is the key version number.)

\# **bos addkey -s** <*machine name*> **-key** <*key*> **-kvno** <*key version number*> **-noauth**

where

addk is an acceptable abbreviation of **addkey**.

machine name names the file server machine whose *KeyFile* you are altering.

key is a character string like a password, not a string of octal numbers. It may be of any length greater than 0 and less than about 1000 characters; the BOS Server uses a scrambling algorithm to convert it into an octal string. A longer string is better because it makes the password harder to break.

Important: Remember the string. You will use it again in steps 7, 8, and 13.

key version number is the key version number you chose in step 3. Legal values are integers between *0* and *255*. Do not use *999* or a number currently used in *KeyFile*.

Important: Remember the number. You will use it again in steps 7, 8, and 13.

-noauth means the BOS Server assigns you the identity *anonymous* rather than trying to authenticate you.

Step 5: **On every database server machine in your cell** (other than the system control machine), perform the procedure for disabling authorization checking described in section 10.6.7.1. 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. (If you need to learn which machines are database server machines, use the **bos listhosts** command as described in section 4.4.)

Step 6: Wait at least 90 seconds after finishing step 5, to allow each of the database server processes (Authentication, Backup, Protection and Volume Location Server) to finish electing a new sync site. Then, verify that the election worked properly, so that communication between the sites is possible. To check on the Protection Server, issue the following command once for each database server machine (including the system control machine if it is a database server machine).

udebug <*machine name*> **7002**

The output for all of the database server machines should agree on which one is the sync site for the Protection Server. For only one of the machines, there should be a line of output that begins

```
I am sync site ...
```

For the other machines, there should be a line that reads

```
I am not sync site
```

For these machines, a subsequent line of output names the "`Sync host`". The IP address shown should belong to the machine claiming to be the sync site.

If the output does not meet these requirements or seems abnormal in another way, contact your AFS Product Support Representative for assistance.

Now, repeat for the Volume Location Server and the Authentication Server, interpreting the output as for the Protection Server:

udebug <*machine name*> **7003**
udebug <*machine name*> **7004**

Note that while all instances of a given process must agree on the sync site, it is acceptable that each of the three different server processes use a different sync site.

Step 7: **On every database server machine in your cell** (other than the system control machine), issue the **bos addkey** command described in step 4. Be sure to use the same values for *key* and *kvno* as you used in step 4.

Step 8: Place the new key (the same one you specified in steps 4 and 7) in the Authentication Database's *afs* entry.

kas setpassword -name afs -kvno <*kvno*> **-noauth**
`new_password:` <*afs password*>
`Verifying, please re-enter new_password:` <*afs password*>

where

sp is an acceptable alias for **setpassword**.

kvno is the same key version number you specified in step 4.

afs password is the same character string you specified as *key* in step 4. It does not echo visibly.

Step 9: **On every database server machine in your cell**, including the system control machine if it is a database server machine, perform the procedure for re-enabling authorization checking described in section 10.6.7.2. If the system control machine is not a database server machine, do not perform this procedure on it yet.

Step 10: Repeat step 6 to verify that each database server process has properly elected a sync site after being restarted in step 9.

Step 11: **On the system control machine, if it is not a database server machine**, perform the procedure for re-enabling authorization checking described in section 10.6.7.2. If it is a database server machine, you already performed this procedure in step 9.

Step 12: **On all remaining (simple) file server machines**, perform the procedure for disabling authorization checking described in section 10.6.7.1.

Step 13: **On all remaining (simple) file server machines**, issue the **bos addkey** command described in step 4. Be sure to use the same values for *key* and *kvno* as you used in step 4.

Step 14: **On all remaining (simple) file server machines**, perform the procedure for re-enabling authorization checking described in section 10.6.7.2.

11. Installing New Server Process Software and Disks

This chapter 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 on a file server machine to house AFS volumes. It has the following sections:

11.1	Summary of Commands	11-1
11.2	About Installing New Server Process Binaries	11-2
11.3	Installing New Binaries	11-3
11.4	Determining the Automatic Restart Time for New Binaries	11-7
11.5	Reverting to an Old Version of Binaries	11-9
11.6	Checking Binary Version Dates	11-11
11.7	Removing Obsolete .BAK and .OLD Copies	11-12
11.8	Adding and Mounting a New Disk and/or Partition	11-13
11.9	Removing and Unmounting a Disk and/or Partition	11-16

11.1. Summary of 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 ReadWrite volumes	**vos move**

11.2. About Installing New Server Process Binaries

The most frequent reason to replace a server process' binaries is in response to an update from Transarc Corporation. In general, installation instructions will accompany an update, but this chapter provides an additional reference.

Each file server machine stores its own copy of all binaries for the AFS server processes in the */usr/afs/bin* directory on its local disk. All machines should run the same version of AFS software. Transarc Corporation will supply you with binaries for each supported CPU/operating system type you use as a file server machine. To be certain that every file server machine is using the same version of software, you install new binaries on a ***binary distribution machine*** of each CPU/operating system type.

This arrangement has several implications for you:

- Each binary distribution machine must run the server portion of the Update Server. All the other file server machines in the cell should be running a client portion of the Update Server that points to the binary distribution machine of their type. The *AFS Installation Guide* explains how to install server and client portions of the Update Server on the appropriate machines.
- You should install new binaries on binary distribution machines only; do not install new binaries on all file server machines. If you do accidentally install binaries on a non-distribution machine and it is running a client portion of the Update Server, then the binaries you install will be overwritten by the binary distribution machine's binaries within a short time (by default, within 5 minutes).
- **You should use the bos command described in this chapter to install binaries**, rather than doing so directly. Direct the command to the binary distribution machines.
- You should compile and install new software on every binary distribution machine in your cell. Do not run different versions of software on different machine types.

The *AFS Installation Guide* provides instructions on how to install binary distribution machines.

11.3. Installing New Binaries

As mentioned in the introduction, you usually install new binaries only when you get update releases from Transarc Corporation. The releases will include installation instructions, but this section is provided as a reference. Remember that an AFS server process will behave predictably only if the same version of it is running on every file server machine in your cell. Do not run a new version of software on only a subset of your file server machines. Transarc Corporation will supply you with binaries for each file server machine type; install the different binaries on the appropriate binary distribution machines.

11.3.1. How and When a Process Starts Using New Binaries

Using new binaries for a server process requires stopping the current instance of the process and starting a new one. This causes a service outage. By default, the BOS Server does not switch a server process to using new binaries immediately. Instead, it restarts processes with new binaries at a certain time each day when system usage is likely to be low (so that the outage affects fewer people). Section 11.4 discusses this daily restart in more detail.

You can force a file server machine to start using new server process binaries immediately by issuing **bos restart**. You can also set the automatic restart time using **bos setrestart**. See section 11.4.

No restart is necessary when you install new binaries for a command suite (such as **bos**) on a file server machine. The next time anyone issues a command that invokes a command, the new binaries will be used automatically. You will probably want to install the new command binaries on client machines, too.

11.3.2. Automatic Saving of the Current Version

In rare instances, a new version of binaries may not work properly, making it necessary to revert to a previous version. When you install new binaries, the BOS Server automatically saves the current binary file by putting a *.BAK* extension on it. The current *.BAK* version, if any, is renamed with a *.OLD* extension, if there is no *.OLD* version already. If there is a current *.OLD* version, then the current *.BAK* must be at least seven days old in order to replace it.

11.3.3. Installation into Alternate Directories

The commands described here allow you to install and distribute binaries into other directories on a file server machine's local disk (besides */usr/afs/bin*). This is useful for non-AFS binaries; AFS binaries should be in */usr/afs/bin* because that is the only directory in which the BOS Server automatically checks for new binaries.

To learn about the **-dir** argument that allows installation into alternate directories, see the *AFS Command Reference Manual*.

⇒ To install new binaries into */usr/afs/bin*

Step 1: Verify that you have the necessary privilege (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that the binaries are available in the directory from which you will install them into */usr/afs/bin*. (If you do not yet have AFS running, this is probably a directory on the machine's hard disk. Once AFS is running, you might maintain a directory in AFS for each system type you use, in which you keep "safe" copies of the binaries that Transarc Corporation sends you.) If necessary, place the new binaries in that directory.

Step 3: Install the binaries into */usr/afs/bin* on a binary distribution machine. (If you have forgotten which machine is the binary distribution machine for a particular CPU/operating system type, see section 4.4.)

% **bos install** <*machine name*> <*files to install*>$^+$

where

i is an acceptable abbreviation of **install**.

machine name specifies a binary distribution machine. If you accidentally specify a non-binary distribution machine and it is running a client portion of the Update Server, then the binaries will be overwritten the next time the client portion receives binaries from its binary distribution machine. By default, this will happen within 5 minutes.

files to install names the binary file that is currently residing in another directory awaiting installation into */usr/afs/bin*. (You might maintain a directory in AFS for each system type you use, in which you keep "safe" copies of the binaries that Transarc Corporation sends you.) Pathnames are interpreted relative to the working directory. The last element in each pathname (the filename itself) should match the name of the file it is replacing, such as *bosserver* or *volserver* for server processes, *bos* or *vos* for commands.

There is a single binary file for each AFS server process except the "fs" process. That process has three components--File Server, Volume Server and Salvager--in three separate files. Installing a new version of one component does not necessarily mean that you need to replace all three.

Step 4: Repeat step 3 for each binary distribution machine in your cell. (If you have forgotten which machines are the binary distribution machines for the different CPU/operating system types you have, see section 4.4.)

Step 5: **(Optional.)** If you want the file server machines to use new server process binaries immediately, restart the relevant process(es) by following the next set of instructions. Note that a service outage will result. (For database server processes, the outage lasts approximately two minutes; for the *fs* process, possibly from 15 to 30 minutes.) No restarting is necessary if installing a new version of the binaries for a command suite.

⇒ To restart server processes to use new binaries immediately

Step 1: Verify that you have the necessary privilege (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Verify that the new binaries are installed in */usr/afs/bin*. If necessary, issue **bos getdate** to check the modification time stamp on the binary file. Section 11.6 describes this command fully.

% **bos getdate** <*machine name*> <*files to check*>$^+$

Step 3: If you are working on an AFS client machine, verify that the binaries for the **bos** command suite are on the local disk. This precautionary step proves helpful if something goes wrong when you issue the **bos restart** command in step 5 below, and you have to reissue it to fix the problem.

Note the following explanation. In all likelihood, you have previously installed the **bos** (and other command suite) binaries into a directory that points into AFS, rather than onto the local disk itself. For instance, you may have created a directory called */usr/afsws* on each client machine to contain AFS-specific subdirectories and files, and then symbolically linked it to an AFS directory containing the version of those files appropriate to the client machine's CPU/operating system type. Using a symbolic link in this way is recommended, as it saves space on the local disk. For the following steps, however, the **bos** binaries must be on the local disk because restarting certain processes (particularly the database server processes) can make the file system inaccessible.

For example, client machines in the Transarc Corporation cell have a subdirectory */usr/afsws/bin* on their local disk to contain the **bos** (and other AFS command suite) binaries. The */usr/afsws* directory is symbolically linked to */afs/transarc.com/**cpu-type***. When a **bos** command is issued on a client, the binaries in */afs/transarc.com/**cpu-type**/bin* are used.

If the **bos** binaries are not available on the local disk, the easiest action is to copy the binaries into */tmp*:

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

(If you are working at the console of a file server machine, you need do nothing; the binaries are already in */usr/afs/bin*).

Step 4: Verify that the version of binaries that you want the machine to use are in its */usr/afs/bin* directory. All you need to do is wait a few minutes after issuing a **bos install** or **bos uninstall** command on this file server machine's associated binary distribution machine; wait long enough for the file server machine's Update Server client portion to bring the binaries over. The maximum amount of time you need to wait is equivalent to whatever time is specified on the command parameter line of the *upclientbin* instance; the default time is 5 minutes.

If you wish to be certain that the right binaries are in place, verify that the time stamp on the binary file is the same as on this file server machine's binary distribution machine. Issue the **bos getdate** command (described completely in section 11.6) on both machines.

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

The output for each file you check tells you the compilation dates for the current, *.BAK* and *.OLD* versions of the binaries. The current version's date should match for the two machines.

Step 5: Restart the server process that has new binaries.

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

where

res is an acceptable abbreviation of **restart**.

machine name specifies the file server machine on which you wish to restart the specified process(es).

instances names each of the processes you want to stop and immediately restart to use new binaries. The standard values are *kaserver*, *ptserver*, *runntp*, *upclientbin*, *upclientetc*, *upserver*, *vlserver* or *fs* (if you installed a new *fileserver* or *volserver*; restarting *fs* is not necessary after installation of a new *salvager*).

If the process does not start up correctly or another error occurs, issue **bos restart** again, this time using the */tmp* version of **bos** that you created during step 3. If repeating **bos restart** does not solve the problem, you may need to reboot the file server machine, either at the console or remotely with **/tmp/bos exec**. Rebooting a file server machine may cause a long outage. Instructions for rebooting a file server appear in section 4.11.

Step 6: Repeat steps 4 and 5 on every file server machine in the cell, so that they all start using the new binaries.

11.4. Determining the Automatic Restart Time for New Binaries

File server machines do not begin using binaries immediately after installation. Instead, they continue to use the current binaries until the BOS Server performs its automatic daily check through */usr/afs/bin* for new binaries. Each day at the appointed time, the BOS Server compares the time stamp on each binary file (which records when the file was modified) with the time when the associated process last restarted (this time stamp is referred to in the output from the **bos status** command as a "proc start" as reported by the **bos status** command. If the time stamp on the binaries is later, the BOS Server stops the process and starts a new instance of it using the new binaries.

Restarting a server process causes the file system to become inaccessible in varying degrees. Restarting a database server process can make the entire system unavailable to everyone for about 2 minutes if it triggers election of a new Ubik coordinator for that process. Restarting a process of some other type inconveniences only the people using the restarted process on that machine. The longest outage is usually 15 to 30 minutes, which results from a restart of the *fs* process (the exact length depends on how many volumes reside on that file server machine).

The default time for binary checking is 5:00 a.m. each day, when any outage resulting from a restart probably affects a small number of people. You may change this time with the **bos setrestart** command described below, and inspect its current value with **bos getrestart**.

(These same commands also allow you to check and set the weekly time at which the BOS Server restarts all the servers, a step that AFS takes to reduce the effects of potential core leaks. The default for that restart is Sunday mornings at 4:00 a.m. See section 4.9.)

11.4.1. Using New Binaries Immediately

If you want the file server machines to begin using newly installed binaries immediately, you must use the **bos restart** command described below to restart the appropriate server process(es). You will need to repeat the command on each file server machine.

Even though the outages caused by restarting usually last no more than 15 minutes, you may wish to schedule the restarts for off-peak times, and warn users of the outage if possible.

⇒ To inspect the scheduled time for binary checking

Step 1: Inspect the time each day at which the BOS Server will restart any process with a new binary file in */usr/afs/bin*.

% **bos getrestart** <*machine name*>

where

getr is an acceptable abbreviation of **getrestart**.

machine name specifies the file server machine whose restart times you want to check.

The second line of output shows you the time for binary checking; the first line shows weekly restart time.

An example output:

```
% bos getrestart fs4.transarc.com
Server fs4.transarc.com restarts at sun 4:00 am
Server fs4.transarc.com restarts for new binaries at 5:00 am
```

⇒ To set the scheduled time for binary checking

Step 1: Verify that you have the necessary privilege (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Set the binary check-and-restart time.

% **bos setrestart** <*machine name*> **"**<*time to restart server*>**" -newbinary**

where

setr is an acceptable abbreviation of **setrestart**.

machine names the file server machine whose BOS Server should check for new binaries at the specified *time to restart server*.

time to restart server
is the time each day that the BOS Server will perform its check. You probably want to use a time when system usage is low, because a temporary outage will occur if the BOS Server finds new binaries and restarts the server.

Note: if this time is within one hour of the current time, the BOS server will not perform the restart until tomorrow.

You can express the time in several ways. Enclose it in quotes to show that it is a single argument with spaces in it. In all cases, separate the hours from the minutes with a colon (*5:00* is the default). Twenty-four hour time (military time) is understood, but you can also use *1:00* through *12:00* if you add *am* (or *a.m.*) and *pm* (or *p.m.*) to distinguish times before and after noon.

You can disable binary checking by specifying *never* as the time. The *AFS Command Reference Manual* defines all of the possible values for this argument.

-newbinary distinguishes this restart time from the weekly one discussed in section 4.9.

11.5. Reverting to an Old Version of Binaries

In rare instances, a new version of binaries may cause problems serious enough to require reverting to the previous version. Just as with installing binaries, you should revert every file server machine back to the same version. Be sure to issue the **bos uninstall** command described here on each binary distribution machine in your cell. The Update Server client portion on each file server machine will pick up the reverted version, just as it picks up new binaries.

11.5.1. How Reverting to an Old Version Works

As section 11.3 explains, when you use **bos install** to install new binaries, the BOS Server saves the current version with a *.BAK* extension and moves the current *.BAK* version to *.OLD* if it is at least a week old.

When you revert to the previous binaries with **bos uninstall**, the *.BAK* version replaces the current (malfunctioning) binaries. The current binaries are discarded, but they should still be in the directory from which they were recently installed, if you want them for some reason (you might maintain a directory in AFS for each system type you use, in which you keep "safe" copies of the binaries that Transarc Corporation sends you). The current *.OLD* version, if any, becomes the new *.BAK* version.

If there is no current *.BAK* version, the **bos uninstall** operation will fail, as indicated by an error message. There may still be a *.OLD* version, and you could move it (with **mv**) to the *.BAK* version before reissuing **bos uninstall**.

11.5.2. Using the Reverted Binaries Immediately

By default, the file server machines will not use reverted server process binaries until the BOS Server does its daily check-and-restart of */usr/afs/bin*. If the reason for reverting is that the current binaries do not work, you should force the relevant process(es) to stop and restart on every file server machine. Instructions for restarting processes appear in section 11.3.

No restart is necessary when you revert to a previous version of binaries for a command suite such as **bos**. The next time someone issues a command that invokes them, the reverted binaries will be used automatically. You will probably want to revert to the previous command binaries on client machines, too.

11.5.3. Reverting Within an Alternate Directory

You can use the **bos uninstall** command for binaries kept in a directory other than */usr/afs/bin*, by adding the **-dir** argument to specify the alternate directory's pathname. See the *AFS Command Reference Manual*.

⇒ To revert to the previous version of binaries

Step 1: Verify that you have the necessary privilege (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Replace the binaries on the binary distribution machine of each CPU/operating system type with their .*BAK* version. (If you have forgotten which machine is the binary distribution machine for a certain CPU/operating system type, see section 4.4.)

% **bos uninstall** <*machine name*> <*files to uninstall*>

where

u is an acceptable abbreviation of **uninstall**.

machine name specifies a binary distribution machine. If you accidentally specify a non-binary distribution machine, the binaries will be overwritten the next time the Update Server client portion checks */usr/afs/bin* on its binary distribution machine and discovers that the binaries it is using do not match. This will happen within a few minutes.

files to uninstall names each binary file in **/usr/afs/bin** you wish to replace with its .*BAK* version. A simple file name (such as *bosserver* or *volserver* for server processes, *bos* or *vos* for commands) is adequate since */usr/afs/bin* is assumed.

Step 3: Repeat step 2 on each binary distribution machine in your cell. (If you have forgotten which machines are the binary distribution machines for the different CPU/operating system types you have, see section 4.4.)

Step 4: If you want the file server machines to use reverted server process binaries immediately, restart the relevant process(es) by following the instructions in section 11.3. A service outage will result. (For database server processes, the outage lasts approximately two minutes; for the *fs* process, possibly from 15 to 30 minutes.) No restarting is necessary if reverting to a previous version of the binaries for a command suite.

11.6. Checking Binary Version Dates

You can check the compilation dates for all three versions of a binary file in */usr/afs/bin*--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.

If the binary dates you wish to check are in a directory other than */usr/afs/bin*, you may check them using the **-dir** argument. See the *AFS Command Reference Manual*.

⇒ To check the compilation dates of binaries

Step 1: At the command shell prompt (%), type

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

where

getd	is an acceptable abbreviation of **getdate**.
machine name	specifies the file server machine for which you want to check binary dates in */usr/afs/bin*.
files to check	names each binary file for which you wish to check the dates. A simple file name (such as *bosserver* or *volserver* for server processes, *bos* or *vos* for commands) is adequate, rather than a complete pathname.

11.7. Removing Obsolete .BAK and .OLD Copies

Once processes with new binaries have been running without problems for a number of days, you may wish to remove all *.BAK* and/or *.OLD* files from */usr/afs/bin*, both to reduce clutter and to free space on the file server machine's local disk (for some AFS processes, each version of a binary file takes up approximately 150 kilobytes).

You can use the **bos prune** command to remove

- All files in */usr/afs/bin* with a *.BAK* extension.
- All files in */usr/afs/bin* with a *.OLD* extension.
- All files in */usr/afs/logs* called *core*, with any extension.
- All three types at once.

⇒ To remove obsolete binaries

Step 1: Verify that you have the necessary privilege (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Remove obsolete files. Provide at least one flag in addition to the machine name.

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

where

p	is an acceptable abbreviation of **prune**.
machine name	specifies the file server machine with the obsolete files.
-bak	removes all the files with a *.BAK* extension from */usr/afs/bin*. You may combine it with **-old** and **-cores**, but not with **-all**.
-old	removes all the files a *.OLD* extension from */usr/afs/bin*. You may combine it with **-bak** and **-cores**, but not with **-all**.
-core	removes all core files from */usr/afs/logs*. You may combine it with **-bak** and **-old**, but not with **-all**.
-all	combines the effect of the other three flags. Do not combine it with the other three flags.

11.8. Adding and Mounting a New Disk and/or Partition

One of the primary advantages of a distributed file system is that it is easy to add storage space simply by adding new file server machines, or by adding disks to existing file server machines. The *AFS Installation Guide* explains how to add file servers; this section explains how to add disks.

Adding a disk requires powering off the machine, which in turn requires shutting down the AFS server processes running on it and causing a service outage. The AFS-related aspects of adding and mounting a new disk are not particularly involved, so the duration of the outage depends mostly on how long the actual installation takes. You may wish to schedule disk additions for low-usage periods, and warn your users of the outage.

Once you complete the following instructions, the new disk will be ready to house AFS volumes, either new ones created using **vos create** or volumes moved from other partitions using **vos move**. See chapter 5 for information about those commands.

Note: These instructions assume that the file server machine's initialization file (*/etc/rc* or equivalent) has been modified to start up the BOS Server (and thus the other AFS server processes) automatically on reboot. The required addition should already appear in */etc/rc* or equivalent, as described in the *AFS Installation Guide*. The addition is:

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

11.8.1. About Adding a New AFS Partition

Each file server machine partition that houses AFS volumes is mounted at a directory whose name has the form

*/vicep**x***

where ***x*** is a letter. By convention, the first AFS partition on a machine is mounted at */vicepa*, the second at */vicepb*, and so on.

In order for the File Server and Volume Server to access the volumes kept on such partitions, the corresponding directory must be a subdirectory of the machine's root (/) directory, not a subdirectory of, for instance, */usr*. Also, the directory must be correctly mapped to the partition's device name in the file server machine's */etc/fstab* file or equivalent.

⇒ To add and mount a new disk to house AFS volumes

Step 1: Verify that you have the privilege necessary for shutting down AFS server processes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: Determine how many AFS partitions you will divide the new disk into. You will need to create a subdirectory in the machine's root (/) directory for each one, which will be used as its mount point.

Directories corresponding to AFS partitions must have names of the form */vicep**x***, where ***x*** is a letter of the alphabet. Use letters consecutively starting with ***a***. If you cannot remember which partition names you have used already, issue the **vos listpart** command.

% **vos listpart** <*machine name*>

where

listp is an acceptable abbreviation of **listpart**.

machine name specifies the file server machine to which you will add a disk.

Step 3: Become "root" in the machine's UNIX file system, if you are not already, by issuing the **su** command.

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

Step 4: Create each directory you decided to add in step 2.

mkdir /vicep<*x*>

Step 5: Using a text editor, modify the machine's file systems registry file (*/etc/fstab* or equivalent) to map each new disk partition to its corresponding directory, which you created in step 4. The proper format for device (that is, disk partition) names and for the format of lines in this file differs for each system type; check the existing file to learn the proper syntax.

Step 6: Have the BOS Server shut down the other AFS server processes running on the machine, in preparation for powering off the machine. The BOS Server will not shut itself down, but will terminate safely when you turn off the machine.

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

where

shut is an acceptable abbreviation of **shutdown**.

machine name specifies the file server machine you are about to turn off.

-wait indicates that the command shell prompt should not return until all processes actually shut down. If you do not provide it, the prompt returns almost immediately, even if all processes have not yet shut down.

Step 7: Install the new disk on the file server machine, format it if required (consult the manufacturer's documentation), and adjust the disk's partition table to reflect the changes you made to */etc/fstab* or equivalent in step 5. The proper method for changing a partition table varies; consult your UNIX administrator's documentation.

Step 8: Verify that when you turn the machine back on, the BOS Server starts itself and all of the other AFS server processes listed in */usr/afs/local/BosConfig*. This should happen automatically if */etc/rc* has been properly modified. If you wish, issue the **bos status** command to check that all are running properly; section 4.3 describes this command fully.

% **bos status** **<***machine name***>**

11.9. Removing and Unmounting a Disk and/or Partition

This section outlines the steps necessary for removing and unmounting a disk from a file server machine. The steps are almost the reverse of those for adding a disk, but additional steps are required to prevent unwanted data loss.

Like adding a disk, removing one requires turning off the file server machine and causing an outage. You may wish to schedule disk removals for low-usage periods, and warn your users of the outage.

Note: These instructions assume that the file server machine's initialization file (*/etc/rc* or equivalent) has been modified to start up the BOS Server (and thus the other AFS server processes) automatically on reboot. The required addition should already appear in */etc/rc*, as described in the *AFS Installation Guide*.

The addition is:

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

⇒ To unmount and remove a disk housing AFS volumes

Step 1: Verify that you have the privilege necessary for shutting down AFS server processes and moving volumes (inclusion in */usr/afs/etc/UserList*). If necessary, issue **bos listusers**, which is fully described in section 20.5.

% **bos listusers** <*machine name*>

Step 2: List the volumes housed on the partition(s) of each disk you are about to remove, decide which of them you want to retain in the file system (if any). If you wish to look at individual partitions, repeat this command for each partition.

% **vos listvldb -server** <*machine name*> **[-partition** <*partition name*>**]**

where

listvl is an acceptable abbreviation of **listvldb**.

machine name specifies the file server machine whose disk you are removing. You must include the **-server** switch, but may shorten it to **-s**. Provide this argument alone to see all volumes kept on this file server machine, or with *partition name*.

partition name specifies a partition on the file server machine. You must include the **-partition** switch, but may shorten it to **-p**.

Step 3: Move any volumes you wish to retain in the file system to alternate partitions. You can only move ReadWrite volumes. See section 5.9 to learn about how to "move" ReadOnly and Backup volumes.

vos move <*volume name or ID*> <*machine name on source*>
<*partition name on source*> <*machine name on destination*>
<*partition name on destination*>

where

volume name or ID
identifies the ReadWrite volume to be moved, either by its complete name or its volume ID number.

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 the volume will move.

partition name on destination
names the partition to which the volume will move.

Note: Unless you have a very good reason for doing so, you should not interrupt a **vos move** command in progress. For more information, see the entry on **vos move** in the *AFS Command Reference Manual*.

Step 4: **(Optional.)** If there are any volumes on the affected partitions that you do not wish to retain in the file system, use the AFS Backup System to transfer them to tape. Keeping a tape archive of deleted volumes is always recommended. See chapter 6.

Step 5: Become "root" in the machine's UNIX file system, if you are not already, by issuing the **su** command.

% **su root**
Password: <*root's password*>

Step 6: Issue the **umount** command for each partition on the disk to be removed.

Note: Repeat this command for each partition to be unmounted.

\# **cd /**
\# **umount /dev/**<*partition's block device name*>

Step 7: Using a text editor, modify the file */etc/fstab* (under AIX, */etc/filesystems*), removing the entry for each disk partition you umounted in step 6.

Step 8: Remove from the machine's root (/) directory the mount points associated with the umounted partitions.

\# **rmdir /vicep**<*x*>

Step 9: Have the BOS Server shut down the other AFS server processes running on the machine, in preparation for powering off the machine. The BOS Server will not shut itself down, but will terminate safely when you turn off the machine.

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

where

shut is an acceptable abbreviation of **shutdown**.

machine name specifies the file server machine you are about to turn off.

-wait indicates that the command shell prompt should not return until all processes actually shut down. If you do not provide it, the prompt returns almost immediately, even if all processes have not yet shut down.

Step 10: Remove the disk from the file server machine, following the manufacturer’s instructions.

Step 11: Verify that when you turn the machine back on, the BOS Server starts and starts all of the other AFS server processes listed in */usr/afs/local/BosConfig*. This should happen automatically if the initialization file (*/etc/rc* or equivalent) has been properly modified. If you wish, issue the **bos status** command to check that all are running properly; section 4.3 describes this command fully.

% **bos status** <*machine name*>

Managing Client Machines

12. Client Machine Processes and Files

This chapter describes the configuration of a machine that will serve as an AFS client. The two basic requirements for an AFS client machine are:

- That its kernel contain the set of modifications that allow it to hook into AFS. The set of modifications is commonly referred to as the Cache Manager. See section 12.1 for further details.
- That it have local disk space adequate to house several configuration files and a cache directory. Diskless machines may access AFS if they are NFS clients and use the NFS/AFS Translator available as an add-on to AFS.

This chapter has the following sections:

12.1 About the Cache Manager 12-2
12.2 Configuration Files in */usr/vice/etc* 12-3
12.3 Cache-related Files in */usr/vice/cache* 12-5

This chapter only describes the AFS-related kernel modifications and files on a client workstation; it does not explain how to install them. See the *AFS Installation Guide* for instructions.

To learn how to maintain a client workstation, see the other chapters in this section of this manual.

12.1. About the Cache Manager

Every AFS client machine must use a kernel modified to include routines that allow communication with the AFS server processes. These kernel modifications are commonly referred to as the ***Cache Manager***. The Cache Manager is often spoken of as an active entity (like a process), both for convenience and because its behavior often makes it appear active. Nevertheless, keep in mind that this usage is not strictly accurate.

As explained in section 1.3.11, the Cache Manager's main responsibility is to fetch files on behalf of application programs running on its machine. It locates files by contacting the VL Server to learn volume locations, then translates the application programs' system-call file requests into remote procedure calls to the File Server. Once the File Server delivers a file, the Cache Manager stores ("caches") the file before delivering it to the application program.

The Cache Manager makes sure that its cached ReadWrite files and ReadOnly volumes match the source version at the File Server by storing the ***callbacks*** sent by the File Server. A callback constitutes the File Server's promise to contact the Cache Manager should the source version of the file or volume change while the Cache Manager has a cached copy. When the File Server breaks the callback associated with a file or volume, the Cache Manager knows it must retrieve the new version when an application program next requests data from that changed file or volume.

12.2. Configuration files in */usr/vice/etc*

The directory */usr/vice/etc* on every client machine's local disk contains files that the Cache Manager needs in order to work properly.

It is not critical that these configuration files be identical on every client machine, but differences can result in noticeable performance differences between machines. For example, if machine A's *CellServDB* includes a foreign cell but machine B's does not, users of machine A will be able to access files in that cell and users of machine B will not. You may wish to update these files from a central source (see the descriptions below for pointers to sections that discuss update strategy). This improves the uniformity of client machine performance, and it is more convenient than altering each machine's copy individually.

The files in this directory include:

afsd The binary file for a program that runs whenever the machine reboots. It initializes several daemon-like processes important for proper Cache Manager functioning, such as the process that handles callbacks.

cacheinfo This one-line file has three fields separated by colons:

- The first field specifies the directory where the Cache Manager mounts AFS. The standard value is `/afs`.
- The second field names the "cache directory" where the Cache Manager should create its cache files. The standard value is `/usr/vice/cache`. This field must contain a value even if the machine uses memory cache.
- The third field specifies the number of 1024-byte blocks that the Cache Manager is allowed to use in the cache directory defined in the second field or in memory.

Following is an example file that grants the Cache Manager 25000 blocks:

```
/afs:/usr/vice/cache:2500
```

The *AFS Installation Guide* explains how to create this file as you install a client machine. You will seldom need to alter the settings in it, with the possible exception of the cache size (the third field). The **fs cachesize** command allows you to change the cache size as needed without rebooting. To change the value permanently, you must edit *cacheinfo* and then reboot the workstation. See section 13.3 for more details.

CellServDB This ASCII file names the database server machines in the local cell and in any foreign cell that you decide the user should be able to contact. (Recall that database server machines are the selected machines that run the Authentication, Protection, Volume Location, and Backup Server processes; see section 1.4.4 for a review.)

The Cache Manager must be able to reach a cell's database server machines in order to fetch files from the cell (this applies to the local cell, too). Incorrect or missing information about a cell will make the Cache Manager unable to fetch files. What you include in this file determines which cells the user of this machine will be able to contact. If you want the user to continue accessing a cell, **it is important to update this file whenever a cell's database server machines change.**

This *CellServDB* is not the same as the *CellServDB* that file server machines keep in their */usr/afs/etc* directory. This, the client version, may list the database server machines for foreign cells as well as the local cell. The file server machine version lists only the local cell's database server machines, because server processes never contact processes in other cells. (Chapter 9 describes how to maintain */usr/afs/etc/CellServDB*.)

The Cache Manager consults this file only at reboot, when the information is transferred into the kernel itself. (This differs from how the server processes consult their version of *CellServDB* each time they need the relevant information.) To draw attention to changes to *CellServDB*, you must reboot the workstation; or change the information in the kernel directly with the **fs newcell** command. See chapter 15 for more details.

CellServDB is one of the files that is most convenient to update from a central location. Chapter 15 offers suggestions.

ThisCell This ASCII file contains a single line that specifies the complete Internet domain-style name of the cell to which this workstation belongs. Examples are `transarc.com` and `andrew.cmu.edu`. This value defines the cell in which the machine's user becomes authenticated by default, and whose server processes will be contacted by the command interpreters (for example, **bos**) working on this machine.

The *AFS Installation Guide* explains how to create this file as you install a machine as an AFS client. Changing its value later redefines which cell this machine belongs to, and so changes the default authentication of the machine's user. See section 13.7.

12.3. Cache-related files in */usr/vice/cache*

The **/usr/vice/cache** directory contains files that the Cache Manager creates and maintains for its own use, with one exception: *AFSLog* contains trace information that developers may find useful when debugging the Cache Manager. The other files are not intended for human use, and the directory is protected even against listing, except to "root" users.

You should never modify or delete any files in this directory—their absence can cause a kernel panic. Rebooting the machine should restore normal performance in case of a panic or accidental modification of files in */usr/vice/cache*.

The files include:

AFSLog The file contains the trace output produced when **fs debug** is used to turn debugging on. It is a technical file which only developers familiar with the AFS source code will find useful.

CacheItems A binary-format file in which the Cache Manager records information about each *V**digits*** file in its cache (see just below), including its file ID number (fID) and data version number.

VolumeItems A binary-format file in which the Cache Manager records the volume-to-mount point relationship of the volumes it has accessed. This helps the Cache Manager to respond correctly to the **pwd** command, among others.

*V**digits*** Each such file holds up to 64 kilobytes of a file. There may be thousands of such files, as determined by the size of the cache. The numbers are sequential, but files that have been in the cache longer do not necessarily have lower numbers. Files larger than 64 kilobytes are divided into multiple cache files, the numbers of which are not necessarily sequential.

13. Configuring the Cache Manager

AFS creates a homogeneous environment for its users in several regards. For example, a cell's AFS file tree is identical when viewed from any machine that can access it, and the server processes behave identically regardless of the system type on which they are running (barring possible differences in response time). Users experience the greatest variation in AFS performance depending on how you have configured a client machine's cache and Cache Manager. This chapter explains the aspects of Cache Manager and client machine performance that you can alter. It includes the following sections:

13.1 Summary of Commands 13-1
13.2 Overview: Controlling Cache Manager Performance 13-2
13.3 Determining Cache Type, Size, and Location 13-4
13.4 Controlling Other Cache Parameters with **afsd** 13-8
13.5 Determining if a Client Allows Programs to "Run Setuid" 13-15
13.6 Setting a Client's File Server Probe Interval 13-17
13.7 Setting a Client's Cell Membership 13-18
13.8 [Logging and Disabling Cache Manager Messages 13-19
13.9 Monitoring Cache Manager Performance 13-21
13.10 Forcing the Update of Cached Data 13-24
13.11 Setting Cache Manager Preferences 13-26

13.1. Summary of Commands

Display cache size set at reboot . . . . . **cat** */usr/vice/etc/cacheinfo*
Display current cache size and usage . . . . . **fs getcacheparms**
Change disk cache size immediately but temporarily . . . . . **fs setcachesize**
Change reboot value of cache size . . . . . Edit */usr/vice/etc/cacheinfo*
Change disk cache directory location . . . . . Edit */usr/vice/etc/cacheinfo*
Initialize Cache Manager . . . . . **afsd**
Check cell's status regarding setuid . . . . . **fs getcellstatus**
Set cell's status regarding setuid . . . . . **fs setcell**
Display machine's cell membership . . . . . **cat** */usr/vice/etc/ThisCell*
Change machine's cell membership . . . . . Edit */usr/vice/etc/ThisCell*
Enable/disable low-level (debugging) trace . . . . . **fs debug**
Examine low-level trace . . . . . **cat** */usr/vice/etc/AFSLog*
Display current recipient of high-level trace . . . . . **fs monitor**
Direct/disable high-level trace . . . . . **fs monitor**
Flush cached file/directory . . . . . **fs flush**
Flush everything from entire cached volume . . . . . **fs flushvolume**
Force notice of other changes . . . . . **fs checkvolumes**
Display Cache Manager preferences . . . . . **fs getserverprefs**
Set Cache Manager preferences . . . . . **fs setserverprefs**

13.2. Overview: Controlling Cache Manager Performance

This section summarizes the aspects of Cache Manager configuration that you can control, directing you to sections that provide detailed information and instructions:

- ***disk or memory cache***. The AFS Cache Manager can use machine memory for caching instead of disk. Deciding which to use is the most basic configuration decision you must make. See section 13.3.
- ***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 section 13.3.
- ***cache location***. For a disk cache, you can alter the standard 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 section 13.3.
- ***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, since there are reasonable defaults, but it allows you precise control. Configuration scripts are available to set intelligent values for Cache Manager parameters according to a machine's configuration and usage patterns. See section 13.4.
- ***knowledge about cells***. You enable the Cache Manager on a client machine to access a foreign cell's AFS file tree by listing the cell's database server machines in */usr/vice/etc/CellServDB* on the local disk. See chapter 15.
- ***setuid privilege***. You can also control whether the Cache Manager allows programs from a cell to execute with setuid privilege. By default programs will execute with no setuid privilege. See section 13.5.
- ***cell membership***. Each client belongs to a certain cell, as defined by the file */usr/vice/etc/ThisCell*. Cell membership determines the default cell in which the machine's users are authenticated and in which AFS commands execute. See section 13.7.
- ***activity traces***. A user may request that the Cache Manager generate a trace of its file fetching activity and direct it to any client machine. It is also possible to track the Cache Manager's internal operations for debugging purposes. These features provide flexible debugging rather than performance enhancement. See section 13.9.
- ***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 section 13.10.

- ***file server machine preferences***. The Cache Manager associates preferences in the form of numerical ranks with file server machines. Preferences bias the Cache Manager to select ReadOnly volumes from one file server machine over those from another. The Cache Manager determines default ranks for all local and foreign file server machines from which it must access ReadOnly volumes. You can modify the preferences to direct the Cache Manager to prefer some file server machines over others. See section 13.11.

You must make all configuration changes on the client machine itself (at the console or over a **telnet** connection); remote configuration is not possible. Some configuration commands require you to log into the machine's UNIX file system as *root*, whereas others with less serious consequences require no privilege. All files mentioned in this chapter must reside on the local disk of each client.

AFS's **package** program can simplify other aspects of client machine configuration, including those you would normally set in the initialization file (*/etc/rc* or equivalent). See chapter 14.

13.3. Determining Cache Type, Size, and Location

This section explains how to select a memory or disk cache, how to examine and set the size of either type of cache, and how to set the location of the cache directory for a disk cache.

13.3.1. Using Disk or Memory Caching

The Cache Manager can cache files in machine memory rather than on disk. Because memory access is faster than disk access, a memory cache generally performs better than a disk cache of the same size.

The Cache Manager uses a disk cache by default. To make it use a memory cache, use the **-memcache** flag on the **afsd** command, which initializes the Cache Manager. (Normally, the **afsd** command is placed in the machine's initialization file—*/etc/rc* or equivalent—rather than typed at the console.) When the **-memcache** flag is used, the Cache Manager does no disk caching, even if the machine has a disk.

13.3.2. How the Cache Manager Determines Default Cache Size and Location

For both disk and memory caches, the Cache Manager learns how big to make the cache by consulting the */usr/vice/etc/cacheinfo* file at initialization. This file also defines the cache directory for a disk cache. You must create this file before running **afsd** on the machine for the first time, as directed in the *AFS Installation Guide*. The file has three fields, separated by colons:

- The first field defines where to mount AFS on the local disk. The standard choice is */afs*.
- The second field defines the local disk directory to use for caching, in the case of a disk cache. The standard choice is */usr/vice/cache*. Something must appear in this field even if the machine uses memory caching.
- The third field defines cache size as a number of kilobyte (1024 byte) blocks.

The following example mounts AFS at */afs* (the standard place), names */usr/vice/cache* as the cache directory (also the standard place), and defines cache size at 50,000 kilobytes:

```
/afs:/usr/vice/cache:50000
```

13.3.3. Setting and Listing Cache Size and Location

You can reset the cache size for both types of cache by using a standard text editor to alter the third field in */usr/vice/etc/cacheinfo* and then rebooting the machine. (You must be logged into the local machine as "root" to edit the file.) You can also use the **-blocks** argument on **afsd** to override the *cacheinfo* value at reboot. See the *AFS Command Reference Manual* for a complete discussion of the **afsd** command.

For a disk cache, you can temporarily reset cache size without having to reboot by using the **fs setcachesize** command; see the following instructions. The value set with this command lasts until the machine reboots, at which time cache size returns to the value in *cacheinfo* or to the size set with the **afsd** command's **-blocks** argument.

The **fs getcacheparms** command displays the current cache size and the amount of space the Cache Manager is using at the moment (see the instructions below).

About the only reason to change disk cache location is if a partition on the machine has substantially more space available than the one on which */usr* is mounted. Using a text editor, alter the second field in *cacheinfo* and reboot, or use the **-cachedir** argument on **afsd** (you must be logged into the local machine as "root" to edit the file).

13.3.4. Recommendations for Cache Size

Cache size influences client machine performance 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 is more likely to force the Cache Manager to discard cached data to make room for newly requested data. The next time a user asks for the discarded data, the Cache Manager must contact the File Server across the network to retrieve the discarded data; and traffic between machines is slower than communication within a machine. A larger cache saves the Cache Manager from discarding the data in the first place. (In discarding data, the Cache Manager makes sure that no changes the user has made are lost; to learn about its "discarding algorithm," see section 13.3.5).

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. <u>With a disk cache you should not devote more than about 95% of the cache directory's partition to the cache, and with a memory cache you must leave enough memory for processes and applications to run.</u>

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 usually run on the machine. The higher the demand from these factors, the larger the cache should be to maintain good performance.

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

Memory caches smaller than 1 megabyte are nonfunctional, and most users find the performance of caches smaller than 5 megabytes to be unsatisfactory. 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 may be able to use a smaller memory cache.

Note: The Cache Manager can accommodate a cache as large as one gigabyte when using AFS 3.2 and higher; an absolute limit of 700 megabytes applies for earlier versions of AFS.

⇒ To display the current cache size

Issue the **fs getcacheparms** command on the client machine.

% **fs getcacheparms**

where **getca** is an 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 change disk cache size without rebooting

Step 1: Become "root" on the client machine, if you are not already, by issuing the **su** command.

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

Step 2: Set a new disk cache size.

Note: This command does not work with a memory cache.

fs setcachesize <*size in 1K byte blocks (0 => reset)*>

where

setca is an acceptable abbreviation of **setcachesize**.

size in 1K byte blocks (0 => reset)
is the number of kilobyte blocks to be used for the cache. A value of *1024* equals a megabyte. The smallest value you can set is *1*. A value of *0* resets the cache size to the value specified in *cacheinfo*, as shown in the following instruction.

⇒ To reset disk cache size to the default, without rebooting

Step 1: Become "root" on the client machine, if you are not already, by issuing the **su** command.

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

Step 2: Reset disk cache size to either the value specified in *cacheinfo* on the client machine (by specifying "0" kilobyte blocks) or the value that was set at the last reboot (by using the **-reset** flag; this will be the value in *cacheinfo* unless the **-blocks** argument was used on **afsd**).

Note: This command does not work with a memory cache.

\# **fs setcachesize 0**

or

\# **fs setcachesize -reset**

where

setca is an acceptable abbreviation of **setcachesize**.

0 resets the cache size to the default value in */usr/vice/etc/cacheinfo*.

-reset resets the cache size to the value set at the last reboot. This will be the value in */usr/vice/etc/cacheinfo* unless the **-blocks** argument was used on **afsd**.

13.3.5. How the Cache Manager Chooses Data to Discard

When a full cache forces the Cache Manager to discard data to make room for more-recently requested data, it decides what to discard based on two factors: how much time has passed since the user last touched the data, and whether the data is "reproducible." The Cache Manager first checks the data that the user has not used for the longest time and, if it is reproducible, discards it. If the data is not reproducible, the Cache Manager moves on to the data used more recently.

The Cache Manager considers any data reproducible if it will be the same when again requested from the File Server as it was when fetched the first time. Data fetched from ReadOnly volumes is by definition reproducible. Data fetched from ReadWrite volumes and changed by the user can be considered reproducible once the changes are permanently stored at the File Server. Normally, at least some data in a cache is reproducible and can be discarded, but a small cache makes it difficult to find enough eligible data.

If the Cache Manager cannot find any data to discard, data requests from application programs will result in I/O errors. Application programs usually have means for notifying the user of such errors, but not for revealing their cause.

13.4. Controlling Other Cache Parameters with afsd

Section 13.3 discusses the only cache configuration parameters you must set: mount directory for AFS, disk cache directory, and cache size (the three fields in */usr/vice/etc/cacheinfo*). Many users find cache performance satisfactory when those are the only parameters that the administrator sets. However, if you want to experiment with fine tuning cache performance, you can use the arguments on **afsd** to control several other parameters. This section discusses a few of these parameters that have the most direct effect on cache performance. To learn about the **afsd** command's arguments, see its *AFS Command Reference Manual* page.

In addition, a number of **afsd** configuration scripts are available to specify predefined parameters for the Cache Manager on a client machine. These scripts allow you to vary aspects of the Cache Manager's initial configuration based on hardware and usage characteristics of the client machine in question. Section 13.4.4 provides information about these configuration scripts.

13.4.1. Some Cache Configuration Parameters

The cache configuration parameters with the most direct effect on cache performance include

- ***total cache size.*** This is the amount of disk space or machine memory available for caching, as discussed in detail in section 13.3.
- ***number of chunks into which the cache is divided.*** With a disk cache, each chunk is called a "V file" (discussed in section 12.3). With a memory cache, each chunk is a set of memory blocks allocated together.

  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 define total size for a disk cache at 50 megabytes and that you define the total number of chunks (V files) at 1000. There are ten users on the machine, and each has cached 100 files averaging 20 kilobytes each. All 1000 V files are used, since each chunk can hold only one file, but the cache holds only about 20 megabytes of data. When a user requests more data from the File Server, the Cache Manager must discard cached data to reclaim some V files, even though the cache is at less than 50% capacity. In such a situation, increasing the number of chunks would probably improve performance.
- ***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, since 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. You may want to increase chunk size to take advantage of very fast (for example, 100 megabit/second) links, or decrease it for especially slow networks. (Changing this parameter in an attempt to match the "average" size of a cached element on a given machine is probably not worth the effort, because the size of cached elements varies widely in most environments—for instance, directory listings, which are usually quite small, are cached along with file data. Were such matching possible, it would make it more likely that each chunk would be filled to capacity, lessening the frequency at which the Cache Manager would run out of chunks before the cache reached capacity.)

- ***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 fID and version number of the AFS file corresponding to the chunk.

  For a disk cache, dcache entries reside in */usr/vice/cache/CacheItems*; a small number (by default, one-half the number of chunks—specified with the **-files** argument—up to a maximum of 2000) are duplicated in machine memory to speed access.

  For a memory cache, all dcache entries must be in memory, because there is no disk. Thus the number of dcache entries equals the number of cache chunks. Section 13.4.3 discusses the implications of this one-to-one correspondence.

Sections 13.4.2 and 13.4.3 discuss how the Cache Manager determines defaults for the last three parameters in a disk or memory cache, respectively (section 13.3 discusses the total size parameter). They also explain how you can use **afsd**'s arguments to override the defaults.

13.4.2. Controlling Disk Cache Configuration

The default number of cache chunks (V files) in a disk cache is calculated by the **afsd** command to be the greater of the following:

- 100
- cachesize/chunksize * 1.5
- cachesize/10,000

You can override this value by specifying a positive integer with the **-files** argument. Consider increasing this value if more than 3/4 of the V files are used up upon completion of Cache Manager initialization; consider decreasing this value if only a small percentage of the V files are used up upon completion of Cache Manager initialization. (Never specify a value less than 100; doing so can cause Cache Manager performance problems.)

The following example sets the number of V files to 2000:

```
/usr/vice/etc/afsd -files 2000
```

(**Note:** Normally, **afsd** commands like the examples in this section are placed in a machine's initialization file, not typed at the console. These examples are presented only as possibilities; no claim is made about their suitability. Before using them, verify that they are appropriate for your situation.)

The default chunk size for a disk cache is 64 kilobytes. As discussed previously, the only reason to change it is to adjust to exceptionally slow or fast networks. You can use the **-chunksize** argument to override the default. Chunk size must be a power of 2. Provide an integer between *0* and *30* to be used as an exponent of 2. For example, a value of *10* sets chunk size to one kilobyte ($2^{10} = 1024$); a value of *16* equals the default for disk caches (2^{16} = 64K). A value of *0* or greater than *30* returns the size to the default. Values less than *10* (one kilobyte) are not recommended. The following example would set chunk size to 16 kilobytes (2^{14}):

```
/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 2000 entries. You can use the **-dcache** argument to change the default, specifying a positive integer (you can override the default maximum number of 2000 entries by explicitly specifying a value with the **-dcache** argument). Duplicating more than half the dcache entries in memory is not usually necessary, but may improve performance slightly because access to memory is faster than access to disk. The following example would set the number to 750:

```
/usr/vice/etc/afsd -dcache 750
```

With a disk cache, any combination of **afsd**'s arguments is allowed. The main reason for this flexibility is that the total size defined for a disk cache (in *cacheinfo* or with the **-blocks** argument) is an absolute maximum limit. It is not possible to override it by setting **-files** or **-chunksize**, or combining them. A related reason is that the Cache Manager does not have to reserve a set amount of memory on disk. V files (the chunks in a disk cache) are initially 0-length, but can expand up to the specified chunk size and shrink again, as needed. If you set the number of V files to a large value, such that expanding all of them to the full allowable size would exceed the total cache size, they simply never grow to full size.

13.4.3. Controlling Memory Cache Configuration

Configuring a memory cache differs from configuring a disk cache in that not all combinations of **afsd**'s arguments are allowed. This is because the configuration parameters interact more in a memory cache than a disk cache. If all combinations were allowed, you could accidentally set the parameters in an inconsistent way. After discussing the default values for a memory cache, the following text lists the legal and forbidden combinations.

The default chunk size for a memory cache is 8 kilobytes. As discussed previously, about the only reason to change it is to adjust to exceptionally slow or fast networks.

There is no pre-defined 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 number of chunks equals number of dcache entries in memory, and that there is no default for number of dcache entries (like number of chunks, it is calculated by dividing total size by chunk size).

The legal combinations of **afsd** arguments when configuring memory cache are

- **-blocks** alone, which overrides the default cache size specified in */usr/vice/etc/cacheinfo*. The Cache Manager divides the value of this argument by the default chunk size of 8 kilobytes to calculate the number of chunks/dcache entries. The following example sets cache size to 5 megabytes (5120 kilobytes) and the number of chunks to 640 (5120 divided by 8):

```
/usr/vice/etc/afsd -memcache -blocks 5120
```

- **-chunksize** alone, to override the default of 8 kilobytes. Chunk size must be a power of 2. Provide an integer between *0* and *30* to be used as an exponent of 2. For example, a value of *10* sets chunk size to one kilobyte (2^{10} = 1024); a value of *13* equals the default for memory caches (2^{13} = 8K). A value of *0* or greater than *30* returns the size to the default. Values less than *10* (one kilobyte) are not recommended. The following example sets the chunk size to 4 kilobytes (2^{12}). Assuming a total cache size of 4 megabytes (4096 kilobytes), the resulting number of chunks would be 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/dcache entries. For example, the following would set cache size to 6 megabytes (6144 kilobytes) and chunksize to 4 kilobytes (2^{12}), resulting in 1536 chunks:

```
/usr/vice/etc/afsd -memcache -blocks 6144 -chunksize 12
```

The following combinations explicitly set the number of chunks/dcache entries. You are discouraged from using them because they set cache size indirectly, forcing you to perform a hand calculation to determine the size of the cache. You should instead use **-blocks**, **-chunksize**, or both, allowing the Cache Manager to determine number of chunks/dcache entries itself. Because they are not recommended, no examples are shown.

- The **-dcache** argument alone explicitly sets the number of chunks/dcache entries. The Cache Manager multiples this value times the default chunk size of 8 kilobytes to derive total cache size (overriding the value in *cacheinfo*).
- The combination of **-dcache** and **-chunksize** sets chunk number and size. The Cache Manager sets the specified values and multiplies them together to obtain total cache size (overriding the value in *cacheinfo*).

The following arguments should not be used with a memory cache:

- **-files** alone. This argument controls the number of V files for a disk cache and so is is ignored if provided for a memory cache. If you want to control the number of chunks/dcache entries, you could use **-dcache**.
- **-blocks** and **-dcache**. An error message results, because it would be possible to provide values such that dividing the first (total size) by the second (number of chunks) would result in a chunk size that is not a power of 2.

13.4.4. Using Cache Manager Configuration Scripts

By default, the **afsd** command initializes the Cache Manager in a specific way. While you can experiment with the command's arguments to determine an effective alternative to the default configuration values, this can be a difficult task.

AFS includes a collection of initialization scripts that modify some of the Cache Manager's default settings when it is started on a client machine. The machine must employ a disk cache to use the scripts. The scripts are named

- *rc.afsd.small*, which initializes a "small" Cache Manager for use with a single-user machine that has approximately 8 megabytes of RAM and a 20-megabyte cache. (Using this script is equivalent to using the **afsd** command's default AFS 3.1 values; with the exception of the value used for the **-dcache** argument, these same values are the defaults for AFS 3.2.)
- *rc.afsd.med*, which initializes a "medium" Cache Manager for a machine with two to six users, 16 megabytes of RAM, and a 40-megabyte cache.
- *rc.afsd.large*, which initializes a "large" Cache Manager suitable for a machine with five to ten users, 32 megabytes of RAM, and a 100-megabyte cache.

Cache Manager performance can vary from site to site and from machine to machine. Choose a script according to the number of users who use the machine, the amount of RAM available on the machine, and the amount of disk space available for the cache on the machine.

Consider also the usage patterns of the users of the client machine. It helps to have users who use the same applications to perform the same types of tasks on the same machine. The type of work to be performed on a machine also affects your choice of the script to use. For example, if large builds and compiles are to be performed on a machine, a larger cache and use of the *rc.afsd.large* script may prove beneficial.

13.4.4.1. Cache Manager Parameters Modified by the Scripts

The scripts modify the default values used for the following **afsd** arguments:

- The **-stat** argument, which specifies the number of entries in the machine's memory to be used for recording status information about the number of AFS files in the cache.
- The **-dcache** argument, which sets the number of "dcache entries" in memory. These entries are used to store information about cache chunks.
- The **-daemons** argument, which specifies the number of "background" daemons to be run on the machine. (The **afsd** command always initializes one "callback" daemon and one "maintenance" daemon on each client machine; the **-daemons** argument has no effect on the number of callback and maintenance daemons run on a machine.)
- The **-volumes** argument, which specifies the number of volumes about which location information is to be cached in memory.

Each script modifies the default values of these arguments in a specific way. The *rc.afsd.small* script initializes the Cache Manager with the following values for these arguments:

- **-stat** is 300 (the default).
- **-dcache** is 100.
- **-daemons** is 2 (the default).
- **-volumes** is 50 (the default).

The *rc.afsd.med* script starts the Cache Manager with the following values for the arguments:

- **-stat** is 2000.
- **-dcache** is 800.
- **-daemons** is 3.
- **-volumes** is 70.

The *rc.afsd.large* script initializes the Cache Manager with the following values:

- **-stat** is 2800.
- **-dcache** is 2400 (this value overrides the default limit of 2000 imposed if the **-dcache** argument is omitted).
- **-daemons** is 5.
- **-volumes** is 128.

Refer to the description of the **afsd** command in the *AFS Command Reference Manual* for more information about the arguments.

⇒ To use a configuration script to configure a disk cache

The configuration scripts can be used only on a machine that employs a disk cache.

Step 1: Become "root" on the client machine, if you are not already, by issuing the **su** command.

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

Step 2: One of the three configuration scripts can be invoked by modifying the way the **afsd** command is called from the machine's initialization file (*/etc/rc* or equivalent). A line similar to the following is typically included in a machine's initialization file to start the Cache Manager with the **afsd** command:

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

To use one of the scripts, substitute the previous line with the appropriate following line in the machine's initialization file. To use a small cache (which employs most of the **afsd** command's default settings), include the following line:

```
/usr/vice/etc/rc.afsd.small > /dev/console
```

To use a medium cache, include the following:

```
/usr/vice/etc/rc.afsd.med > /dev/console
```

To use a large cache, include the following:

```
/usr/vice/etc/rc.afsd.large > /dev/console
```

13.5. Determining if a Client Allows Programs to "Run Setuid"

By default, AFS does not allow programs to execute with setuid privilege; however, a cell's setuid privilege status can be changed. To determine a cell's current setuid privilege, use the **fs getcellstatus** command. To enable or disable setuid privilege for a cell, use the **fs setcell** command.

When you issue **fs setcell**, you directly alter the client's kernel-resident record of the cell's setuid status, so there is no need to reboot the machine. However, changed settings do not survive reboots of the machine (the settings return to their defaults at reboot). You can alter the reboot defaults by placing the appropriate **fs setcell** command in the machine's initialization file (*/etc/rc* or equivalent).

⇒ To determine a cell's setuid status

Step 1: Issue the **fs getcellstatus** command to check the setuid status of each desired cell.

% **fs getcellstatus** <*cell name*>

where

getce	is an acceptable abbreviation of **getcellstatus**.
cell name	specifies the complete Internet domain name of the cell whose setuid status is to be determined.

The output includes a line for each specified cell, stating either

- `no setuid allowed`, to indicate that programs from the cell may not run with setuid privilege,
- `setuid allowed`, to indicate that programs from the cell may run with setuid privilege.

⇒ To change a cell's setuid status

Step 1: Become "root" on the client machine, if you are not already, by issuing the **su** command.

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

Step 2: Issue the **fs setcell** command to change the setuid status of the cell.

% **fs setcell** <*cell name*>+ **[-suid] [-nosuid]**

where

setce	is an acceptable abbreviation of **setcell**.
cell name	specifies the complete Internet domain name of each cell whose setuid status is to be altered. All specified cells receive the same status.
-suid	indicates that programs from each *cell name* may execute with setuid privilege. Provide it or **-nosuid**; omit both flags to prevent programs from each *cell name* from executing with setuid privilege.

-nosuid indicates that programs from each *cell name* may not execute with setuid privilege. Provide it or **-suid**; omit both flags to prevent programs from each *cell name* from executing with setuid privilege.

13.6. Setting a Client's File Server Probe Interval

Each AFS client machine probes file server machines to determine if those file servers are available. Specifically, the client machine probes the database file servers in its cell and those file servers that house data that the client has cached.

If a file server does not respond to a probe, the client machine assumes that it is unavailable. By default, the interval between probes is three minutes. Therefore, it could take up to three minutes for a particular client to realize that a file server has become available again after being unavailable.

For some client machines, a three minute probe interval may be too long (or too short). To adjust the length of this probe interval, use the **fs checkservers** command with the **-interval** argument.

Only the user "root" can issue the **fs checkservers** command with this argument. Once set, the probe interval will remain set until it is changed via this command or the client machine is re-booted (at which time it will return to the default setting).

⇒ To set a client's file server probe interval

Step 1: Login to the client machine as "root."

Step 2: Issue the **fs checkservers** command with the **-interval** argument.

% **fs checkservers -interval** <*seconds between probes*>

where

checks is an acceptable abbreviation of **checkservers**.

seconds between probes
specifies the length of the probe interval in seconds.

13.7. Setting a Client's Cell Membership

Each client machine belongs to a particular cell, as defined by the contents of */usr/vice/etc/ThisCell* on its local disk. The machine's cell membership determines three defaults important to users of the machine:

- Which cell users of the machine authenticate in when they issue **login** or **klog**. The effect is two-fold:
  - The **login** and **klog** programs contact an Authentication Server in the cell indicated in *ThisCell*.
  - By default, the login program combines the contents of *ThisCell* 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 *ThisCell* is incorrect, the user cannot authenticate in the local cell.
- which cell the Cache Manager considers its local, or home, cell. As explained in section 13.5, programs from the local cell may execute with setuid privilege by default, while programs from foreign cells may not.
- which cell's server processes are contacted by AFS commands issued on this machine.

⇒ To examine a client machine's cell membership

Step 1: Examine the contents of */usr/vice/etc/ThisCell*.

% **cat /usr/vice/etc/ThisCell**

For example, the following command was issued on a machine that belongs to the Transarc Corporation cell:

% **cat /usr/vice/etc/ThisCell**
`transarc.com`

⇒ To set a client machine's cell membership

Step 1: Become "root" on the client machine, if you are not already, by issuing the **su** command.

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

Step 2: Using a text editor, replace the cell name in */usr/vice/etc/ThisCell*.

Step 3: **(Optional).** Reboot the workstation, if you wish the Cache Manager to notice the change in cell name immediately. On many system types, the appropriate command is **/etc/reboot**; consult your UNIX administrator's guide. The **login** and **klog** programs and command interpreters notice the change the next time they are invoked, without requiring a reboot.

\# **sync**
\# **/etc/reboot**

13.8. Logging and Disabling Cache Manager Messages

By default, the Cache Manager displays user messages on user screens and system-level or console messages on the Cache Manager console. The **fs messages** command allows you to direct the Cache Manager to:

- Log user and/or console messages to the *AFSLog* file (and disable logging after enabling it).
- Disable the display of user and/or console messages (and enable message display after disabling it).

Using **fs messages**, you can log and/or disable user messages, console messages, or both.

⇒ To log Cache Manager messages (and disable logging)

Step 1: Become "root" on the machine containing the appropriate Cache Manager.

Step 2: Issue the **fs messages** command using the **-log** argument to specify the type of messages to be logged (or to disable logging).

% fs me -log <user|console|all|none> [-writethru]

where

me is an acceptable abbreviation of **messages**.

-log specifies which types of messages are to be logged to the file *AFSLog* stored in the directory specified by **afsd** (by default */usr/vice/etc*). The valid values are listed below:

- **user**: Log user messages to *AFSLog*.
- **console**: Log console messages to *AFSLog*.
- **all**: Log user messages and console messages to *AFSLog*.
- **none**: Do not log any messages to *AFSLog*. This is the default.

-writethru indicates that messages being logged should be written directly to *AFSLog* rather than being stored in the log buffer. By default, logged messages are stored in a two kilobyte buffer; when the buffer is filled, the contents are written to *AFSLog*. Using this flag makes messages immediately available in *AFSLog*, but adversely affects Cache Manager performance.

⇒ To disable Cache Manager message display (and re-enable display)

Step 1: Become "root" on the machine containing the appropriate Cache Manager.

Step 2: Issue the **fs messages** command using the **-show** argument to specify the type of messages to be displayed.

% **fs me -show <user|console|all|none>**

where

me is an acceptable abbreviation of **messages**.

-show specifies which types of messages are to be sent to the screens/console. The valid values are listed below:

- **user**: Send user messages to user screens.
- **console**: Send console messages to the console.
- **all**: Send user messages to user screens and console messages to the console. This is the default.
- **none**: Do not send any messages to the screens/console.

13.9. Monitoring Cache Manager Performance

By default, the Cache Manager produces no trace of its activities. However, it can be directed to produce two types of traces:

- A low-level trace that can be helpful in debugging the Cache Manager.
- A high-level trace of the directory and file elements it is fetching and storing. The trace also reports when the Cache Manager loses and regains contact with a file server machine. This can be useful in everyday monitoring of Cache Manager activity.

You will normally have no use for the low-level (debugging) trace, because interpreting it requires knowledge of the AFS source code. If Transarc Corporation staff request a low-level trace to help them solve a problem you are having, you use the **fs debug** command to enable and disable it. When you enable the trace, the Cache Manager directs information it produces into the file */usr/vice/etc/AFSLog* (unless a different directory or name is specified for the file with the **-logfile** argument of the **afsd** command).

When you enable the high-level trace using the **fs monitor** command, the Cache Manager directs output to a socket on the local machine by default. It is possible to direct the trace to a remote machine, which allows it to be monitored there. In this situation, traces from more than one Cache Manager (the local one and one or more remote ones) may appear at a single machine. The **fs monitor** command will also tell you which machine is currently the target for the high-level trace.

The high-level trace is available because AFS's original development environment included a program capable of intercepting and displaying such information. **Transarc Corporation does not provide this type of monitoring program.** However, a suitable program called "Console" is available as part of the Andrew Toolkit developed at Carnegie Mellon University's Information Technology Center.

Another way to monitor Cache Manager performance is to use AFSMonitor. See chapter 7 for information on using this program.

⇒ To turn on the low-level (debugging) trace

Step 1: Issue the **fs debug** command to enable the low-level trace.

% **fs debug on [**<*afs debug level*>**] [-syslog]**

where

de is an acceptable abbreviation of **debug**.

on directs the Cache Manager to write debugging output to the *AFSLog* file.

afs debug level specifies the types of debugging output the Cache Manager writes to the *AFSLog* file. The following values can be specified for this argument:

- *1*, which causes the Cache Manager to write standard debugging information to the file. Using this value provides a good deal of general debugging output.

- *2*, which causes the Cache Manager to write low-level debugging information about the AFS network. Use this value only if you are convinced that network problems exist.
- *4*, which causes the Cache Manager to write debugging information about the RX protocol.
- *8*, which causes the Cache Manager to write debugging information about the interface layer to AFS. This value produces no useful information on machines running a Sun operating system.

If you specify a value of *1*, *4*, or *8*, the Cache manager also records in the *AFSLog* file the AFS UID of each user who accesses data from a file server machine, recording the AFS UID with each operation that accesses data. You can add the legal values to specify different combinations of output. For example, a value of *15* specifies that all possible types of output are to be provided. A default value of *1* is used if you omit the argument.

Note: The *AFSLog* file also records the type of volume (ReadWrite, ReadOnly, or Backup) accessed from a file server machine. The type of the volume is displayed along with the volume ID in the "state" flag in bitmap form. If a ReadWrite volume is accessed, the bits are clear; if a ReadOnly volume is accessed, the 1 bit is set; if a Backup volume is accessed, the 4 bit is set.

-syslog redirects debugging output to the **syslogd** daemon. Use this flag only on machines running Sun OS 4.1 or higher.

⇒ To turn off the low-level (debugging) trace

Step 1: Issue the following **fs debug** command to disable the low-level trace.

% **fs debug off**

where

de is an acceptable abbreviation of **debug**.

off directs the Cache Manager to stop writing debugging output to the *AFSLog* file.

⇒ To inspect the low-level trace

Step 1: Become "root" on the client machine, if you are not already, by issuing the **su** command.

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

Step 2: Read the contents of the */usr/vice/etc/AFSLog* file. It contains debugging output corresponding to the value you specified for the *afs debug level* argument when you enabled the trace. Interpreting the output requires familiarity with the AFS source code. (Recall that the location and name of the file can be changed with the **afsd** command.)

more /usr/vice/etc/AFSLog

Note: The **more** command is not available on AIX systems; use **pg** instead.

⇒ To check which machine is receiving the high-level trace

Step 1: Issue the **fs monitor** command.

% **fs monitor**

where **mo** is an acceptable abbreviation of **monitor**.

Two types of messages are possible:

- `Cache monitoring is currently disabled`, which indicates that the high-level trace has been disabled. This setting is appropriate if no monitoring program is available.
- `Using host` *host name* `for monitor services`, which indicates that *host name* is currently displaying the high-level trace produced by this machine's Cache Manager. If *host name* is not running a monitoring program, the trace is lost.

⇒ To disable the high level trace

Step 1: Issue the **fs monitor** command.

% **fs monitor off**

where

mo is an acceptable abbreviation of **monitor**.

off disables the high-level trace.

⇒ To direct the high-level trace to a certain machine

Step 1: Issue the **fs monitor** command.

% **fs monitor** <*host name*>

where

mo is an acceptable abbreviation of **monitor**.

host name is the complete Internet-style name of the machine that is to receive this machine's high-level trace.

Note: The *host name* specified with **fs monitor** will continue to receive the high-level trace even after the issuer logs out. If this is not desirable, the issuer should set the target back to the local machine or disable the high-level trace altogether before logging out.

13.10. Forcing the Update of Cached Data

AFS's system of callbacks guarantees that the Cache Manager has the most current version of files and directories available at the File Server. In the rare case when you suspect that the callback mechanism has not worked properly, you can force the Cache Manager to discard (***flush***) cached data. The next time an application program requests the data, the Cache Manager fetches the latest version available at the File Server.

You can control how many elements you 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 at 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 in the cache that comes from a certain volume, use the **fs flushvolume** command. This command works like **fs flush**, 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 that do not involve callbacks at all. For instance, if a volume that has not been accessed by a Cache Manager changes location, the Cache Manager may still have a record of the volume's old location. To resynchronize the Cache Manager's volume location records, use the **fs checkvolumes**. When you issue the command, the Cache Manager discards its table of mappings between volume names, volume ID numbers, and VLDB locations and creates a new one. This forces it to reference newly relocated ReadOnly volumes and renamed volumes before it can provide data from them.

⇒ To flush certain files or directories

Step 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 you wish to flush from the cache. If you omit this argument, the working directory is flushed. Flushing a directory does not flush any files or subdirectories cached from it.

⇒ To flush all data from a volume

Step 1: Issue the **fs flushvolume** command.

% fs flushvolume [<*dir/file path*>**⁺]**

where

flushv	is an acceptable abbreviation of **flushvolume**.
dir/file path	names a file or directory from each volume you wish the Cache Manager to flush. In other words, the Cache Manager flushes from its cache everything from the same volume as each specified file or directory. If you omit this argument, all data from the volume that contains the working directory is flushed.

⇒ To force the Cache Manager to notice other volume changes

Step 1: Issue the **fs checkvolumes** command.

% fs checkvolumes

where **checkv** is an acceptable abbreviation of **checkvolumes**.

13.11. Setting Cache Manager Preferences

Each Cache Manager stores a table of preferences for file server machines. A preference is stored as a file server machine's IP address and an associated "rank." A file server machine's rank is an integer in the range from 1 to 65,534 that determines the Cache Manager's preference for selecting the server machine when the Cache Manager must access a ReadOnly replica that resides on it. Preferences can bias the Cache Manager to access ReadOnly replicas from machines that are "near" rather than from those that are "distant." Effective preferences can generally reduce network traffic and result in faster access of data.

When the Cache Manager needs to access a ReadOnly replica, it first contacts the Volume Location (VL) Server to determine the names of the file server machines on which the replica resides. It then checks its internal table to determine the rank associated with each of the machines. After comparing the ranks of the machines, it attempts to access the replica on the machine that has the lowest integer rank.

If the Cache Manager cannot access the replica on the machine with the lowest rank (possibly because of a server process, machine, or network outage), it attempts to access the replica on the machine with the next lowest rank. It continues in this manner until it either accesses the replica or determines that all of the file server machines on which the replica is housed are unavailable.

The Cache Manager assigns preferences to file server machines in the local cell and from foreign cells as it accesses files from volumes on the machines. (See the *AFS Command Reference Manual* for information about how the Cache Manager calculates file server machine preferences.) It stores the preferences as IP addresses and associated ranks in the kernel of the client machine. Because they are stored in the kernel, the preferences are recalculated when the client machine is rebooted.

If the Cache Manager encounters a bad ReadOnly volume (possibly because the volume was deleted by the **salvager**), it moves the rank of that volume/server combination to the end of its preferences list. This prevents the Cache Manager from repeatedly attempting to access a bad ReadOnly volume. If the Cache Manager is unable to access a particular Database Server Machine, it will move that Database Server Machine to the end of its preferences list.

AFS provides two commands for displaying and modifying a Cache Manager's file server machine preferences. The **fs getserverprefs** command can be used to display a Cache Manager's preferences. The command's **-file** argument can be used to direct the output to a file rather than to stdout, which is the default.

The **fs setserverprefs** command can be used to set the preference rank for one or more file server machines. Preferences are specified with the command as pairs of file server machines and ranks. The first value is the name or IP address of the file server machine; the second is the rank to be associated with the machine.

Pairs of file server machines and ranks can be specified

- On the command line with the command's **-servers** argument.
- From a file with the command's **-file** argument.
- From stdin with the command's **-stdin** flag.

The arguments and flag are not mutually exclusive, so multiple sources of preferences can be specified with the same issuance of the command. You can include the **fs setserverprefs** command in a machine's initialization file (*/etc/rc* or equivalent) to load server preferences at reboot.

⇒ To display the Cache Manager's current preferences

Step 1: Issue the **fs getserverprefs** command to display the Cache Manager's file server machine preferences.

% **fs getserverprefs [-file** <*dir/file path*>**] [-numeric]**

where

gp is an acceptable alias for **getserverprefs**.

dir/file path specifies the pathname of a file to which server machine names and ranks are to be written. Omit this argument to display the output on stdout.

-numeric specifies that the IP addresses of the file server machines that will be displayed. Omit this flag to display the names of the machines. Displaying only the IP addresses is quicker than displaying the names. It is also satisfactory if the output is to be used as input to the **fs setserverprefs** command.

The output from the **fs getserverprefs** command includes a separate line for each file server machine that has a rank in the kernel of the client machine on which the command is issued. Each line displays the name of a file server machine (if its IP address can be resolved) and its rank, as follows:

```
% fs gp
fs5.transarc.com                  20000
fs1.transarc.com                  40000
fs3.transarc.com                  20001
fs4.transarc.com                  40001
145.2.50.120                      40002
fserver1.athena.mit.edu           1000
145.2.50.119                      40001
    .   .   .   .   .               . .
```

If the **-numeric** flag is included, the output displays only the IP addresses of the machines, not their names.

⇒ To set the Cache Manager’s preferences

Step 1: Become "root" on the client machine, if you are not already, by issuing the **su** command.

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

Step 2: Issue the **fs setserverprefs** command to set the Cache Manager’s preferences for one or more file server machines.

```
% fs setserverprefs [-servers <machine name and rank>+]
                    [-file <dir/file path>] [-stdin]
```

where

sp is an acceptable alias for **setserverprefs**.

machine name and rank
specifies one or more pairs of file server machines and their respective ranks. Identify file server machines by name or by IP address.

dir/file path specifies the pathname of a file that contains pairs of file server machines and their respective ranks. Identify file server machines by name or by IP address.

-stdin indicates that pairs of file server machines and their respective ranks are to be read from stdin. Identify file server machines by name or by IP address.

14. Configuring Client Machines with package

The **package** program allows you to automate the client configuration process. With **package**, you can easily configure the local disk of numerous clients by defining global configuration files.

This chapter has the following sections:

14.1 Summary of Commands 14-1
14.2 Using the **package** Program 14-2
14.3 Package Overview 14-2
14.4 The **package** Directory Structure 14-4
14.5 Example Prototype and Library Files 14-5
14.6 Package Configuration File Line Syntax 14-9
14.7 Constructing Prototype and Library Files 14-13
14.8 The Package ***Makefile*** 14-14
14.9 Modifying the ***Makefile*** 14-19
14.10 Compiling Prototype Files 14-21
14.11 Modifying Client Machines 14-22
14.12 Running **package** 14-24

14.1. Summary of Commands and Prototype/Configuration File Lines

Configure a client machine's local disk . . . . **package**

Define directory	**D**[*<update code>*] *<directory>* *<owner name>* *<group name>* *<mode bits>*
Define file	**F**[*<update code>*] *<file>* *<source file>* [*<owner name>* *<group name>* *<mode bits>*]
Define symbolic link	**L**[*<update code>*] *<link>* *<actual file>* [*<owner name>* *<group name>* *<mode bits>*]
Define block special device	**B** *<device name>* *<major device number>* *<minor device number>* *<owner name>* *<group name>* *<mode bits>*
Define character special device	**C** *<device name>* *<major device number>* *<minor device number>* *<owner name>* *<group name>* *<mode bits>*
Define socket	**S** *<socket name>* [*<owner name>* *<group name>* *<mode bits>*]

14.2. Using the package Program

The **package** program uses system-independent ***prototype files*** to define a standard disk configuration; a prototype file indicates what files should reside on the local client disk, what files should be links into AFS, etc. The prototype files are then compiled into ***configuration files*** for each different system type.

Not all clients will be configured the same. 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, **package** 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 **dkload**) 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.

14.2.1. Using Package on File Server Machines

While **package** 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, **package** will be unable to access the volumes during reboot (and until the File Server process and Volume Server process start up again).

Since **package** 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 **package** is being used for client configurations.

14.3. Package Overview

There are three main steps to follow before running **package**:

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.

14.3.1. Preparing Prototype Files

We recommend that you begin by deciding what different functions or "roles" client machines will fill. Then, you need to determine how the local disk should be configured to satisfy that role. For example, different client roles might include a regular "staff" client (providing users with AFS access), a print server (driving a printer), or a backup machine (running the AFS Backup System). Since each function/role requires a slightly different disk configuration, we suggest that you create a different ***prototype file*** for each role.

Prototype files define how the client's disk should be configured in order to fulfill a desired 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 **package** is invoked.

Section 14.5 presents methods for building flexible prototype files that are easy to maintain.

14.3.2. Compiling Prototype Files

Prototype files are usually system-independent, but may include `ifdef` statements to satisfy the needs of different system types. The prototype files are compiled to generate operating-system specific versions. During compilation, **package** 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*, as described in section 14.8.

14.3.3. Preparing Clients

Once system-specific configuration files exist, **package** is ready to run on the clients. You must first make the **package** binary available and specify the correct configuration file.

We recommend that you 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 */etc/rc* to include a call to **package**.

These steps are discussed more completely in section 14.11.

14.4. The package Directory Structure

This section assumes that the **package**-related files have been installed in three subdirectories of */afs/**cellname**/wsadmin*: *src*, *lib* and *etc*, as recommended in the *AFS Installation Guide*.

These directories contain several *sample* prototype, library, and configuration files. We suggest that you study these sample files to understand how **package** works. **You must modify these files before you attempt to use them at your site.**

14.4.1. The *src* directory

The *src* directory contains some sample prototype files (used to build the configuration files), the *Makefile* used to build them, and the resulting compiled configuration files.

Prototype files are named <*function*>.**proto**. For example, *minimal.proto* could be used to define the minimum set of library files need to run AFS, or *staff.dkload.proto*, to define a client configuration using **dkload**. Prototype files can also contain definitions for system administrative files, such as *hosts.equiv*.

The *Makefile* is used to compile the system-independent prototype files into system-specific configuration files. See section 14.8 to learn how to modify this file for use in your cell.

Configuration files are the compiled version of the prototype files and are named <*function*>.<*systype*>. Configuration files also appear in the *etc* subdirectory, that **package** accesses when configuring disks.

14.4.2. 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 name*, *group name* and *mode bits* fields in the file and the symbolic link definitions.

14.4.3. 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 may include *minimal* and *staff* prototype files compiled for different system types.

14.5. 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 should 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 **package** more effective and easy to maintain, you should create prototype files that are modular and generic, instead of specific. For example, we recommend that you create prototype files that include library files and variables.

- **Library Files**

  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**

  Variables enable you to change definitions simply by changing the variable's value.

14.5.1. 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* may 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 rt_aix221
%include ${wsadmin}/lib/rt_aix221.readonly
%include ${wsadmin}/lib/rt_aix221.AFS
%endif rt_aix221
```

```
%ifdef pmax_ul3
%include ${wsadmin}/lib/pmax_ul43.readonly
%include ${wsadmin}/lib/pmax_ul43.AFS
%endif pmax_ul3
%ifdef rt_aos4
%include ${wsadmin}/lib/rt_aos4.readonly
%include ${wsadmin}/lib/rt_aos4.AFS
%endif rt_aos4
%ifdef sun3_35
%include ${wsadmin}/lib/sun3_35.readonly
%include ${wsadmin}/lib/sun3_35.AFS
%endif sun3_35
         .
         .
         .
```

14.5.1.1. Understanding the Prototype File

In the previous example, the first uncommented line includes the */lib/base.generic* library file. This library file may contain definitions appropriate for many prototype files; the *base.generic* library file might also be included in other prototype files, like a *staff.proto* or *backup.proto*. An example library file appears in the following section.

Note that system-specific definitions are permitted through the use of `ifdef`s and variables (for example, ${wsadmin} is used to specify pathnames). Thus, the same prototype file can be used to configure a machine running AIX 2.2.1 or a Sun 3.5, 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 will be compiled into unique configuration files. For instance, the following lines in this prototype file tell **package** to use the library files *lib/rt_aix221.readonly* and *lib/rt_aix221.AFS* for the configuration file when the value `rt_aix221` has been declared. (The system-type definition is declared in the *Makefile*; see section 14.8.)

```
%ifdef rt_aix221
%include ${wsadmin}/lib/rt_aix221.readonly
%include ${wsadmin}/lib/rt_aix221.AFS
%endif rt_aix221
```

Similarly, the following lines tell **package** to use the library files *lib/sun3_35.readonly* and *lib/sun3_35.AFS* when the value `sun3_35` has been declared.

```
%ifdef sun3_35
%include ${wsadmin}/lib/sun3_35.readonly
%include ${wsadmin}/lib/sun3_35.AFS
%endif sun3_35
```

14.5.2. Example Library File

The following is part of an example library file for basic configuration definitions. A similar file, called *base.generic*, may reside in your *lib* subdirectory. Note that configurations are defined using standard `ifdef` statements.

```
         .
         .
         .
#
# Base package definitions.
#
%ifndef cell
%define cell    transarc.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  rt_aix221
%       define  AIX
%       define  rootlinks
%ifndef noafsd
%       define  afsd
%endif  noafsd
%endif  rt_aix221
         .
         .
         .
#
# 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 line***, defines a specific component of disk configuration. The proper syntax for these lines is briefly described in section 14.6; see the *AFS Command Reference Manual* for detailed descriptions.

In this example, the library file contains instructions specific to the configuration of an rt_aix221 machine. You may have similar library files in your *lib* subdirectory.

```
         .
         .
         .
#
# Generic configuration for an AFS rt_aix221 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}
```

14.6. Package Configuration File Line Syntax

Within a library file, configuration file lines are used to define the specific disk configuration. Each line can be used to define a file, directory, socket, or device on the client machine. The syntax for each valid line type is described briefly here; detailed descriptions of the fields appear in the *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: all configuration lines must appear on a single line.

14.6.1. Local Files versus Symbolic Links

To take advantage of the AFS distributed file system, we recommend you keep the number of files on the local client disk to a minimum; instead, create symbolic links that point into AFS. This will 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) definition line, **not** the `L` (symbolic link) definition line.

The following types of files **must** reside on the local disk of all AFS clients:

- **Boot sequence files executed before *afsd*.**

  Until **afsd** is executed and the Cache Manager is initialized, AFS is inaccessible from the client. Any files that are executed *before* **afsd must** reside on the local client disk.

  For example, 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 */usr/vice/cache*.

  In addition, certain UNIX files, such as */etc/rc* and */etc/fstab*, must reside on the local disk.

- **AFS Diagnostic and Recovery Files**

  Certain commands can be used to diagnose and recover from problems caused by a file server outage. AFS binaries for these commands should reside on the local client disk.

  For example, we recommend that you load the **bos** and **fs** binaries in */usr/vice/etc* on the local disk, as well as in */usr/afsws* (*/usr/afsws* should be a symbolic link into AFS). Then, set PATH variables so that */usr/afsws* appears before */usr/vice/etc*. Thus, even if users cannot access AFS (for example, due to a file server outage) they will still be able to access copies of **bos** and **fs** in */usr/vice/etc* on the local disk.

- **Files in** */usr/vice*

  The contents of the */usr/vice* directory, including the cache files in the *cache* subdirectory and the configuration files in the *etc* subdirectory, should reside on the local disk. Chapter 12 describes the contents of this directory.

14.6.2. Defining a Directory

A `D` line defines a directory to be created on the local disk. If a directory of this name already exists, **package** does nothing. However, if an element of the same name exists on the disk as a symbolic link, **package** replaces the symbolic link with an actual directory.

Use the following line to define a directory:

`D[`<*update code*>`]` <*directory*> <*owner name*> <*group name*> <*mode bits*>

The following example defines the */usr* directory:

```
D /usr root wheel 755
```

14.6.3. Defining a File

A `F` line 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, **package** replaces it with the source file.

Caution: As described in 14.6.1, some files **must** reside on the local disk; they cannot be symbolic links.

Use the following line to define a file:

`F[`<*update code*>`]` <*file*> <*source file*> `[`<*owner name*> <*group name*>
<*mode bits*>`]`

An example which creates/updates the file */bin/grep* on the local disk, using */afs/transarc.com/rt_aix221/bin/grep* as the source:

```
F /bin/grep /afs/transarc.com/rt_aix221 root wheel 755
```

In the following example, two update codes are used, and the *owner name*, *group name* 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/transarc.com/common/etc/ThisCell
```

14.6.4. Defining a Symbolic Link

An `L` line defines a symbolic link to be created on the local disk. The symbolic link may point to the AFS file system or the local disk. If the identical symbolic link already exists, **package** does nothing. However, if an element of the same name exists on the disk as a file or directory, **package** replaces the element with a symbolic link.

Note: As described in 14.6.1, some files **must** reside on the local disk; they cannot be symbolic links.

Use the following line to define a symbolic link:

L[<*update code*>] <*link*> <*actual file*> [<*owner name*>
<*group name*> <*mode bits*>]

Note: You must not create a symbolic link to a file that begins with '#' or '%' because AFS will believe that the link is actually a mount point to a regular or ReadWrite volume, respectively.

The following example creates a symbolic link from */etc/ftpd* on the local disk to */afs/transarc.com/vax_ul3/etc/ftpd* in AFS. Since the *owner name*, *group name* and *mode bits* fields are empty, the symbolic link adopts values for those fields from the actual file:

```
L /etc/ftpd /afs/transarc.com/vax_ul3
```

This example uses the `A` update code:

```
LA /etc/printcap /afs/transarc.com/common/etc/printcap.remote
            root wheel 644
```

14.6.5. Defining a Block Special Device

The `B` line defines a block special device —a device that uses data in units of multi-byte blocks, such as a disk. If a device of the same name already exists, **package** replaces it with the specified block device.

Use the following line to define a block special device:

B <*device name*> <*major device number*> <*minor device number*>
<*owner name*> <*group name*> <*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
```

14.6.6. Defining a Character Special Device

The `C` defines a character special device—a device that uses 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, **package** replaces it with the specified character device.

Use the following line to define a character special device:

`C` <*device name*> <*major device number*> <*minor device number*> <*owner name*> <*group name*> <*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
```

14.6.7. Defining a Socket

The `S` line defines a socket—a communications device for UDP and TCP/IP connections. If a socket of the same name already exists, **package** replaces it.

Use the following line to define a socket:

`S` <*socket name*> `[`<*owner name*> <*group name*> <*mode bits*>`]`

The following example defines a socket called */dev/printer*:

```
S /dev/printer root wheel 777
```

14.7. 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 will be different for each cell.

⇒ To construct a prototype file and its component library files

Step 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 *AFS Installation Guide*.

Step 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:

- ***"staff" workstation***: provides users with access to files in AFS
- ***printer server***: drives a printer; may be combined with "staff" functionality
- ***backup machine***: performs backups of AFS volumes to tape by running the AFS Backup System software
- ***dkload***: runs the dkload program that allows AFS to be dynamically loaded into the kernel

Step 3: Determine the minimum functionality needed for all clients (such as AFS setup) and place these generic definitions in one or more library file(s).

Step 4: For each type of client (printer server, backup machine, etc), place all system-independent definitions in one file, and all operating-system dependent definitions in another file.

14.8. The Package *Makefile*

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* defines the prototype and library files used and the order of compilation. We recommend that you create your *Makefile* by modifying the example provided by Transarc Corporation, as described in this section. If your system has been installed as recommended, this Makefile is */afs/**cellname**/wsadmin/src/Makefile*.

14.8.1. Overview

The **package** *Makefile* consists of several sections, which are summarized immediately below. The following subsections describe the *Makefile* sections in more detail.

Makefile Header	Function
`CONFIG=`	lists all of the configuration files to be created and defines which prototype files are compiled for which system types. See section 14.8.2.
`BASE_LIBS=`	lists the pathnames of all operating-system- and function independent library files included in any prototype files. See section 14.8.3.
`MACHINE_LIBS=`	lists the pathnames of all operating-system-specific library files included in any prototype files. See section 14.8.4.
`LIBS=`	a one line instruction that defines `LIBS` as the combination of `BASE_LIBS` and `MACHINE_LIBS`. See section 14.8.5.
`.SUFFIXES`	defines all of the suffixes that may appear on a prototype or configuration file. See section 14.8.6.

Finally, the *Makefile* contains a set of instructions that **package** follows to generate configuration files. You should not need to alter this section. See section 14.8.7.

14.8.2. 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* defines which prototype files should be compiled for which operating system types. The resulting compiled file is a system-specific configuration file.

Study the following example taken from the sample *Makefile*. Configuration files are defined by specifying the prototype-system combination as <*prototype_file*>.<*system_type*>. Note that it is not necessary to generate a configuration file for each prototype-system type combination.

```
# Makefile in /afs/transarc.com/wsadmin/src directory.
#       (C) Copyright Transarc Corporation 1989
#       Licensed Materials - Property of Transarc
#       All Rights Reserved.
#
CONFIG = \
        staff.rt_aix221 \
        staff.pmax_ul3 \
        staff.xdm.pmax_ul3 \
        staff.dkload.pmax_ul3 \
        staff.dkload.xdm.pmax_ul3 \
        staff.rt_aos4 \
        staff.dkload.rt_aos4 \
        staff.sun3_35 \
        staff.sun3_40 \
        staff.sun3x_40 \
        staff.sun4_40 \
        staff.sun4c_40 \
        staff.vax_ul3 \
        minimal.rt_aix221 \
        minimal.rt_aos4 \
        minimal.pmax_ul3 \
        minimal.vax_ul3 \
        minimal.sun3_35 \
        minimal.sun3_40 \
        minimal.sun3x_40 \
        minimal.sun4_40 \
        minimal.sun4c_40
```

14.8.2.1. Format of Entries

- The first part of the entry defines the prototype file and should be 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*, as described in section 14.8.6. This *<prototype_name>.<system_name>* definition becomes the name of the compiled configuration file.

  For example, *staff.rt_aix221* indicates that the *staff.proto* should be compiled for machines running AIX 2.2.1. The resulting compiled configuration file is called *staff.rt_aix221*.

- Each configuration file must appear on a separate line.
- A backslash should follow the `CONFIG=` header and every name but the last one. A backslash must also appear on blank lines.

14.8.3. 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 may 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
```

14.8.4. 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/rt_aix221.generic \
        ${wsadmin}/lib/rt_aix221.generic.dev \
        ${wsadmin}/lib/rt_aix221.readonly \
        ${wsadmin}/lib/rt_aix221.readwrite \
 \
        ${wsadmin}/lib/rt_aos4.generic \
        ${wsadmin}/lib/rt_aos4.generic.dev \
        ${wsadmin}/lib/rt_aos4.generic.dkload \
        ${wsadmin}/lib/rt_aos4.generic.printer \
        ${wsadmin}/lib/rt_aos4.readonly \
        ${wsadmin}/lib/rt_aos4.readwrite \
 \
          .
          .
          .
        ${wsadmin}/lib/rt_aos4.AFS \
        ${wsadmin}/lib/pmax_ul3.AFS \
        ${wsadmin}/lib/vax_ul3.AFS \
        ${wsadmin}/lib/sun3_35.AFS \
        ${wsadmin}/lib/sun3_40.AFS \
        ${wsadmin}/lib/sun3x_40.AFS \
        ${wsadmin}/lib/sun4_40.AFS \
        ${wsadmin}/lib/sun4c_40.AFS \
        ${wsadmin}/lib/rt_aix221.AFS
```

14.8.5. 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}
```

14.8.6. The .SUFFIXES Section

This section lists the valid machine-type suffixes. This list includes all machines currently supported by Transarc Corporation. Unused suffixes are ignored.

```
.SUFFIXES: .rt_aix221 \
                .pmax_ul3 \
                .rt_aos4 \
                .proto \
                .sun3_35 \
                .sun3_40 \
                .sun3x_40 \
                .sun4_40 \
                .sun4c_40 \
                .vax_ul3
```

14.8.7. The Makefile Instructions Section

The remainder of the *Makefile* controls what **package** does while generating configuration files.

Study the following instructions; it is assumed that you are familiar with programming and *Makefiles* concepts.

```
#The following appear on a single line each in the actual file

.proto.rt_aix221:; mpp -Dwsadmin=${wsadmin} -Dsys=rt_aix221
                       -Dname=$* $*.proto > $@
.proto.pmax_ul3: ; mpp -Dwsadmin=${wsadmin} -Dsys=pmax_ul3
                       -Dname=$* $*.proto > $@
.proto.rt_aos4:  ; mpp -Dwsadmin=${wsadmin} -Dsys=rt_aos4
                       -Dname=$* $*.proto > $@
.proto.sun3_35:  ; mpp -Dwsadmin=${wsadmin} -Dsys=sun3_35
                       -Dname=$* $*.proto > $@
.proto.sun3_40:  ; mpp -Dwsadmin=${wsadmin} -Dsys=sun3_40
                       -Dname=$* $*.proto > $@
.proto.sun3x_40: ; mpp -Dwsadmin=${wsadmin} -Dsys=sun3x_40
                       -Dname=$* $*.proto > $@
.proto.sun4_40:  ; mpp -Dwsadmin=${wsadmin} -Dsys=sun4_40
                       -Dname=$* $*.proto > $@
.proto.sun4c_40: ; mpp -Dwsadmin=${wsadmin} -Dsys=sun4c_40
                       -Dname=$* $*.proto > $@
.proto.vax_ul3:  ; mpp -Dwsadmin=${wsadmin} -Dsys=vax_ul3
                       -Dname=$* $*.proto > $@
all: ${CONFIG}

${CONFIG}: ${LIBS}

system: install

install: ${CONFIG}
        cp ${CONFIG} ${wsadmin}/etc

clean:
        rm -f ${CONFIG} *.BAK *.CKP
```

14.9. Modifying the Makefile

You should modify the **package** *Makefile* 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* for these reasons.

14.9.1. Adding a New Prototype File

When you create a new prototype file, add the file name and each system type for which it will be built into the CONFIG section of the Makefile.

For example, to add <*function*>.proto for pmax_ul3 and vax_ul3, you would add the following entries to the CONFIG section:

```
CONFIG = \
...
        <function>.pmax_ul3 \
        <function>.vax_ul3 \
...
```

If you have added new library files for this prototype function, add those to the MACHINE_LIBS section.

14.9.2. 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. You will also need to add any new libraries to the MACHINE_LIBS section, and the new system type to the .SUFFIXES section.

The following example shows the modifications you would make to build the "staff" and "minimal" prototype files for this new system type.

```
CONFIG = \
...
        staff.<system-type> \
        minimal.<system-type> \
...
```

If you have created corresponding library files for this new machine type, add them to the MACHINE_LIBS section.

```
MACHINE_LIBS = \
...
        ${wsadmin}/lib/<system-type>.generic \
        ${wsadmin}/lib/<system-type>.generic.dev \
        ${wsadmin}/lib/<system-type>.readonly \
        ${wsadmin}/lib/<system-type>.readwrite \
...
```

Add the new system type to the SUFFIXES section.

```
.SUFFIXES: ...\
        .<system_type> \
...
```

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.<system_type>: ; mpp -Dwsadmin=${wsadmin} \
-Dsys=<system-type>  -Dname=$* $*.proto > $
```

14.9.3. Adding New Library Files

If you added a new library file for each system type, *<system-type>.<library-file>*, add these files to the MACHINE_LIBS section of the Makefile.

```
MACHINE_LIBS = \
...
        ${wsadmin}/lib/rt_aix221.<library-file> \
...
        ${wsadmin}/lib/rt_aos4.<library-file> \
...
        ${wsadmin}/lib/pmax_ul3.<library-file> \
...
        ${wsadmin}/lib/sun3_35.<library-file> \
...
        ${wsadmin}/lib/sun3_40.<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> \
...
```

14.10. Compiling Prototype Files

The **package** program generates configuration files and installs them in the *etc* and *src* subdirectories of the **wsadmin=** directory. 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*.

Step 1: Verify that you have all privileges in */afs/**cellname**/wsadmin* and in its *src*, *lib* and *etc* subdirectories. If necessary, issue the **fs listacl** command.

% **fs listacl [**<*dir/file path*>**]**

Step 2: Change into the */afs/**cellname**/wsadmin/src* subdirectory.

% **cd /afs/***cellname***/wsadmin/src**

Step 3: Create a backup copy of the Transarc-supplied *Makefile*.

% **cp Makefile Makefile.trexample**

Step 4: Modify the `CONFIG`, `BASE_LIBS` and `MACHINE_LIBS` section of the *Makefile*, as described in Sections 14.8.2 through 14.8.4.

Step 5: Compile the prototype files using **make**.

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 **wsadmin=** directory.

% **make system wsadmin=/afs/***cellname***/wsadmin**

14.11. Modifying Client Machines

To prepare clients to run **package** automatically, we recommend that you perform the following steps on *each* client. The following instructions enable you to modify the *rc* file generically; the name of the system-specific configuration file is not used. If desired, you can invoke **package** with specific arguments, as described in the *AFS Command Reference Manual*.

1. Specify the configuration file **package** should use.

   The *.package* file in the client's root (/) directory is redirected as an argument to the **package** command; the *.package* file specifies which configuration file **package** should use.

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 will be inaccessible.
   - The **Package** executable is approximately 80 blocks. When on the local disk, it enables you to run **package** even if a file server is down. However, a file server machine outage usually makes it difficult to run **package** because most configuration file definitions refer to files in AFS. A local copy of the **package** binary will be more beneficial in large cells, where binaries may be available on several machines.

3. Modify the client machine's appropriate *rc* file to invoke **package** at reboot. When following the *rc* file, the client reboots a second time if **package** updates any files marked with the `Q` update code.

⇒ To prepare a client machine to run package

Repeat these instructions on *every* client that will run **package**.

These instructions assume that the **package** configuration files (created when the prototype files were compiled) reside in */afs/**cellname**/wsadmin/etc*.

Step 1: Log into the client as root, or issue the **su** command.

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

Step 2: Create the *.package* file in the root (`/`) directory and specify the name of the prototype file **package** should use. The name of the file should *not* include the suffix for system type (such as *.rt_aix221*, *.sun3_40*); **package** automatically determines the correct machine type.

 # echo "/afs/<*cellname*>/wsadmin/etc/<config file>" >> /.package

If you wanted a client to be configured as a staff machine (assuming the proper prototype file had been defined and compiled for the system type), you would type:

echo "/afs/<*cellname*>/wsadmin/etc/staff" >> /.package

Step 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 package locally:

% **cp /afs/<*cellname*>/<*systype*>/usr/afsws/etc/package /etc/package**

or

To create a symbolic link:

% **ln -s /afs/<*cellname*>/<*sys type*>/usr/afsws/etc/package /etc/package**

Step 4: Add the following lines to the appropriate *rc* file, *after* **afsd** is executed. If this is a file server machine, the command to execute **bosserver** should occur later in the *rc* file than the command to execute **package**.

We recommend that you use the `-v` and `-c` options. The `-v` flag directs **package** to produce a detailed trace of its actions, while the `-c` option directs **package** to append the system type to the base name of the configuration file. See the *AFS Command Reference Manual* for a description of other options.

Note: You should replace the **/etc/reboot** 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
        /etc/reboot
        ;;
*)
        echo "Update failed, continuing anyway" >/dev/console 2>&1
        ;;
esac
fi
```

14.12. Running package

After you have created and compiled prototype files and modified client machines, you are ready to run **package**. The **package** program can be invoked by reboot (if the *rc* file has been modified) or by issuing the **package** command from the command line.

Before you actually run **package**, you may want to invoke **package** with the **-n** and **-d** options. These options display a list of potential problems or debug messages without executing the **package** command.

The **package** program follows these general rules. Complete explanations are in 14.6.

- **Package** 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, **package** 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` line in the configuration file, **package** 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 *rc* file is modified as recommended in 14.11, **package** reboots the workstation automatically if any files marked with the `Q` update code are updated, and if **package** has been invoked from the *rc* file. Files updated with `Q` cause **package** to exit with status code 4, causing a reboot (as directed in the *rc* file). Files that require a reboot before changes are recognized (such as */vmunix* and */usr/vice/etc/CellServDB*) should be marked with a `Q` update code in the configuration file.
- **Package** copies the configuration file it has just used to */etc/package.**sys_type***, where ***sys_type*** reflects this machine's system type. The **package** command interpreter consults this file if you do not provide a configuration file name. You may also want to compare this file against the actual local disk configuration.

⇒ To invoke package by rebooting

Step 1: Log in to the client machine as "root" or issue the **su** command.

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

Step 2: **(Recommended.)** Verify that 1) the */.package* file identifies the desired configuration file, 2) the ***package*** binary is available as **/etc/package**, and 3) *rc* is properly modified to invoke package automatically.

Step 3: Reboot the machine, using the appropriate command.

\# **/etc/reboot**

⇒ To invoke package directly (without rebooting)

Step 1: Log in to the client as "root" or issue the **su** command.

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

Step 2: **(Recommended.)** Verify that: 1) the */.package* file identifies the desired configuration file, 2) the ***package*** binary is available as **/etc/package**, and 3) *rc* is properly modified to invoke package automatically.

Step 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]
[-debug] [-help]

where

base name of configuration file
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. **Package** 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.rt_aix221*. You could also have **package** refer to */.package* to learn the configuration file name by providing the value

‘cat /.package‘

Use either the **-config** argument with this argument or the **-fullconfig** argument described below.

full name of configuration file, or stdin for standard input
specifies the full name of the configuration file to use, complete with the machine-type extension. Examples are *staff.rt_aos4* and *minimal.vax_ul3*.

Another possibility is the string **stdin**, which indicates that the issuer will provide configuration information via standard input (stdin), either as a piped file or by typing the configuration file at the keyboard. Type **^D** (Control-D) to conclude the input.

Use either the **-config** argument with this argument or the **-fullconfig** argument described above.

-overwrite
tells **package** to overwrite elements on the local disk with the source version indicated in the configuration file, even if the disk version's "user" `w` mode bit is turned off. Files protected by the **I** update code are not overwritten; see the "F" line definition.

-noaction
indicates that **package** should not actually perform the command, but instead should print a list (at standard out) of any problems it would encounter in executing the command. If the **-verbose** flag is added, **package** produces a trace of all the actions it would perform in executing the command, but does not actually perform them.

-silent explicitly invokes **package**'s default level of reporting, which includes only a list of problems encountered while executing the command.

-verbose
indicates that **package** should produce a detailed trace of its actions (at standard out). The trace records on the transfer and ownership/mode bit setting of each element in the configuration file.

-rebootfiles
indicates that **package** should not overwrite any element marked with the `Q` update-mode code in the configuration file. This effectively prevents the machine from rebooting automatically again when **package** is invoked from */etc/rc*.

-debug enables
debugging output, which is directed to standard out (stdout) unless the issuer directs it into a file. By default, no debugging output is produced.

Step 4: **(Recommended.)** If you think files marked with the `Q` update code were updated, reboot the machine. This reboot will not occur automatically.

15. Maintaining *CellServDB* and Cell Knowledge on Client Machines

In order to fetch files from a cell, the Cache Manager must be able to contact the cell's database server machines (those machines running the Authentication, Protection and Volume Location Servers). The file */usr/vice/etc/CellServDB* on every client machine's local disk lists the database server machines in the local cell and any foreign cell that you as the system administrator decide should be reachable from the machine.

This chapter explains how to ensure that client machines in your cell have up-to-date information about local and foreign database server machines. It has the following sections:

15.1 Summary of Commands 15-1
15.2 About *CellServDB* 15-2
15.3 Listing the Contents of *CellServDB* and the Kernel List 15-6
15.4 Changing the Kernel List of Cells Only 15-8
15.5 Changing *CellServDB* 15-10
15.6 Establishing a Central Update Source for *CellServDB* 15-13

15.1. Summary of Commands

Examine contents of *CellServDB*	**cat** */usr/vice/etc/CellServDB*
Examine kernel-resident cell information.	**fs listcells**
Change kernel-resident cell information	**fs newcell**
Change *CellServDB*.	use text editor

15.2. About *CellServDB*

The Cache Manager and other processes running on a client machine must know how to contact a cell's database server machines in order to perform several common but vital functions, including:

- Fetching files. The Cache Manager contacts the VL Server in order to learn the location of the volume containing a requested file or directory. The Cache Manager cannot fetch files from a cell whose VL Server it cannot reach.
- Authenticating users at login. The login program contacts the Authentication Server in order to obtain a server ticket, 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.

Every client machine keeps a */usr/vice/etc/CellServDB* file on its local disk to record the names and Internet addresses of the database server machine(s) in the local cell. It also records any foreign cell that you as the system administrator decide the users of the machine should be able to contact.

Including or excluding a cell from *CellServDB* is part of your decision if users of the client machine can access the cell (if access is to be possible, you must also create a mount point for the cell at the level of your file tree just below */afs*). Incorrect or incomplete information in *CellServDB* about a cell can hinder or prevent access to it. See section 2 for details.

It is vital that you update *CellServDB* **to reflect changes in the identity of a cell's database server machines, particularly the local cell's.** Similarly, you need to inform other cells if you change the database server machines in your cell. The easiest way to accomplish this is to inform Transarc Corporation of the changes, so that they can be incorporated into the files in */afs/transarc.com/service/etc*; see section 15.2.2.

15.2.1. How the Cache Manager and Command Interpreters Use *CellServDB*

The Cache Manager reads */usr/vice/etc/CellServDB* only once per reboot. As the machine reboots, it copies the contents of *CellServDB* into the kernel, and thereafter uses only the kernel-resident list of cells. In contrast, AFS command interpreters consult *CellServDB* each time a user issues an AFS command, rather than using the kernel-resident list. This has several implications:

- If you want to change the cell information available to the Cache Manager, you need to change the kernel-resident information. To do this without rebooting, use the **fs newcell** command; see section 15.4 for details. The change lasts only until the client machine reboots.
- If you want to change the cell information available to command interpreters, you need to modify *CellServDB* but rebooting is not necessary (see section 15.5). If you want to change the cell information available to both the Cache Manager and command interpreters, you must change *CellServDB* and either reboot the workstation immediately or issue **fs newcell** at the same time. This is the recommended alternative because it leads to the most consistent performance.

15.2.2. Maintaining *CellServDB*

Every client machine maintains a separate copy of *CellServDB*. This allows you to vary workstation behavior if you choose, making some cells "visible" from one workstation but not another by using different versions of *CellServDB* on them. In most cases, however, it is desirable that all of a cell's client machines have the same information about cells. If users move frequently from machine to machine, it would be particularly confusing to make a cell visible on one machine but not another.

You must also update *CellServDB* on each client when a cell's list of database server machines changes. The preferred method is to use the **package** program to copy the appropriate version from a central source in your cell's file space. Normally **package** runs at reboot, but you may invoke it directly at other times if you choose. See chapter 14 for information on **package**. See section 15.6 to learn more about setting up a central *CellServDB* as a source copy for client machines in your cell.

The suggested central source for *CellServDB* is the directory

> */afs/**cell name**/common/etc*

where ***cell name*** is your cell name. You can configure **package** to consult this directory automatically.

The alternative to **package** is to alter each client machine's *CellServDB* individually. However, this becomes tedious if there are many client machines and it can lead to unpredictable differences between client machines if you forget to update *CellServDB* on some of them.

Symbolically linking */usr/vice/etc/CellServDB* to the central update directory in AFS is not a viable option. The Cache Manager copies the information from *CellServDB* into the kernel during a stage of reboot when AFS is not yet accessible (because **afsd** has not run).

15.2.2.1. Transarc Maintains a Global *CellServDB*

Transarc Corporation maintains a client version of *CellServDB* that contains up-to-date information about every cell which has agreed to make the information available. The pathname of this file is:

> */afs/transarc.com/service/etc/CellServDB.export*

You may wish to refer to Transarc's central version in determining if your central version is correct.

15.2.3. The Format of *CellServDB*

Because you create the client version of */usr/vice/etc/CellServDB* using a text editor, you need to understand its proper format. The example below should help to clarify the following formatting rules:

Rule 1: Each cell has a separate entry. The entries may appear in any order, but it is recommended (for convenience) that you put the local cell first.

Rule 2: No blank lines should appear in the file, even at the end after the last entry.

Rule 3: The first line of each cell's entry begins with the > character and specifies the cell's Internet domain name. Optionally, some white space and a comment (preceded by a # sign) may follow, to "translate" the cell name for the benefit of human readers.

Rule 4: Each subsequent line in a cell's entry names one of the cell's database server machines. The following must appear on the line, in the order given:

- The Internet address of the server, in the standard four-component decimal form.
- One or more spaces.
- A # sign, followed by the machine's complete Internet host name. Note especially that the # sign does not indicate that the host name is a comment—it is a required field.

The following example comes from */usr/vice/etc/CellServDB* on a client machine in the Transarc Corporation cell. These entries would allow users of the machine to access files in some of the cells at Carnegie Mellon University (CMU) and the Massachusetts Institute of Technology (MIT). (In the interests of space, this example includes only a few cells; it should not be interpreted as complete or accurate.)

```
>transarc.com     #Transarc (local) cell
192.54.226.1        #transarc.com
192.54.226.13       #bigbird.transarc.com
>andrew.cmu.edu   #Main CMU cell
128.2.10.2          #vice2.fs.andrew.cmu.edu
128.2.10.7          #vice7.fs.andrew.cmu.edu
128.2.10.11         #vice11.fs.andrew.cmu.edu
>athena.mit.edu   #MIT/Athena
18.72.0.43          #orf.mit.edu
18.80.0.2           #maeander.mit.edu
>cs.cmu.edu       #CMU School of C.S.
128.2.242.86        #lemon.src.cs.cmu.edu
128.2.242.45        #apple.srv.cs.cmu.edu
128.2.222.199       #papaya.srv.cs.cmu.edu
```

15.2.4. Negative Consequences of Incorrect Cell Information

You can omit a cell entirely from *CellServDB* and the resulting kernel-resident list if you do not want it to be accessible from a machine. However, negative consequences result when the list of machines is wrong for a cell that is included. The exact effect depends on whether the error is one of omission or inclusion.

- If a database server machine does not appear in *CellServDB* or the kernel-resident list, then the command interpreters and Cache Manager will not attempt to contact it. Because other database server machines (presumably) do appear in the list, the omission of a machine does not present a serious problem unless all the ones listed go down. In that case, the client cannot access the cell even though the non-listed database server machine is actually up.
- You should not include a machine that is not actually a database server machine for the cell. Since no database server processes are actually running on the machine, requests from the

client will time out. This causes delays because the client will wait the full time out period before contacting another database server machine.

15.2.5. File Server Machines Consult */usr/afs/etc/CellServDB*

File server machines keep a similar file, */usr/afs/etc/CellServDB*, on their local disks. It is not the same as the client version under discussion here: the file server version lists only the local cell's database server machines, since server processes never need to contact foreign cells.

You must update both the server and client versions of *CellServDB* when the identity of your cell's file server machines changes. See chapter 9 for information about the server version of *CellServDB*.

15.3. Listing the Contents of *CellServDB* and the Kernel List

The Cache Manager copies cell information into the kernel from */afs/vice/etc/CellServDB* as the client machine reboots, and consults the kernel-resident information from that point on. Discrepancies between *CellServDB* and the kernel-resident cell list can arise in two ways:

- If someone uses **fs newcell** to change the kernel-resident list, without also changing *CellServDB*
- If someone changes *CellServDB* but does not immediately introduce the change into the kernel, either by rebooting the machine or using **fs newcell**

If there is a discrepancy, you need to examine both *CellServDB* (with the **cat** command or a text editor), and the kernel-resident list, (with the **fs listcells** command). Remember, the information you see applies only to the workstation on which you issue the command.

⇒ To examine the contents of *CellServDB*

Step 1: Examine the contents of *CellServDB*.

% cat /usr/vice/etc/CellServDB

(You could also use a text editor. The file should consist of a list of cell entries that obey the formatting conventions outlined in section 15.2.3.)

⇒ To examine the kernel-resident cell information

Step 1: Issue the **fs listcells** command.

% fs listcells

where

listc is an acceptable abbreviation of **listcells**.

Note: If it is important that your shell prompt return immediately, run this command in the background by adding **&** on the end. Otherwise the listing can take time, because the kernel stores the Internet addresses of each cell's database server machines, rather than their names. When you issue **fs listcells**, the **fs** command interpreter contacts a name server in order to translate the addresses into names, a potentially slow process. The reappearance of the command shell prompt indicates completion. Using **^C** (or another command-interrupt signal) to interrupt the output has no ill effect.

The output for each cell will have the form

`Cell` <*cell name*> `on hosts` <*list of machines*>

The need to contact the name server can also cause possible variation in how machines are identified in the output. Depending on the success of the translation, the machine may be identified by name (in all uppercase or all lowercase letters) or by Internet address, in four-part decimal form (such as `192.54.226.1`).

A condensed example of output:

% **fs listcells**

```
   .
   .
Cell psy.cmu.edu on hosts THISTLE.PSY.CMU
Cell transarc.com on hosts transarc.com bigbird.transarc.com
Cell athena.mit.edu on hosts orf.mit.edu 18.80.0.2
   .
   .
```

15.4. Changing the Kernel List of Cells Only

The **fs newcell** command allows you to change a client machine's knowledge of a cell's database server machines immediately, without having to reboot. This command changes the list of cells and their database server machines kept in the machine's kernel, but does not change *CellServDB*. The **fs newcell** command also allows you to change a cell's name, and to define a new cell in the kernel list (that is, create an entry for a cell not previously listed).

There are several things to remember when using **fs newcell**:

- It does not allow you to remove a cell from the kernel entirely (that is, by specifying the cell's name but omitting any database server machine names). To do that, you must modify *CellServDB* and reboot the machine. See section 15.5.
- The command interpreters use *CellServDB* rather than the kernel list, so they will not notice the change. This will cause some commands to fail entirely, **klog** for example.
- The effect of **fs newcell** lasts only until you reboot the machine. To make changes permanent, you must add them to *CellServDB*.

⇒ To change the kernel-resident list of cell information

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

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

Step 2: Change the kernel-resident list of database server machines for a cell.

\# **fs newcell** <*cell name*> <*primary servers*>$^+$

where

n is an acceptable abbreviation of **newcell**.

cell name is the complete Internet domain name of the cell for which you wish to change cell information. Provide this argument even when changing information for the local cell. Examples: *transarc.com* and *andrew.cmu.edu*.

primary servers specifies each machine that serves as a database server machine for *cell name*. Include all of the cell's database server machines, even if the current list includes some of them: *primary servers* replaces the current information, rather than being added to it. You may provide either Internet-style host names (such as `fs2.transarc.com`) or four-part decimal host addresses (such as `192.54.226.13`). In the former case, the command interpreter translates the names into numerical addresses before storing them in the kernel.

Step 3: Repeat step 2 for each cell whose information you wish to alter.

Step 4: **(Optional.)** If you wish the changes you made in step 2 to remain after the next reboot of the workstation, make the same changes in the local copy of *CellServDB*. Either use a text editor or copy of *CellServDB* from your central update directory. See section 15.5.

15.5. Changing *CellServDB*

You can change a client machine's */usr/vice/etc/CellServDB* using a central update source (with or without the **package** program), or by altering each machine's copy separately. Section 15.6 has more information about setting up a central update source.

There are three methods for altering *CellServDB*, listed here in decreasing order of preference. Using **package** can simplify client machine configuration in several ways. See chapter 14 for more detailed information on how to use **package**.

- If you maintain a central update source and use **package**, alter the central copy of *CellServDB* (and make sure that the client machine's **package** configuration files refer to it).
- If you maintain a central update source, but do not use **package**, alter the central copy of *CellServDB* and copy it over to the client machine individually.
- If you do not use a central update source, edit each client machine's *CellServDB* directly.

With all three methods, if you want the machine to use the changed *CellServDB* immediately, then you must reboot it or make the same changes with **fs newcell** (described in section 15.4).

⇒ To update a client's *CellServDB* using package

Step 1: Verify that you have WRITE, READ and LOOKUP access in the directory containing your central copy of *CellServDB*. The recommended directory is

*/afs/**cell**/common/etc*

where ***cell*** is your cell name (see section 15.6). If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

\# **fs listacl [**<*dir/file path*>**]**

Step 2: Become "root" in the machine's UNIX file system, if you are not already, by issuing the **su** command.

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

Step 3: Alter the central update copy of *CellServDB*, using your favorite text editor. Section 15.2.3 defines the proper format for entries.

Step 4: Invoke the **package** program. The best method depends on whether this client's */etc/rc* file (or equivalent) includes a call to **package**:

- If */etc/rc* does call **package**, reboot the workstation. With many system types, the appropriate command is **/etc/reboot**; consult your UNIX administrator's guide. The source copy of *CellServDB* will be copied over to the machine and into the kernel-resident list immediately.

  \# **sync**
  \# **/etc/reboot**

- If */etc/rc* does not call **package**, invoke **package** directly.

  \# **/etc/package -v -c** <*name of package file*>

  where

  -v indicates that **package** should produce verbose trace output.

  name of package file is the **package** configuration file that references your cell's central update source for *CellServDB.*

  Then, if you wish the kernel-resident list of cells to be updated immediately, either reboot the machine, or use the **fs newcell** command described in section 15.4.

⇒ To update a client's *CellServDB* from a central source, without package

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

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

Step 2: Verify that you have at least WRITE, READ and LOOKUP access in the directory containing your central copy of *CellServDB* resides. The recommended directory is

*/afs/**cell**/service*

where ***cell*** is your cell name. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

\# **fs listacl [**<*dir/file path*>**]**

Step 3: Alter the central update copy of *CellServDB,* using your favorite text editor. Section 15.2.3 defines the proper format for entries in this file.

Step 4: Working at the console of the client you are altering, copy the central source over to the client, overwriting the current copy. If you are using the recommended update directory, the following command is appropriate.

\# **cp /afs/***cell***/service/CellServDB /usr/vice/etc/CellServDB**

where *cell* is your cell name.

Step 5: **(Optional).** If you wish the client to start using the information in the new *CellServDB* immediately, you must reboot it. On many system types, the appropriate command is **/etc/reboot**; consult your UNIX administrator's guide.

```
# sync
# /etc/reboot
```

⇒ To change a client’s *CellServDB* directly

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

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

Step 2: Alter */usr/vice/etc/CellServDB* using your favorite text editor. Section 15.2.3 defines the proper format for entries in this file.

Step 3: **(Optional).** If you wish the client to start using the information in the new *CellServDB* immediately, you must reboot it. On many system types, the appropriate command is **/etc/reboot**; consult your UNIX administrator’s guide.

\# **sync**
\# **/etc/reboot**

15.6. Establishing a Central Update Source for *CellServDB*

It is not difficult to set up a central update source for CellServDB. You must decide where in the file system it should reside, and who is allowed to make changes to it.

The recommended location for a central *CellServDB* is

*/afs/**cell**/common/etc/CellServDB*

where ***cell*** is your cell name.

Transarc Corporation maintains a client version of *CellServDB* that contains up-to-date information about every cell that has agreed to make such information available to Transarc. The pathname of this file is:

/afs/transarc.com/service/etc/CellServDB.export

You may wish to consult this file periodically, particularly before updating the central version of *CellServDB* that you distribute to all clients. The designated site contact should notify Transarc Corporation of any changes in your cell's database server machines so that the central version of *CellServDB* can be kept up to date.

⇒ To set up a central source for your client *CellServDB*

The following instructions assume that you choose */afs/**cell**/common/etc* as your central update directory, and that neither it nor the parent *common* currently exist.

Step 1: Verify that you have at least INSERT, WRITE, READ and LOOKUP access in your */afs/**cell*** directory. If necessary, issue the **fs listacl** command, which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

Step 2: Create the central update directory.

% **cd /afs/**<*cell*>
% **mkdir common common/etc**

where *cell* is your cell name.

Step 3: Create an initial version of */afs/**cell**/common/etc/CellServDB*. You need only do this one time, but you will probably want to continue updating *CellServDB* as more AFS cells come on-line and existing cells change their database server machines.

There are at least three methods for creating an initial *CellServDB*. They are listed here in decreasing order of preference.

- Copy Transarc Corporation's central version (*/afs/transarc.com/service/etc/CellServDB.export*). To use this method, the machine on which you are issuing commands must already have kernel-resident information about the `transarc.com` cell's database server machines. Using

fs newcell is the most practical method to put that information into the kernel; see section 15.4 for details.

- Copy into */afs/**cell**/service* a version of *CellServDB* from elsewhere in your cell's file space.
- Use a text editor to create a version from scratch. Section 15.2.3 defines the proper format for entries in *CellServDB*.

You may repeat step 3 as necessary if you find it a practical way to update the central copy of *CellServDB* in your cell.

Managing Users, Groups, and Protections

16. Creating and Deleting User Accounts With uss

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/or delete multiple accounts with the **uss bulk** command.

A single command is sufficient to create a complete account because **uss add** refers to a template file where you predefine constant or variable values for many of the components of the account (such as the name of the user's volume and the site housing it). Although you can delete the major elements of a user account with **uss delete**, the command does not use a template file; therefore, other administrative actions may be required.

You may find that the ability to automate account creation and deletion can be enhanced by using shell scripts or other programs to perform those parts of account creation and deletion that are unique to your site. Shell scripts and programs can be included in the template file used by **uss add** and in the bulk input file used by **uss bulk**. This chapter provides some ideas on setting up these scripts.

If you prefer to create user accounts by issuing a series of separate commands to the various AFS servers, see sections 17.2 and 17.5.

This chapter has the following sections:

16.1 Summary of Commands 16-1
16.2 About Creating and Deleting AFS User Accounts Using uss 16-2
16.3 About Creating Local Password File Entries Using uss 16-8
16.4 About Using uss When Converting Existing UNIX Accounts 16-10
16.5 Constructing a Template for Account Creation 16-12
16.6 Creating Individual Accounts with **uss add** 16-39
16.7 Deleting Individual Accounts with **uss delete** 16-45
16.8 Creating and Deleting Multiple Accounts with **uss bulk** 16-48

16.1. Summary of Commands

Add a single user account using a template file . . . . **uss add**
Delete a single user account . . . . **uss delete**
Add and delete multiple accounts using a bulk input file . . . . **uss bulk**

16.2. About Creating and Deleting AFS User Accounts Using uss

The **uss** suite of commands allow you to automate the creation and deletion of AFS user accounts:

- The **uss add** command can help you create all of the components of an account, one account at a time. This command uses a template file to get information about how to set up the account.
- The **uss delete** command can help you delete the major components of an account, one account at a time. To complete the account deletion, you may have to perform additional tasks manually.
- The **uss bulk** command can help you create and delete multiple accounts. It uses a bulk input file that can contain multiple calls to the **uss add** and **uss delete** commands. The bulk input file can also contain additional lines that further automate account creation and deletion.

16.2.1. Components of an AFS User Account

A user account can have many components. Only the Protection Database and Authentication Database entries are required; however, the other components add functionality and usability.

- An entry in the ***Protection Database*** defining the user's "user name" (the name typed at login). It is mapped to the user's AFS UID, the number that the AFS servers use internally when referencing users. The entry also records all the protection groups to which the user belongs. See chapter 18 for more on maintaining Protection Database entries.
- An entry in the ***Authentication Database***. This entry records the user's password in a scrambled form suitable for use as an encryption key. When the user provides a password at login (or with the **klog** command), the login program scrambles it the same way so the Authentication Server can verify that the user has provided the correct password.
- A ***volume*** that keeps all the files in the user's home directory together on a single partition of a file server machine. See chapter 5 for more about volumes.
- A space ***quota*** on the volume that limits its maximum size. When you create a volume with **vos create**, a default quota of 5000 kilobyte blocks is assigned. The **uss add** command does not assign a default, so you must define the quota yourself on the template "V" line. For more information about volume quota, see section 5.12.
- A ***mount point*** that associates the user's volume with the home directory. The mount point makes the contents of the volume visible and accessible in the file system tree. For more information about mount points, see section 5.2.6.
- Full ***ownership*** of and ***access rights*** to the home directory. In order to be able to manipulate all the files and subdirectories in the home directory, the user must own the home directory (in the standard UNIX sense) and have all the available access rights (as defined by AFS).
- An **entry in the local password file** (*/etc/passwd* or equivalent) of each machine the user will log into. Logging into a machine is necessary because AFS files are accessible only through a Cache Manager, which resides in the client machine kernel. See section 17.3.

- ***Standard files***. AFS does not require these files, but you may install them to help the user log in and out of the system more easily, receive electronic mail, enable printing, etc.

16.2.2. The Two Methods for Creating and Deleting Accounts

There are two methods for creating and deleting the components of an account. The recommended method is to use the **uss** command suite described in this chapter. This method is recommended because it creates or deletes the components of each account in the appropriate order and because it automates many of the steps associated with account creation and deletion.

The second method requires you to issue many commands (both UNIX and AFS). Creating and deleting multiple accounts at once is possible, but requires careful bookkeeping. Section 17.5 outlines all of the steps in this method.

16.2.3. The Advantages of Using a Template File for Account Creation

The **uss add** command allows you to make the same decisions about account creation as the manual method described in section 17.5. For example, you still decide the format for user names and user volume names, the location in the file tree at which to mount user volumes, and the appropriate ACL for user home directories.

However, implementing these decisions by constructing a template file for use by **uss add** provides you with several advantages over issuing individual commands:

- You do not have to remember the order in which to create or delete account components or the order of the arguments on each command because the **uss** command interpreter directs the information to the right AFS server process at the right time. Thus, you are less likely to skip a command or make other data entry mistakes.
- You do not have to type the same information over and over again. If some aspect of account creation is the same for every user, you can put a constant in the appropriate field in the template, eliminating the need to type repetitive information. (A common use of a constant is for setting space quota, which is usually the same for all new user volumes.) If some aspect of account configuration varies for each user, you can put a variable in the template; you then supply the value of the variable by using the appropriate argument on the **uss add** command line. You can combine variables and constants in a variety of ways.
- You can create multiple templates, for different groups of users. Instead of having to remember the differences between groups of users, you specify the appropriate template when issuing the **uss add** command.
- You can create different different types of user accounts, with different levels of functionality (see below).

The **uss** interface allows you to create three main types of accounts. You set the type by including or omitting certain information in the template; section 16.5 explains which information to include. Below is a list of the three types of accounts:

- **Authentication-only Account:** The user is granted entries in the Authentication and Protection Databases only. The user can authenticate in AFS, but has no home directory/volume. This type of account gives the user access to files as a locally authenticated user (instead of as *system:anyuser*), but not the ability to store anything in a personal home directory. This level can be useful for administrative accounts and guest accounts.
- **Basic Account:** In addition to the necessary Database entries, the user has a home directory/volume which he or she owns and to which he or she has full access rights. The account is otherwise "bare bones," without any standard directories or configuration files that may help with printing, logging in and out, or mail delivery.
- **Full Account:** This type of account includes standard directories and configuration files in the home directory, to make it more user-friendly (for example, *.login*, *.cshrc*, and *.logout*). Most users, especially novices, will find an account like this much more usable than the Basic Account type.

16.2.4. Previewing Account Creation and Deletion

You can preview the effects of issuing a **uss** command by including the **-dryrun** flag on the command line. Instead of actually performing the command, the **uss** command interpreter prints a trace on the screen of the actions it would perform were the command issued without **-dryrun**.

The **-dryrun** flag is useful for verifying that the account(s) will be created or deleted as you specify. However, because the command interpreter is not actually attempting to perform the specified actions, using the **-dryrun** flag will not necessarily reveal failures that can result when the **uss** command interpreter encounters problems. Some of these failures do not cause the account creation to abort, but others do. Sections 16.2.5 and 16.2.6 discuss some common causes of failures and suggest how to rectify them.

16.2.5. Avoiding Failures During Account Creation and Deletion

As in any complex operation, there are a number of possible causes for failure in account creation and deletion. Common causes include the following:

- Lack of the privilege necessary to perform a requested action. These errors tend to cause the account creation or deletion to abort, so you should verify that you have all the privileges necessary before creating or deleting accounts. Doing so is included as a step in the instructions in sections 16.6, 16.7, and 16.8.
- Incorrect syntax in template or bulk input file lines, or attempting to set values illegally. Many of these errors are fatal, so you should check your template and bulk input files carefully before issuing commands. Sections 16.5.7 through 16.5.14 detail the correct syntax of template lines and alert you to values that can cause errors. Section 16.8.1 details the correct syntax for bulk input file lines.
- File server machine or server process outages. Checking for these explicitly (with **bos status**) can be quite time-consuming, depending on the size of your cell; however, you should not issue **uss** commands if you are aware of outages.

When the **uss** command interpreter encounters error conditions that are not fatal, it generally prints a message that begins

```
uss: Warning:
```

and tells you what action it is taking to avoid exiting prematurely. For example, if a Protection Database entry already exists under the requested name, you will get the following message when attempting to create the account:

```
uss: Warning: User 'user' already in the protection database
The uid for user 'user'is AFS UID
```

The word "`Warning`" is not included in errors that are serious. These error messages describe why the command interpreter was completely unable to execute the requested action. Not all of these errors are fatal, although they still require corrective action. For example, **uss add** may be unable to mount a volume in the requested place if you lack the necessary privilege or if there is something wrong with the pathname in the "V" line <*mount point*> field. This error will not cause a fatal exit until later lines in the template attempt to install subdirectories or files under the nonexistent mount point. The appearance of an error message directly preceding the return of the command shell prompt generally indicates the error was fatal.

16.2.6. Reissuing uss Commands (to Recover from Creation/Deletion Failures)

This section describes what happens when you reissue a **uss** command and some or all of the elements already exist (for account creation) or have already been deleted (for account deletion). The most common reason to reissue a **uss** command is if you observe an error during account creation or deletion. The best recovery method is to find and fix the cause, and then reissue the same **uss** command.

The following describes what **uss add** does when it discovers that certain account components already exist:

- If a Protection Database entry already exists for the user name, **uss add** does not create a new one. A message reports the AFS UID of the existing entry.
- If an Authentication Database entry already exists for the user name, **uss add** does not create a new one. A message reports its existence.
- If a volume already exists under the name indicated in the "V" line's <*volume name*> field, **uss add** does not create a new one, reporting that one already exists. It also reports that it cannot create a VLDB entry because one already exists. However, it does alter the volume's quota, mount point, or ACL if you made changes in those "V" line fields since running the command the first time. In the case of an altered mount point, **uss add** does not remove other mount points to the volume.
- Any changes to the password/authentication restrictions and limits indicated on the "A" line are made without comment.
- The **uss add** command will attempt to issue any commands indicated on an "X" line. If the effect of a command already holds, the effect of issuing it again is the same as when you reissue it outside the context of **uss**.

The **uss add** command treats all of the remaining components in the same way: if it discovers an existing directory, file, or link with the same name as a definition on a template "D," "E," "F," "L," or "S" line, it replaces the existing element with one that fits the template definition. Use the **-overwrite** flag to control whether the **uss** command interpreter prompts for confirmation that you wish to overwrite a given element:

- Without the **-overwrite** flag, **uss add** prompts once to ask if you want to overwrite all qualifying elements; **uss bulk** prompts once for each "`add`" line in the bulk input file.
- With the **-overwrite** flag, **uss add** automatically overwrites all qualifying elements without asking for confirmation.

The following describes what **uss delete** does when it discovers that certain account components have already been deleted.

- If the volume no longer exists for the user name, **uss delete** reports this. It also reports that it cannot delete the corresponding VLDB entry because it does not exist.
- If the Authentication Database no longer exists for the user name, **uss delete** reports this.

16.2.7. The *-admin* Argument: An Important Note for Ultrix Users

If you are issuing **uss** commands on machines running the Ultrix operating system, you must fulfill two extra requirements. First, you must log into the UNIX file system as "root," because Ultrix allows only "root" to issue **/etc/chown**, even on files in AFS. (With other operating systems, all members of *system:administrators* can issue **chown** on AFS files; however, "root" privilege is still necessary for files in the UNIX file system.) The **uss add** and **uss bulk** commands both invoke **/etc/chown** on all file and directory elements created from template lines (the mount point created on the "V" line, and any element created by a "D," "E," or "F" line).

Second, you must include the **-admin** argument on the **uss add**, **uss delete**, and **uss bulk** command line. This flag is necessary because part of the task of creating or deleting an account is creating or deleting an entry in the Authentication Database. To do this, the command issuer must provide his or her password (that is, the Authentication Server does not accept the user's token, but requires a separate authentication). And since you are logged in as "root" (and for security reasons "root" should not have an entry in the Authentication Database), the Authentication Server will not recognize the pairing of "root" and an AFS password, and the account creation or deletion attempt will fail. By using **-admin**, you can specify the user name to be paired with the password and sent to the Authentication Server.

Users of other operating systems do not normally need to use the **-admin** argument, because all members of *system:administrators* can issue **/etc/chown** on AFS files. However, it can be useful when using **uss** to create files and directories in a machine's local UNIX file system. See sections 16.5.9.3, 16.5.10.3, and 16.5.11.3.

16.2.8. Privilege Requirements for the uss Commands

The privilege requirements for **uss** commands are more extensive than for commands in most other suites because a single command is making changes in many parts of the system. All three types of AFS administrative privilege are required: membership in *system:administrators*, an entry in */usr/afs/etc/UserList*, and the `ADMIN` flag in the Authentication Database.

16.3. About Creating Local Password File Entries Using uss

In addition to the AFS-specific components of an account, a user must also have an entry in the local password file (*/etc/passwd* or equivalent) of each client machine he or she will log into. This applies at any of the three levels of account functionality mentioned in section 16.2.3 and described further in section 16.5.2.

The instructions for using **uss add** and **uss bulk** (in sections 16.6 and 16.8 respectively) advise you on creating a local password file entry for each user. They refer to section 16.5.11.5 where you can learn how to use the "E" template line to create entries in your cell's global password file (the one used for updating local password files, perhaps using **package**).

16.3.1. Assigning AFS and UNIX UIDs

For simplicity and ease of administration, you should make the UNIX UID and AFS UID match for each user in your cell. It is assumed throughout this guide that you have done so.

Being able to assign AFS UIDs explicitly is particularly useful if you want to use **uss** to convert existing UNIX accounts into AFS accounts, since you can set the AFS UID to match the existing UNIX UID. For more about handling UIDs when converting existing UNIX accounts, see section 16.4.1.

If your cell has existing UNIX accounts and you are creating an AFS account for a new user (who does not have an existing UNIX account), you are encouraged to allow the Protection Server to allocate an AFS UID automatically, rather than using **-uid**. However, you need to make sure that no AFS UIDs overlap with existing UNIX UIDs. See section 16.4.1.

If your cell does not have existing UNIX accounts to be converted, you can safely allow the Protection Server to allocate AFS UIDs starting at its default, or use **pts setmax** to designate any starting point you wish. You can also explicitly grant AFS UIDs using **-uid** with **uss add** or <*uid*> in a bulk input file, though this is not recommended.

16.3.2. Specifying Passwords in the Local Password File

If you do not use the AFS version of **login**, you must place an actual password in the local password file (*/etc/passwd* or equivalent) on each machine the user will log into. Non-AFS versions of **login** log issuers into the local machine's UNIX file system only. Users then need to issue **pagsh** and **klog** to authenticate with AFS. It is simplest if the passwords in */etc/passwd* and the Authentication Database are the same, but this is not required.

If you use the AFS version of **login**, the password field in a machine's password file is no longer the primary "gate" for access to your system. If the user provides the correct AFS password, then AFS **login** never consults the local password file. However, you can still use the password field to control access, in the following way:

- An asterisk (*) in this field prevents the user from either authenticating with AFS or logging into the machine UNIX file system. This is useful mainly in emergencies, when you want to prevent a certain user from logging into the machine.
- A string of any length other than the standard thirteen characters means that if the user does not provide the correct AFS password, he or she will not be allowed to log into the local machine's UNIX file system. This is appropriate if you want to make sure that only people with AFS accounts in your cell can log into your machines. Inserting a single "X" or other character is the simplest way to do this.
- A string of thirteen characters representing the encrypted form of a password set with **/bin/passwd** means that if the user does not provide the correct AFS password, he or she can still login to the local UNIX file system by providing the correct UNIX password.

For more details on the AFS version of **login**, see section 2.11.2 or the **login** page in the *AFS Command Reference Manual*.

16.4. About Using uss When Converting Existing UNIX Accounts

Many cells installing AFS for the first time have existing UNIX accounts which they wish to convert to AFS accounts. There are three main issues the administrators of these cells need to address. Sections 16.4.1 through 16.4.3 discuss them in detail:

- Making UNIX and AFS UIDs match. See section 16.4.1.
- Setting the password field in the local password file appropriately. See section 16.4.2.
- Moving UNIX files into AFS. See section 16.4.3.

16.4.1. Making UNIX and AFS UIDs Match

As mentioned in section 16.3.1, AFS users retain an entry in the local password file on client machines, and administration is much simpler when a user's UNIX UID and AFS UID are the same. You have two alternatives when converting UNIX accounts: you can either retain the existing UNIX UIDs (setting the new AFS UIDs to match), or change the UNIX UIDs to match the AFS UIDs.

16.4.1.1. Alternative 1: Retain UNIX UIDs

If you decide to retain the UNIX UIDs and make the AFS UIDs match, you must perform two tasks:

1. You need to use **-uid** argument available on **uss add** to assign an AFS UID that matches the existing UNIX UID, rather than allowing the Protection Database to assign one. Because the UNIX UID is not being changed, you may not need to alter the user's entry in the local password file.

   However, if you are using AFS **login**, you may wish to change the entry's password field; see section 16.3 to learn how the contents of this field affect the behavior of AFS **login**. If you do not change the entry, do not include in the template file the type of "E" line described in section 16.5.11.5.

2. You need to make sure that no new AFS UIDs overlap with existing UNIX UIDs. Therefore, before creating accounts for any new users (ones without existing UNIX accounts), you need to make sure that the `UserMax` counter is set higher than your cell's largest UNIX UID. You can examine the current value of `UserMax` with **pts listmax** and set it with **pts setmax**. Both commands are described in section 18.13.

16.4.1.2. Alternative 2: Change UNIX UIDs

If you decide to convert existing UNIX UIDs with new AFS UIDs, you can allow the Protection Server to allocate the AFS UIDs automatically. You should use an "E" line to create new */etc/passwd* entries.

There is one drawback to changing the UNIX UID: the owner of the original UNIX files and directories will no longer be computed correctly. However, because you will need to move these files and directories into AFS anyway, changing their owner at the same time may not be a problem. See section 16.4.3.

16.4.2. Setting the Password Field Appropriately

The user with an existing UNIX account already has an entry in the local password file, presumably with a (scrambled) password in the password field. What should appear in that field after the conversion to AFS depends on which version of **login** you are using:

- If you are not using the AFS version of **login**, an actual password must appear (in scrambled form) in the password field of the local password file on every machine the user will log into.
- If you are using AFS **login**, then what you put in the password field of the local password files controls some aspects of **login**'s behavior. Section 16.3 describes the choices and their effect.

If you want to place an actual password in this field, then you can use the "E" line as described in section 16.5.11.5 to place a dummy initial password in the field. Then issue **/bin/passwd** to replace the dummy with an actual (scrambled) UNIX password.

16.4.3. Moving UNIX files into AFS

Users with existing UNIX accounts presumably have files and directories in them, and you may wish to transfer these files into AFS. The easiest method is to move all relevant UNIX files and directories onto the local disk of an AFS client machine, and then move them from the local disk into the user's new AFS home directory (using **mv**). (An alternative is to move all the UNIX directories and files onto a single machine and then configure the machine as an AFS client, but you could run out of machines before accounts.)

If you chose to change the user's UNIX UID to match a new AFS UID (Alternative 2 in section 16.4.1), you must change the ownership of the former UNIX files and directories. Once you move elements into AFS, only members of *system:administrators* can issue **chown**. Remember, that in AFS only the "owner" mode bits are relevant, and their effect interacts with the access rights on the ACL of the file's parent directory. The setting of the "group" and "other" bits no longer protects a file from unauthorized access; instead the file needs to be placed in a directory with an equally protective ACL.

16.5. Constructing a Template for Account Creation

As detailed in section 16.2.3, it is easier to create user accounts with **uss** than with individual commands because **uss** allows you to predefine many aspects of account configuration in a template which you can use repeatedly. You construct this template before issuing **uss add** commands, which gives you time to decide on configuration details and edit any typing mistakes. This section defines the syntax and function of the various types of "instruction lines" that can appear in a template, and describes how to combine the lines to achieve different types of account configuration.

16.5.1. The Instruction Lines: A Functional Summary

Several types of instruction lines may appear in a template, each with a different function. A single letter at the start of the line identifies its type.

The following list summarizes the purpose of each type of template line, and refers you to a detailed description of the line's syntax and suggestions for its use. The lines are listed in the order that they should generally appear in the template file so that dependencies between the lines are handled properly. There is one exception: the "V" line is discussed before the "G" line, even though "G" lines should appear in the template first.

V Creates a volume and mounts it in the file system at a specified directory location (presumably the user's home directory) and sets the quota, owner, and ACL for the directory. This line must appear in any template that is not 0-length. See section 16.5.7.

G Names one of a set of existing directories in the file system. Its intended use is to distribute user home directories randomly but evenly into a number of "grouping" directories. The command interpreter selects the directory from the specified set that has the least number of entries as the value for the `$AUTO` variable. This variable normally appears in the "V" line's <*mount point*> field. See section 16.5.8.

D Creates a directory, generally a subdirectory of the new home directory, and sets its UNIX mode bits, owner, and ACL. See section 16.5.9.

F Creates a file by copying a prototype and sets its UNIX mode bits and owner. See section 16.5.10.

E Creates a file by "echoing" what appears on this line into the new file; sets the file UNIX mode bits and owner. See section 16.5.11.

L Creates a hard link. See section 16.5.12.

S Creates a symbolic link. See section 16.5.12.

A Specifies the level of password/authentication security. See section 16.5.13.

X Executes the indicated command. See section 16.5.14.

16.5.2. The Three Types of User Accounts

Section 16.2.3 mentioned briefly that **uss** allows you to create three types of accounts that provide three basic levels of functionality. This section explains more fully the practical effects of the three types, and how to construct a template to create each type.

16.5.2.1. Level 1: The Authentication-Only Account

When you issue the **uss add** command, you automatically create entries in the Protection Database and Authentication Database for the new user, resulting in assignment of a user name, AFS UID, and password. These are the minimal components of an AFS user account. These components allow the user to authenticate in AFS and to issue commands (including privileged ones if the necessary permissions are granted). This level of functionality is of little use to a regular user because without a volume or home directory a user cannot create files of his or her own. It is useful for creating administrative accounts and for enabling users from foreign cells to access information as locally authenticated users.

To create this type of account, build the template as a 0-length file.

16.5.2.2. Level 2: The Basic Account

The Basic Account contains a volume for the user, in addition to Authentication and Protection Database entries. The volume is mounted in the file system at the user's home directory, a space quota is set for it, and the initial entries on its access control list are defined. The user can then create and store files in his or her own volume.

To create this type of account, put a single "volume-creation" ("V") line in the template. You may include "G" lines, if appropriate (see section 16.5.8).

16.5.2.3. Level 3: The Full Account

A Basic Account is completely functional, but may not be as convenient and useful as one that comes equipped with directories and configuration files that help provide some basic functions, such as logging in, printing, and mail delivery. Section 2.9.8 suggests some of the appropriate elements to create.

Several lines in the template can help you create the necessary subdirectories, files, and links to create a Full Account:

- To create subdirectories in a new user home directory, put a "D" line in the template for each subdirectory.
- To create files in the home directory or subdirectories of it, put an "F" line or "E" line in the template for each file.
- To create links between files found in the home directory or subdirectories of it and those in other directories, put an "L" or "S" line in the template for each link.

16.5.3. Setting Template Values with Variables and Constants

Each type of instruction line has several fields where you define exactly the characteristics of the element created by that line. The "D" line, for instance, has fields that define the directory's pathname, owner, mode bits, and access control list.

You can place three types of values in a field: a variable, a constant, or a combination of the two. What you place in a field depends on what you want to achieve and affects what you type on the **uss add** command line.

16.5.3.1. Using Constants

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, you may wish to impose a space quota of 2000 one-kilobyte blocks on every user's volume. To achieve this, put the string `2000` in the <*quota*> field on the "V" line.

16.5.3.2. Using Variables

If an aspect of account configuration varies from user to user, put a variable in the appropriate field. You then must provide a value for the variable when creating each account. There is a predefined set of variables available for use in a template, each with a corresponding source (or set of sources) for values, most commonly an argument on the **uss add** command line.

One example use of a variable is for defining the partition housing the user's volume, which often varies from user to user. You could put the `$PART` variable in the <*partition*> field on the "V" line, and then provide the desired value for each user with the **-partition** argument on the **uss add** command line.

Table 1 summarizes all the variables and the source of values for each one. Sections 16.5.7 through 16.5.14 explain the intended uses of the variables. These are the only variables you can use in a template—you cannot make up your own; however, the predefined "number" variables are adaptable to nearly any purpose. See section 16.5.3.4 for a discussion of their use. If the source of value for a variable is shown as an argument, it means that the issuer must provide a value for the variable by means of that command-line argument.

Variable	Source for value
`$USER`	**-user** <*login name*>
`$NAME`	**-realname** <*full name in quotes*> if not provided, defaults to name provided by **-user** <*login name*>
`$SERVER`	**-server** <*FileServer for home volume*>
`$PART`	**-partition** <*FileServer's disk partition for home volume*>
`$MTPT`	**-mount** <*home directory mount point*> for occurrence in "V" line; value of "V" line <*mount point*> field for subsequent occurrences
`$1` through `$9`	**-var** <*auxiliary argument pairs(Num val)*>
`$UID`	AFS UID allocated by Protection Server or **-uid** <*uid to assign the user*>
`$AUTO`	Previous "G" lines in template
`$PWEXPIRES`	**-pwexpires** <*password expires in [0..254] days (0 => never)*>

Table 1: The Template Variables and Sources for Values

16.5.3.3. Combining Constants and Variables

If an aspect of account configuration is partly constant and partly variable, you can combine variables and character strings in a single field, providing a value for the variable just as with solitary variables. You can also put more than one variable in a single field.

One example use of a variable/string combination is in defining the mount point for a volume. For instance, user volumes are often mounted in the */afs/***cellname***/usr* directory under the user's user name, which corresponds to the variable `$USER`. Thus administrators in the Transarc Corporation cell would put

```
/afs/transarc.com/usr/$USER
```

in the "V" line's <*mount point*> field, and provide a value for `$USER` with the **-user** argument on the **uss add** command line.

16.5.3.4. A Note on the Number Variables

While the predefined variables may not meet your needs completely, the number variables `$1` through `$9` are adaptable to any purpose. If you place one or more of these variables in the template, provide a value for each one using the **-var** argument. This argument is unusual in that each instance for it has two parts—the number index and the value, separated by a space.

For example, suppose that an administrator in the Transarc Corporation cell wants to vary the value of the *<quota>* field in the "V" line. (Section 16.5.7 explains the syntax of the "V" line, if you are not familiar with it.) To vary this value, he uses the `$1` number variable on the "V" line. (He could in fact use any of the nine available variables, in any order.) The "V" line would look like this:

```
V user.$USER $SERVER.transarc.com $PART $1
        /afs/transarc.com/usr/$USER $UID $USER all
```

Then, to specify a quota of 5000 kilobyte blocks, the administrator would include the following on the **uss add** command line:

-var 1 5000

Sections 16.5.7.4 and 16.5.7.5 present other examples of the use of a number variable.

16.5.4. Where to Place Template Files

A template must be available when the **uss** command interpreter tries to execute a **uss add** command. Even if you only create Protection and Authentication Database entries, the template file must be present (though it may be a 0-length file).

By default, the command interpreter searches through the following directories, in the order shown, for a template called *uss.template*:

1. The current working directory.
2. */afs/**cellname**/common/uss* where ***cellname*** is the local cell.
3. */etc*

However, you can use templates with different names and/or locations in other directories by using the **-template** argument on the **uss add** command line. If you provide only a template filename, the command interpreter looks for it in the listed directories. If you provide either a full or partial pathname, it looks only in the specified directory, interpreting a partial pathname relative to the current working directory.

16.5.5. Some General Rules for Constructing a Template

If a variable takes its value from an element elsewhere within the template, then the definition must precede the reference. Putting the instruction lines in the following order should result in correct resolution of variables:

G V D F E L S A X

The fields on each line must appear in the order specified in the definitions in sections 16.5.7 through 16.5.14. It is not permissible to skip a field. Separate each field from its neighbors with a space.

Specify full pathnames when they appear in the template. Incomplete paths will be interpreted relative to the directory from which the command is issued, with potentially unpredictable results.

Do not break an instruction line into more than one line in the template; you may insert carriage returns only at the end of each line. Some example instruction lines appear in this document on more than one line, but that is only for legibility reasons.

Provide a value for every variable that appears in the template; failure to do so results in failure of the entire **uss add** command. Providing a command-line argument when the corresponding variable does not appear in the template has no effect since there is no place for the argument's value to register. The value in the template (presumably a constant) is used instead.

The command interpreter treats lines beginning with the number sign (#) as comments and ignores blank lines completely.

16.5.6. Some Example Templates

This section provides some examples uses of the template file. An example is given for each of the major types of accounts.

16.5.6.1. Some simple template examples

The simplest type of template file is an empty file. A template file of this type is used to create Authentication-only Accounts.

The next simplest type of template file has a "V" line (and possibly, a "G" line) in it. A template file of this type is used to create Basic Accounts. In brief, the fields in the "V" line define the volume name, file server machine, partition, quota, mount point, owner, and access control list. In the Transarc Corporation cell, a template of this type might look like this:

```
G /afs/.transarc.com/usr1
G /afs/.transarc.com/usr2

V user.$USER $SERVER.transarc.com /vicep$PART 5000 $AUTO/$USER
 $UID $USER all transarc:staff rl
```

If you used a template file like this, you would need to define a value for each variable: the user name for `$USER`, the unique part of the file server machine name for `$SERVER`, and the partition for `$PART`, when issuing the **uss add** command. The Protection Server automatically assigns an AFS UID for `$UID`.

16.5.6.2. A complicated template example

A more complicated template file might contain any or all of the different instruction lines, in addition to the required "V" line. A template file of this type is typically used to setup Full Accounts for users.

The following example template might be used to set up Full Accounts for users in the Transarc Corporation cell. This example is referred to throughout the explanation of the template lines in sections 16.5.7 through 16.5.14.

```
#
# Specify the available grouping directories
#
G /afs/.transarc.com/usr1
G /afs/.transarc.com/usr2
#
# Create the user's home volume
#
V   user.$USER $SERVER.transarc.com /vicep$PART 5000 $AUTO/$USER
 $UID $USER all transarc:staff rl
#
# Create directories and files for mail
#
D $MTPT/.MESSAGES 0700 $UID $USER all transarc:staff none
D $MTPT/.Outgoing 0700 $UID $USER rlidwk postman rlidwk
D $MTPT/Mailbox 0700 $UID $USER all transarc:staff none
 system:anyuser lik
#
# Here are some useful scripts for login etc.
#
F $MTPT/.Xbiff 0755 $UID /afs/transarc.com/admin/user/proto
F $MTPT/.Xresources 0644 $UID /afs/transarc.com/admin/user/proto
F $MTPT/.Xsession 0755 $UID /afs/transarc.com/admin/user/proto
F $MTPT/.cshrc 0755 $UID /afs/transarc.com/admin/user/proto
F $MTPT/.login 0755 $UID /afs/transarc.com/admin/user/proto
F $MTPT/.logout 0755 $UID /afs/transarc.com/admin/user/proto
F $MTPT/.twmrc 0644 $UID /afs/transarc.com/admin/user/proto
F $MTPT/preferences 0644 $UID /afs/transarc.com/admin/user/proto
#
# Make a passwd entry
#
E /afs/.transarc.com/common/etc/newacct/passwd_$USER 0644 root
 "$USER:X:$UID:11:$NAME:$MTPT:/bin/csh"
#
# Put in the standard password/authentication checks
#
A $USER 250 noreuse 5 10
#
# Create a mount point to a public volume for usr
#
X "create_public_mount $AUTO $USER"
#
# Here we set up the symbolic link to public directory
#
S /afs/transarc.com/public/$USER $MTPT/public
```

16.5.7. Creating a Volume with the "V" Line

Unless the template is a 0-length file, a "V" line must appear in it; however, a template may contain only one "V" line which should be used to setup a user's home volume. (To create other volumes for a user, within the template file, use the "X" line.)

Note: If the `$AUTO` variable appears on the "V" line, then the "G" lines that provide values for it must appear above the "V" line.

The "V" line has several functions. Use it to

- Create a volume and VLDB entry with the indicated volume name.
- Place the volume at a specified site (file server machine and partition).
- Mount the volume in the file system at the specified directory location (presumably the user's home directory).
- Set the volume's space quota.
- Set the home directory's owner (presumably the user).
- Set the home directory's access control list (should at least give full rights to the user).

The following should appear in the template as a single line:

`V` <*volume name*> <*server*> <*partition*> <*quota*> <*mount point*> <*owner*> <*access list*>

Below is a "V" line from the example in section 16.5.6.2:

```
V  user.$USER $SERVER.transarc.com /vicep$PART 5000 /afs/.transarc.com
 $AUTO/$USER $UID $USER all transarc:staff rl
```

The capital letter `V` tells the command interpreter that this is a volume-creation instruction.

16.5.7.1. The <*volume name*> Field and $USER Variable

The <*volume name*> field defines the volume's name as it will be listed in the VLDB. The name of a user's volume usually includes the user's user name. The `$USER` variable may therefore appear as part of the name in this field, and the appropriate user name specified with the **-user** argument of the **uss add** command.

In the Transarc example (see section 16.5.6.2), the system administrator used the recommended form `user.`***user name***. So the field reads

```
user.$USER
```

To create a volume for the user with user name *smith*, a Transarc system administrator would include **-user smith** on the **uss add** command line.

16.5.7.2. The <*server*> Field and $SERVER Variable

The <*server*> field defines the file server machine that will house the volume. It is best to provide a complete Internet-style host name in this field. Shortened forms are allowed, but their correct resolution depends on the state of the cell's naming service when the **uss add** command is issued.

The $SERVER variable, which is mapped to the **-server** argument, is often used in this field. One easy way to provide a complete name (without having to type it all on the command line) is to combine a constant and the $SERVER variable. Specifically, the constant can specify the part of the complete name that is common to all the file server machines, while the variable specifies the variant part. Then you type only the shorter, variant part on the command line.

In the Transarc example (see section 16.5.6.2), the system administrator used this type of combination of variable and constant. (All of the file server machines in the Transarc cell share the *.transarc.com* suffix in their Internet host name.) So the field reads

```
$SERVER.transarc.com
```

Then, to assign a new user's volume to "fs1.transarc.com," the administrator includes **-server fs1** on the **uss add** command line.

16.5.7.3. The <*partition*> Field and $PART Variable

The <*partition*> field defines which partition on the file server machine will house the volume. Normally, the partition differs from user to user, so the $PART variable goes in this field and you provide a value with the **-partition** argument.

As discussed on page xvii of the introductory *About This Manual* chapter, the standard form for partition names is */vicep***x**, but the following variants are acceptable:

/vicepa = vicepa = a = 0

/vicepb = vicepb = b = 1

and so on up to

/vicepz = vicepz = z = 25

In the Transarc example (see section 16.5.6.2), the system administrator used a recommended form so that the system administrator need only type a single letter when specifying the $PART variable with the **-partition** argument of the **uss add** command. So the field reads

```
/vicep$PART
```

16.5.7.4. The <*quota*> field and the Number Variables

The <*quota*> field defines the maximum number of one-kilobyte disk blocks the volume will be allowed to occupy. There is no predefined variable for setting quota, because most cells use a constant value for all user volumes. However, if you want to vary quota size, place one of the number variables ($1 through $9) in this field. For example, if you put $3 in this field then you could grant a 1500 block quota by including **-var 3 1500** as a command line argument.

In the Transarc example (see section 16.5.6.2), the system administrator gives a 5000 kilobyte initial quota to new users. So the field reads

```
5000
```

16.5.7.5. The <*mount point*> Field and the $USER, Number, $AUTO, and $MTPT Variables

The <*mount point*> field defines where to mount the volume in the file system and the name and location of the user's home directory. Another function of the <*mount point*> field is to provide a value for any occurrences of the `$MTPT` variable in lines that follow the "V" line in the template.

One convention used by most cells is to name home directories with the user name, so the `$USER` variable often appears as the last element in this field. Otherwise, the contents of this field vary more from cell to cell than any other field, because different cells choose to group user home directories in different ways. Whichever scheme you choose, you need to create the "grouping directories" (in which you are going to mount user volumes) with **mkdir** before including them in the template.

There are several common grouping possibilities, each achievable with a different combination of variables and constants:

- Put all user home directories in a single directory with a name like */afs/*<***cellname***>*/usr* is an AFS-appropriate variation on standard UNIX practice. This is most appropriate for a small cell, where the number of accounts is small and only the name of the home directory varies from user to user, so the <*mount point*> field needs to contain only one variable (normally, `$USER`). (This kind of grouping could also be achieved with the `$MTPT` variable; see the final bulleted paragraph in this list.)
- Distribute home directories randomly but evenly into more than one grouping directory. The grouping does not reflect any real-world divisions. This is especially appropriate in large cells, where the number of home directories is large enough to slow down directory lookup significantly if they all reside together in one grouping directory, but distribution according to real-world divisions is not feasible.

  The `$AUTO` variable is predefined for evenly dividing home directories into a specified set of grouping directories. It takes its value from the template "G" lines, which must appear in the template before the "V" line. See section 16.5.8 to learn how to use the "G" line and `$AUTO` variable in this field.
- Distribute user home directories into multiple directories that reflect real-world divisions, such as academic or corporate departments. You can accomplish this with the "number" variables, `$1` through `$9`. (This kind of grouping could also be achieved with the `$MTPT` variable; see the last bulleted paragraph in this list.)

  As an example, suppose a Transarc Corporation cell administrator establishes group directories called *usr/marketing*, *usr/research*, *usr/finance* and so on, to correspond to divisions in the company. The administrator could put the following in the <*mount point*> field:

  ```
  /afs/transarc.com/usr/$1/$USER
  ```

Then, to mount a new volume in the *usr/finance* directory, the administrator includes **-var 1 finance** as an argument on the **uss add** command line.

Note: The number variable in the <*mount point*> field does not have to be `$1`; any of `$1` through `$9` will give the same result. Similarly, use of the number variables is not limited to this field.

- Distribute home directories into alphabetic subdirectories of *usr* (*usr/a*, *usr/b* and so on), based on the first letter of the user name. The advantage is that anyone who knows the user's user name can find the user's home directory. One potential drawback to this scheme is that the distribution is not likely to be even—many more names start with the letter "s" than the letter "q," for instance. And users whose names begin with more commonly used letters may experience slower lookup if the number of users is sufficiently large.

  One way to implement this scheme would be to include a number variable in the <*mount point*> slot as in the following example. When creating the account, the administrator would supply the appropriate letter as a value for the variable.

  ```
  /afs/transarc.com/usr/$3/$USER
  ```

- Use the `$MTPT` variable to implement the first, third, and fourth grouping schemes discussed here, as well as others. A potential advantage of using `$MTPT` is that you can decide where to mount the volume as you issue the **uss add** command. However, it also means you have to type more, providing a complete pathname for the **-mount** command-line argument.

  As an example, suppose a Transarc Corporation administrator has put the `$MTPT` variable alone in the <*mount point*> field and wants to mount a new user volume as */afs/transarc.com/usr/smith*. He would have to type that entire pathname as the instance of **-mount** argument on the **uss add** command line.

  He could also combine `$MTPT` with a constant, as in

  ```
  /afs/transarc.com/$MTPT
  ```

  in which case he would need to type only **usr/smith** as the **-mount** argument.

  Note: The `$MTPT` variable is unusual in that it takes its value from two different sources, depending on where it appears in the template. Occurrences of `$MTPT` in or before the <*mount point*> field on the "V" line take their value from the **-mount** argument on the **uss add** command line. Occurrences of `$MTPT` after the "V" line's <*mount point*> field take their value from that field. This means that occurrences of `$MTPT` after that field will take their value directly from the **-mount** command-line argument only if the `$MTPT` variable appears by itself in the <*mount point*> field.

In the Transarc example (see section 16.5.6.2), the system administrator uses the `$AUTO` and `$USER` variable to randomly divide user accounts into the directories */afs/.transarc.com/usr1* and */afs/.transarc.com/usr2*. So the field reads

```
$AUTO/$USER
```

and the G lines that will be substituted for the `$AUTO` variable are

```
G /afs/.transarc.com/usr1
G /afs/.transarc.com/usr2
```

16.5.7.6. The <*owner*> field and the $UID Variable

The <*owner*> field defines the owner of the (home directory) mount point named in the <*mount point*> field. You should make the new user the owner.

This field must contain a UNIX UID to comply with UNIX requirements. For new users, put the `$UID` variable here (and set the user's UNIX UID to match this AFS UID). If you are converting existing UNIX accounts using **uss**, see section 16.4.

In the Transarc example (see section 16.5.6.2), the system administrator uses the recommended form. So the field reads

```
$UID
```

16.5.7.7. The <*access list*> field

The <*access list*> field defines the access control list for the new home directory. The list may define one or more pairs, separated from one another by a space. Each pair in the list specifies

- A user name or Protection Database group name (it must already exist in the Protection Database).
- The access rights.

These are separated by a space. See the **fs setacl** command to learn about the access rights available with AFS.

The new user must be given all access rights for minimal use which could be achieved with

```
$USER all
```

The pairs you specify in this field are added to a default pair that grants *system:administrators* all rights. Do not attempt to grant any rights to the issuer of the **uss** command. As the last step in account creation, the **uss** command interpreter automatically deletes that person from any access control lists set during the creation process.

In the Transarc example (see section 16.5.6.2), the system administrator gives each user all rights and members of the group *transarc:staff* read and lookup rights. So the field reads

```
$USER all transarc:staff rl
```

16.5.8. Evenly Distributing User Home Directories with the "G" Line

The discussion in section 16.5.7.5 mentioned that some cells distribute user home directories into several grouping directories. This is most useful for large cells, where the number of home directories is large enough to cause slowed lookup if they all reside together in one directory. While one solution to this problem is to group directories according to real-world divisions, such as academic or corporate departments, this is not always feasible. If the groups are unequal in size, and some of them large enough to cause slowed lookup, then members of larger groups are unfairly penalized. Such groupings are also inconvenient if people commonly move around among the real-world groups.

An alternative to real-world divisions is a random but even distribution of user home directories into multiple grouping directories. You can achieve this automatically by placing the `$AUTO` variable in the "V" line's <*mount point*> field.

To define the possible values for the `$AUTO` variable, put each one on a separate "G" (for "group") line in the template before the `$AUTO` variable. When the **uss** command interpreter encounters the `$AUTO` variable in a field, it compares all the directories named on "G" lines and selects the one with the fewest entries in it to become the value of `$AUTO`.

The syntax for the "G" line is

`G` <*pathname*>

where *pathname* specifies either a complete directory pathname or only the final element (the directory itself), depending upon the value of <*mount point*>.

Below are "G" lines from the example in section 16.5.6.2:

```
G /afs/.transarc.com/usr1
G /afs/.transarc.com/usr2
```

In this example, the system administrator uses the "G" line to distribute user volumes evenly into */afs/.transarc.com/usr1* and */afs/.transarc.com/usr2*.

If the system administrator at Transarc set the <*mount point*> field to the value `/afs/.transarc.com/$AUTO/user.$USER`, he could have defined the "G" lines as follows:

```
G usr1
G usr2
```

Note: Even though the `$AUTO` variable's intended use is in the "V" line's <*mount point*> field, it can appear in any field on any line that includes a directory name (normally, as part of a pathname). The behavior is the same in all cases: the command interpreter selects the least full directory named on a "G" line as the value for `$AUTO`.

16.5.9. Creating Directories with the "D" Line

The "D" line creates a directory anywhere in the file system. Its intended use is creation of subdirectories in the new user's home directory. The template file should include a "D" line for each subdirectory to be created. "D" lines should appear in the template after the "V" line.

It is possible, though not recommended, to use the "D" line to create directories outside AFS (i.e., in the UNIX file system of the workstation where the **uss** command is being issued). Doing so introduces two complications, explained in sections 16.5.9.3 and 16.5.9.4. The preferred method for automatic placement of directories on the local disk is the **package** program; see chapter 14.

The following should appear in the template as a single line:

D <*pathname*> <*mode*> <*owner*> <*access list*>

Below is a "D" line from the example in section 16.5.6.2:

```
D $MTPT/Mailbox 0700 $UID $USER all transarc:staff none
 system:anyuser lik
```

This particular example from section 16.5.6.2 creates a subdirectory called *Mailbox* in the user’s home directory.

The capital letter D tells the command interpreter that this is a directory-creation instruction.

16.5.9.1. The <*pathname*> Field and $MTPT Variable

The <*pathname*> field defines the full pathname of the directory to create. Most often, the "D" line creates a subdirectory of the newly created home directory, so the first part of the pathname should refer to the directory where you mounted the user’s volume. An easy way to specify the home directory pathname is with the $MTPT variable, which takes its value from the <*mount point*> field on the "V" line. (This dependency is the main reason that "D" lines should appear after "V" lines.)

In the Transarc example (see section 16.5.6.2), the system administrator identifies the pathname to the *Mailbox* subdirectory by using the $MTPT variable, so the field reads

```
$MTPT/Mailbox
```

If the system administrator chose to use the $USER variable rather than $MTPT, the field might have read

```
/afs/transarc.com/usr/$USER/Mailbox
```

16.5.9.2. The <*mode*> Field

The <*mode*> field sets the UNIX mode bits for the directory. You must provide a four-digit decimal number in this field; but keep in mind that when calculating privilege, AFS only checks the three "user" bits (in tandem with the access control list). Keep in mind also that the "user" x bit must be turned on for a directory.

Suitable values for this field are

`0755` (equivalent to `rwxr-xr-x`)

`0744` (equivalent to `rwxr--r--`)

In the Transarc example (see section 16.5.6.2), the system administrator uses the value `0700` for the *Mailbox* subdirectory (equivalent to `rwxr-----`).

16.5.9.3. The <*owner*> field and $USER or $UID Variable

The <*owner*> field defines the owner of the new directory. Specify the new owner here. This field must contain a UID to comply with UNIX requirements. If the directory is being placed in AFS for a new user, put the `$UID` variable here (and set the user's UNIX UID to match this AFS UID). If converting existing UNIX accounts using **uss**, see section 16.4.

If the directory is being placed in the UNIX file system instead of AFS, then it is simplest if this field contains the user name or UNIX UID of the **uss** command's issuer. A complication arises if the issuer wishes to set the <*owner*> field to another value. The default owner of a UNIX file system directory is the user who created it (in this case, the issuer of the **uss** command). Changing this default, which **uss** will attempt if anything other than the issuer's name appears in the <*owner*> field, requires issuing the **chown** command, which in turn requires being the super-user "root." If the issuer wishes to specify an alternate owner, he or she must be "root" in the UNIX file system and use the **-admin** flag on **uss add** and **uss bulk** in order to be recognized in AFS as a privileged administrator. For more on the **-admin** argument, see section 16.2.7.

In the Transarc example (see section 16.5.6.2), the system administrator gives ownership of the *Mailbox* subdirectory to the owner. So the field reads

```
$UID
```

16.5.9.4. The <*access list*> field

The <*access list*> field defines the access control list for the new directory. The list may define one or more pairs, separated from one another by a space. Each pair in the list specifies:

- A user name or Protection Database group name (it must already exist in the Protection Database).
- The access rights.

Each element is separated by a space. See the **fs setacl** command to learn about the access rights available with AFS.

The new user should be given all access rights, as specified in the following example:

```
$USER all
```

The directories created with a "D" line have no default ACL entries; however, the directory created inherits the ACL of its parent directory. As with the "V" line, you should not attempt to grant any rights to the issuer of the **uss** command. In the last step in account creation, the **uss** command interpreter automatically deletes that person from any access control lists set during the creation process.

If you are placing this directory in the UNIX file system, **uss** will always print a warning message because it is not possible to associate an ACL with a non-AFS directory. It will create the directory nonetheless, and the <*access list*> field must always be provided.

In the Transarc example (see section 16.5.6.2), the system administrator gives ALL rights to the user; no rights to members of *transarc:staff*; and LOOKUP, INSERT, and LOCK rights to members of *system:anyuser*. So the field reads

```
$USER all transarc:staff none system:anyuser lik
```

The purpose of giving such extensive rights to members of *system:anyuser* is so that any system user (including a mail delivery daemon) will be allowed to drop mail into the *Mailbox* directory. Note, however, that members of *system:anyuser* will not be allowed to read the mail files.

16.5.10. Creating Files from a Prototype with the "F" Line

The "F" line creates a file anywhere in the file system by copying the contents of an existing "prototype" file into the new file. Its intended use is creation of standard files in the new user’s home directory and its subdirectories. The template should include one "F" line for each file to be created from a prototype. "F" lines should appear in the template after the "D" lines that define the subdirectories where they reside.

It is possible, though not recommended, to use the "F" line to create a file outside AFS (i.e., on the local disk of the workstation where the **uss** command is being issued). Doing so introduces the complication discussed in section 16.5.10.3. The preferred method for automatic placement of files on the local disk is the **package** program, which runs as "root." See chapter 14 for details.

The following should appear in the template as a single line:

F <*pathname*> <*mode*> <*owner*> <*prototype*>

Below is a "F" line from the example in section 16.5.6.2:

```
F $MTPT/.login 0755 $UID /afs/transarc.com/admin/user/proto
```

This example creates a *.login* file in the user’s home directory.

The capital letter `F` tells the command interpreter that this is a file-creation instruction.

16.5.10.1. The <*pathname*> Field and $MTPT Variable

The <*pathname*> field defines the complete pathname of the file to create. Files such as these usually reside in the user's home directory or a subdirectory of it, so the pathname should include a reference to the directory where you mounted the user's volume. An easy way to specify the home directory pathname is with the `$MTPT` variable, which takes its value from the <*mount point*> field on the "V" line.

In the Transarc example (see section 16.5.6.2), the system administrator uses the `$MTPT` variable to define the placement of the *.login* file. So the field reads

```
$MTPT/.login
```

If the Transarc system administrator chose not to use the `$MTPT` variable, he might have set up the field as follows:

```
/afs/transarc.com/usr/$USER/.login
```

16.5.10.2. The <*mode*> Field

The <*mode*> field sets the UNIX mode bits for the directory. Provide a four-digit decimal number in this field, but keep in mind that when calculating privilege AFS only checks the three "user" bits (in tandem with the access control list).

Common values for this field are

`0666`	(equivalent to `rw-rw-rw-`)
`0644`	(equivalent to `rw-r--r--`)
`0444`	(equivalent to `r--r--r--`)

In the Transarc example (see section 16.5.6.2), the system administrator uses the value `0755` for the *.login* file (equivalent to `rwxr-xr-x`).

16.5.10.3. The <*owner*> field and the $UID Variable

The <*owner*> field defines the owner of the new file. Use the new user's name for this value. This field must contain a UID to comply with UNIX requirements. If the file is being placed in AFS for a new user, put the `$UID` variable here (and set the user's UNIX UID to match this AFS UID). If converting existing UNIX accounts using **uss**, see section 16.4.

If the file is being placed in the UNIX file system instead of AFS, then this field should contain the user name or UNIX UID of the **uss** command's issuer. A complication arises if the issuer wishes to set the <*owner*> field to another value. The default owner of a UNIX file system directory is the user who created it (in this case, the issuer of the **uss** command). Changing this default, which **uss** will attempt if anything other than the issuer's name appears in the <*owner*> field, requires issuing the **chown** command, which in turn requires being the super-user "root." If the issuer wishes to specify an alternate owner, he or she must be "root" in the UNIX file system and use the **-admin** flag on **uss add** and **uss bulk** in order to be recognized in AFS as a privileged administrator. For more on the **-admin** argument, see section 16.2.7.

In the Transarc example (see section 16.5.6.2), the system administrator gives ownership of the *.login* file to the owner. So the field reads

```
$UID
```

16.5.10.4. The <*prototype*> Field

The <*prototype*> field specifies the complete pathname of the file whose contents are to be copied into the new file. It usually resides in AFS, but may reside on the local disk of the machine on which you are issuing the **uss** command.

In the Transarc example (see section 16.5.6.2), the system administrator stores prototype files in */afs/transarc.com/admin/user/proto*. So the field reads

```
/afs/transarc.com/admin/user/proto
```

16.5.11. Creating One-Line Files with the "E" Line

The "E" line creates a file anywhere in the file system by echoing what appears in its <*contents*> field into the new file. Like the "F" line, its intended use is creation of files in the new user home directory or its subdirectories (especially for the creation of password file entries).

The "E" line's advantage over the "F" line is that variables may appear in the <*contents*> field, which 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" line are the same for everyone.) The <*contents*> field may contain spaces, but if it does the field must be surrounded by double quotes (" "); no carriage returns can appear in this field. "E" lines should appear in the template after the "D" lines that create the directories in which the files reside.

It is possible, though not recommended, to use the "E" line to create a file outside AFS (i.e., in the UNIX file system of the client machine where the **uss** command is being issued). Doing so introduces the complication discussed in section 16.5.11.3. The preferred method for automatic placement of files on the local disk is the **package** program, which runs as "root." See chapter 14.

The following should appear in the template as a single line:

```
E <pathname> <mode> <owner> "<contents>"
```

Below is a "E" line from the example in section 16.5.6.2:

```
E /afs/.transarc.com/common/etc/newacct/passwd_$USER 0644 root
 "$USER:X:$UID:11:$NAME:$MTPT:/bin/csh"
```

This example creates a file called *passwd_**username*** in the directory */afs/transarc.com/common/etc/newaccts* (which must exist already). The file is suitable for incorporation into the cell's global */etc/passwd* file. Section 16.5.11.5 suggests ways of incorporating this line into the global */etc/passwd*.

Note: If you do not use the AFS version of **login**, you need to use **/bin/passwd** to replace the `X` with an actual (scrambled) password.

For the user Joe L. Smith with user name *smith*, the file created from this line would be called */afs/transarc.com/common/etc/newaccts/passwd_smith* and might contain the following line:

```
smith:X:1205:11:Joe L. Smith:/afs/transarc.com/usr/usr1/smith:/bin/csh
```

The capital letter `E` tells the command interpreter that this is a file-creation instruction.

16.5.11.1. The <*pathname*> Field and $MTPT Variable

The <*pathname*> field defines the complete pathname of the file to create. If files created by "E" lines are to reside in the user's home directory or a subdirectory, the pathname should include a reference to the directory where you mounted the user's volume. An easy way to specify the home directory pathname is with the `$MTPT` variable, which takes its value from the <*mount point*> field on the "V" line. However, the "E" line files can be placed anywhere in the file system, not just in a user's home directory.

In the Transarc example (see section 16.5.6.2), the system administrator places the files created by the "E" line into the */afs/.transarc.com/common/etc/newacct/passwd_$USER*. So the field reads

```
/afs/.transarc.com/common/etc/newacct/passwd_$USER
```

16.5.11.2. The <*mode*> Field

The <*mode*> field sets the UNIX mode bits for the directory. Provide a four-digit decimal number in this field, but remember that when calculating privilege AFS only checks the three "user" bits (in tandem with the access control list).

Common values for this field are

`0666` (equivalent to `rw-rw-rw-`)

`0644` (equivalent to `rw-r--r--`)

`0444` (equivalent to `r--r--r--`)

In the Transarc example (see section 16.5.6.2), the system administrator assigns the value `0644` for the file */afs/.transarc.com/common/etc/newacct/passwd_$USER* (equivalent to `rwxr-xr-x`).

16.5.11.3. The <*owner*> field and the $UID Variable

The <*owner*> field defines the owner of the new file. You should make this the new user if the file is to reside in the new home directory. This field must contain a UID to comply with UNIX requirements. If the file is being placed in AFS for a new user, put the `$UID` variable here (and set the user's UNIX UID to match this AFS UID). If converting existing UNIX accounts using **uss**, see section 16.4.

Exception: If this is a file to be created in an AFS directory and then incorporated into global and local versions of */etc/passwd* (as described in section 16.5.11.5 below), set the owner to "root."

If the file is being placed in the UNIX file system instead of AFS, then this field should contain the user name or UNIX UID of the **uss** command's issuer. A complication arises if the issuer wishes to set the <*owner*> field to another value. The default owner of a UNIX file system directory is the user who created it (in this case, the issuer of the **uss** command). Changing this default, which **uss** will attempt if anything other than the issuer's name appears in the <*owner*> field, requires issuing the **chown** command, which in turn requires being the super-user "root." If the issuer wishes to specify an alternate owner, he or she must be "root" in the UNIX file system and use the **-admin** flag on **uss add** and **uss bulk** in order to be recognized in AFS as a privileged administrator. For more on the **-admin** argument, see section 16.2.7.

In the Transarc example (see section 16.5.6.2), the system administrator gives ownership of the */afs/.transarc.com/common/etc/newacct/passwd_$USER* file to root. So the field reads

```
root
```

16.5.11.4. The "<*contents*>" Field

The <*contents*> field specifies the string which becomes the contents of the new file. The string may contain template variables, which the command interpreter will resolve before creating the file. If it contains spaces, surround the entire contents with double quotes (" ").

In the Transarc example (see section 16.5.6.2), the system administrator creates an entry for the password file */afs/.transarc.com/common/etc/newacct/passwd_$USER*. So the field reads

```
$USER:X:$UID:10:$NAME:$MTPT:/bin/csh
```

where the user name provided with **-user** will replace `$USER`, the AFS UID assigned by the Protection Server will replace `$UID`, the user's real name provided as **-realname** will replace `$NAME` and the value from the "V" line <*mount point*> field will replace `$MTPT`.

16.5.11.5. Using the "E" Line to Create a Global */etc/passwd*

This section shows how you might use an "E" line in the **uss** template to create an entry suitable for incorporation into your cell's global password file (global */etc/passwd* or equivalent). This is useful because users will be unable to login to any machine that does not have an entry for them in its */etc/passwd*. You should direct the files created by such "E" lines into a directory dedicated to them, similar to the example */afs/transarc.com/common/etc/newaccts*.

After **uss add** creates one of these one-line files, it needs to be incorporated into the */etc/passwd* file on all of the client machines that the user might log into. One easy way to achieve this is to incorporate the files into a global */etc/passwd* kept in AFS and then use **package** to distribute the updated global version to client machines. See chapter 14 to learn about **package**.

A shell script is probably the easiest way to incorporate these one-line files into a global */etc/passwd*. Two sample shell scripts appear here. To automate the process even further, you could set up a "cron" process in a file server machine's */usr/afs/local/BosConfig* to execute the shell script, perhaps each day at a given time; see section 4.2.2.3 for details.

Note: The following examples are intended solely to demonstrate how you might incorporate new entries into a global */etc/passwd* automatically. If you choose to copy them directly, you do so at your own risk; their correctness or appropriateness to your situation is in no way guaranteed. You are urged to test your own shell script before actually running it.

Example C-Shell Script

The first example is a simple C shell script suitable for the Transarc Corporation cell. It incorporates the individual files found in */afs/transarc.com/common/uss/newaccts* into a new version of the global *passwd* file found in */afs/transarc.com/common/etc*, 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/transarc.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 following, more complicated example is a Bourne shell script that before running verifies that there are new *passwd_**username*** files to be incorporated. While running, it checks that each new entry does not already exist in the global file. Like the shorter C shell example, it incorporates the individual files found in */afs/transarc.com/common/uss/newaccts* into a new version of the global *passwd* file found in */afs/transarc.com/common/etc*.

```
#!/bin/sh
DESTDIR=/afs/transarc.com/common/uss/newaccts
cd $DESTDIR
DEST=/afs/transarc.com/common/etc
cp /afs/transarc.com/common/etc/passwd
\         /afs/transarc.com/common/uss/newaccts/passwd
echo "copied in passwd file."
PASSWD=/afs/transarc.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/transarc.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
```

16.5.12. Creating Links with the "L" and "S" Lines

The "L" line creates a "hard link" between two files, as achieved by the standard UNIX **ln** command. The "S" line 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. It does not create a second copy of the file.

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 different volumes. "L" and "S" lines should appear in the template after "F" and "E" lines if the file being linked to was created on such a line.

The UNIX file system records the UNIX user name of the issuer of the **uss** command as the owner of either type of link. The mode bits are set to *777* (`rwxrwxrwx`).

The syntax for the two types of lines is identical except for the first letter. Like other lines, they should appear in the template as a single line:

```
L <existing file> <link>
S <existing file> <link>
```

Note: You must not create a symbolic link to a file that begins with '#' or '%' because AFS will believe that the link is actually a mount point to a regular or ReadWrite volume, respectively.

Below is a "S" line from the example in section 16.5.6.2:

```
S /afs/transarc.com/public/$USER $MTPT/public
```

This example line creates a symbolic link to the user's public directory from his or her home directory. The user's public volume is mounted automatically by the preceding "X" line; see section 16.5.6.2.

The capital letter `L` or `S` tells the command interpreter that this is a link-creation instruction for a hard link or a symbolic link, respectively.

16.5.12.1. The <*existing file*> Field

The <*existing-file*> field specifies the complete pathname of the existing file for which you want to create a second name. If the pathname includes the user's home directory, you can represent it with the `$MTPT` variable, which takes its value from the <*mount point*> field on the "V" line.

In the Transarc example (see section 16.5.6.2), the system administrator specifies the pathname to the newly created public volume mounted in */afs/transarc.com/public/$USER* from the user's home directory. So the field reads

```
/afs/transarc.com/public/$USER
```

16.5.12.2. The <*link*> Field

The <*link*> field defines the complete pathname of the second name you wish to create for the *existing file*. Variables may appear in this field.

In the Transarc example (see section 16.5.6.2), the system administrator specifies link from the home directory to the newly created public volume mounted; the link is the entity *public* in the user's home directory. So the field reads

```
$MTPT/public
```

16.5.13. Specifying the Level of Password/Authentication Security with the "A" Line

The "A" line is used to specify the level of password/authentication security in the Authentication Database for the specified user. More specifically, it is used to define

- The lifetime of the user's password.
- Whether or not the user can immediately reuse an old password, or must wait for at least twenty password changes to reuse an old password.
- The limit on the number of consecutive failed authentication attempts allowed and the amount of time that the account will be locked against additional authentication attempts if the limit is exceeded.

The following should appear in the template as a single line:

`A` *<username>* *<password lifetime>* *<reuse/noreuse>* *<failures>* *<locktime>*

Below is an "A" line from the example in section 16.5.6.2:

```
A $USER 250 noreuse 5 10
```

This particular example sets a password lifetime of 250 days, prohibits the user from reusing any of his or her last twenty passwords, limits the number of failed authentication attempts to five, and creates a lockout time of ten minutes if the authentication limit is exceeded (this locktime will be rounded up to 17 minutes because the time must be in intervals of 8.5 minutes).

The capital letter `A` tells the command interpreter that this line sets the password/authentication security levels.

16.5.13.1. The <*username*> Field

The <*username*> field identifies the authentication ID associated with the Authentication Database entry to be changed. You can use the $USER variable here.

In the Transarc example (see section 16.5.6.2), the system administrator uses the `$USER` variable to identify the Authentication Database entry to be changed. So the field reads

```
$USER
```

16.5.13.2. The <*password lifetime*> Field

The <*password lifetime*> field sets the lifetime in days of the user password. Beyond the specified number of days, the password will be invalid (that is, the user will be unable to authenticate with his or her account). The range of valid values is 0 through 254. Use 0 to indicate that the password never expires; use 1 through 254 to specify the number of days that the password will be valid.

In the Transarc example (see section 16.5.6.2), the system administrator sets the password lifetime to 250 days.

16.5.13.3. The <*reuse/noreuse*> Field

The <*reuse/noreuse*> field specifies whether or not the user can reuse a password (during a password change) that he or she has used at any time during the last twenty password changes. The valid values for this field are *yes* (the user can immediately reuse an old password) and *no* (the user can only reuse an old password after 20 password changes).

In the Transarc example (see section 16.5.6.2), the system administrator restricts password reuse.

16.5.13.4. The <*failures*> Field

The <*failures*> field sets the maximum number of times that a user can consecutively fail to correctly type in his or her password during authentication (for example, using AFS **login** or **klog**), before being locked out of his or her user account for the amount of time specified by the <*locktime*> field. The range of valid values is 0 through 254. Use 0 to indicate that the user will never be locked out of his or her account based on the number of consecutive failed authentication attempts.

In the Transarc example (see section 16.5.6.2), the system administrator has set the limit to five attempts.

16.5.13.5. The <*locktime*> Field

The <*locktime*> field sets the amount of time that a user will be locked out of his or her user account if he or she exceeds the maximum number of failed authentication attempts specified by the **attempts** argument.

You can specify a time in either hours and minutes (hh:mm) or minutes only (mm). If a 0 is specified, the account will be locked indefinitely (until a system administrator unlocks it using **kas unlock**). If a time is entered (other than 0), it will be rounded up to the next highest increment of 8.5 minutes; so all locked accounts will be locked for some multiple of 8.5 minutes. The maximum finite value is 36 hours; for example, if *99:99* is specified, the locktime will be 36 hours.

In the Transarc example (see section 16.5.6.2), the system administrator has set the locktime to ten minutes (which means the actual locktime will be seventeen minutes).

16.5.14. Executing Commands with the "X" Line

The "X" line executes the indicated command, which may be a standard UNIX command, a shell script or program, or an AFS command. While you may execute any command you wish, the command should have something to do with the account just created.

The following should appear in the template as a single line:

```
X "<command>"
```

Below is an "X" line from the example in section 16.5.6.2:

```
X "create_public_mount $AUTO $USER"
```

In this example, the system administrator has created a script called **create_public_mount** which accepts two variables (the value of `$AUTO` and `$USER`) to create a mount for the user's public directory in */afs/transarc.com/usr/$AUTO/$USER*. Note that since the field contains spaces, it is enclosed in double quotes.

The capital letter `X` tells the command interpreter that this is a command to be executed.

16.5.14.1. The <*command*> Field

The <*command*> field specifies the UNIX or AFS command to be executed. It may contain spaces; however, if it does, the entire command must be surrounded by double quotes (" "). It may contain variables, which the command interpreter will resolve before executing the command, but it may not contain carriage returns.

16.6. Creating Individual Accounts with uss add

Once you have created a template, you can create an individual account by issuing the **uss add** command. When you issue the command, the **uss** command interpreter

- Instructs the Protection Server to create a Protection Database entry and assign an AFS UID which becomes the value of the `$UID` variable.
- Instructs the Authentication Server to create an Authentication Database entry associated with the specified password.
- Executes the instructions in the indicated template. This may require contacting the File Server, Volume Server, and VL Server.

Section 16.5 explains the effect of including different types of lines in the template. If the template is empty, only the Protection and Authentication Database entries are created.

16.6.1. Template Variables and Command Line Arguments: A Review

When issuing the **uss add** command, you must provide a command line argument for any variable that appears in the template you are using (in addition to the three arguments that are always required). You need not provide arguments for which the corresponding variable does not appear in the template; they will simply be ignored. If you fail to supply a required variable, it will be replaced by a null string, which usually causes the account creation to fail.

Table 2 summarizes the mappings between arguments and variables. It is adapted from Table 1, which appears in section 16.5.3.2, but includes only those variables that take their value from command line arguments.

Variable	Command Line Argument
`$USER`	**-user** <*login name*>
`$NAME`	**-realname** <*full name*>
`$SERVER`	**-server** <*file server for home volume*>
`$PART`	**-partition** <*partition for home volume*>
`$MTPT`	**-mount** <*home directory mount point*> (for first instance)
`$1` through `$9`	**-var** <*auxiliary argument pairs*>
`$UID`	AFS UID allocated by Protection Server or **-uid** **<***uid to assign the user***>**
`$PWEXPIRES`	**-pwexpires** **<***password expires in [0..254] days (0 => never)***>**

Table 2: Template Variables Taking Their Value from Command Line Arguments

⇒ To create an AFS account

Step 1: Verify that you have authenticated with AFS under an identity possessing all of the privileges necessary for performing the following steps. If necessary, issue the **klog** command. The required privileges and the commands used to confirm them are:

- Membership in *system:administrators*, necessary for creating a Protection Database entry, setting file and directory ownership, and setting access control lists. If necessary, issue **pts membership** which is fully described in section 20.3.

  `%` **pts membership system:administrators**

- Inclusion in */usr/afs/etc/UserList*, necessary for creating a volume. If necessary, issue **bos listusers** which is fully described in section 20.5.

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

- The `ADMIN` flag in your Authentication Database entry, necessary for creating an Authentication Database entry. If necessary, issue **kas examine** which is fully described in section 20.4.

  `%` **kas examine** **<***name of user***>**
  `Password for` *name of user*`:` <*your password*>

- The INSERT and ADMINISTER rights in the directory where you will mount the user's volume, necessary for creating a mount point in that directory. If necessary, issue **fs listacl** which is fully described in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: **Required for users of Ultrix. Optional with other operating systems, except when placing elements in the** UNIX **file system.** Become "root" in the machine's UNIX file system, if you are not already, by issuing the **su** command. (Section 16.2.7 explains why this is necessary for Ultrix users, and sections 16.5.9.3, 16.5.10.3, and 16.5.11.3 why it is necessary when creating elements in the UNIX file system.)

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

Step 3: Verify the location of the template you wish to use and that it functions as you expect. Section 16.5.4 explains where **uss** expects to find the template, though you can always provide an alternate pathname. You may wish to check the variables it contains, so you know which arguments you must provide on the command line.

Step 4: **(Optional.)** Change into the directory where the template resides.

% **cd** <*template directory*>

Step 5: **(Optional.)** Run **uss add** with the **-dryrun** flag to preview the creation of the account. Note any error messages and correct the cause before reissuing the command without **-dryrun**. The next step describes the **uss add** command's syntax.

Step 6: Create the account.

% **uss add -user** <*login name*> **[-realname** <*full name in quotes*>**]**
[-pass <*initial passwd*>**] [-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*>**]**
[-verbose] [-var <*auxiliary argument pairs (Num val)*>$^+$**]**
[-cell <*cell name*>**] [-admin** <*administrator to authenticate*>**]**
[-dryrun] [-skipauth] [-overwrite]
[-pwexpires <*password expires in [0..254] days (0 => never)*>**]**
[-help]

where

ad is an acceptable abbreviation of **add**.

-user specifies the user name that the user will type at the login prompt. It also becomes the name of the user's Protection and Authentication Database entries and provides the value of the `$USER` template variable. Section 2.9.4 discusses restrictions on user names and suggests some schemes for choosing them.

To comply with AFS and UNIX restrictions, this name may be no longer than eight characters and should not include the following characters: colon (:), period (.), newline, at sign (@), or spaces.

-realname specifies the user's real name and the value of the `$NAME` variable. Surround it with double quotes as delimiters, since it contains spaces and possibly punctuation. If not provided, it defaults to the user name provided with **-user**.

-pass defines the user's initial password. Many UNIX applications and utilities require that this be no longer than eight characters; AFS itself allows passwords of virtually unlimited length. You should instruct the user to change *initial password* to a truly secret password when logging in the first time (the *AFS User's Guide* provides instructions on changing passwords).

Possible choices for initial passwords include the username, a string of digits from a Social Security number or a standard string such as "changeme." If not provided, this argument defaults to the string "changeme."

-server specifies the file server machine on which the user's volume will reside. You must provide this argument if the variable `$SERVER` appears in the template.

You should provide the machine's complete Internet-style host name. You can avoid typing the entire name on the command line by combining the `$SERVER` variable with a constant in the "V" line's <*server*> field. See section 16.5.7.2.

-partition specifies the partition on *FileServer for home volume* on which the user's volume will reside. You must provide this argument if the variable `$PART` appears in the template.

Partition names conventionally have the form */vicep**x***, where ***x*** is a letter of the alphabet. See section 16.5.7.3 for a list of acceptable variants of this name format.

-mount specifies the pathname to the user's home directory. If the issuer does not provide a full pathname, it is interpreted relative to the working directory.

You must provide this argument if the `$MTPT` variable appears on the "V" line in the template. However, any occurrences of that variable after the "V" line's <*mount point*> fields take their value from the field, rather than directly from this command line argument. See section 16.5.7.5 for more information about this, and for suggestions about how to use this argument/variable pair.

-uid specifies the AFS UID to be assigned to the new user. It should be a positive integer. The issuer may wish to verify with **pts listmax** that the intended number is greater than the current largest AFS UID: if the requested AFS UID is already in use, the account creation will terminate with an error. If the issuer does not provide this argument, the Protection Server assigns an AFS UID automatically; as with manual account creation, automatic allocation is preferred. This argument provides the value for the `$UID` variable.

-template specifies which template file the command interpreter should consult. If the issuer does not provide this argument, the command interpreter looks for a template file called *uss.template* in the following directories (in the specified order):

1. The working directory.
2. */afs/**cellname**/common/uss*, where ***cellname*** names the local cell.
3. */etc*

If the issuer provides a name other than *uss.template* without a pathname, the command interpreter looks for it in the directories just listed, in the same order. If the issuer provides a full or partial pathname, the command interpreter consults the specified file without consulting the directories just listed; it interprets partial pathnames relative to the working directory.

-verbose causes the command interpreter to produce a more detailed trace of the actions it is executing. By default, only warnings and error messages appear.

-var specifies values for each of the number variables `$1` through `$9` found in the template. The number variables allow the issuer to provide values for variables not built in to the **uss add** command.

Each instance of this argument has two parts, separated by a space:

- An integer between 1 and 9, not preceded by the dollar sign. Each number matches the variable in the template.
- A value.

-cell specifies the cell in which to run the command.

-admin names the user whom the Authentication Servers should authenticate for purposes of creating an Authentication Database entry.

Note: Issuers of this command on machines running Ultrix must use this argument in order to adopt a privileged AFS identity while remaining "root" in the local machine's UNIX file system.

-dryrun indicates that the command interpreter should not actually execute the command, but should report all the actions it would perform if executing it.

-skipauth directs the command to bypass the Authentication Server, allowing a site using Kerberos to use the command with that form of authentication.

-overwrite instructs the command interpreter to overwrite any directories, files, and links that exist in the file system and for which it also finds definitions on template "D," "E," "F," "L," and "S" lines. If this flag is omitted, the command interpreter prompts once for confirmation that the issuer really wants to overwrite all such elements.

-pwexpires specifies the length of time in days that *name of user*'s password will be valid. Prior to the end of this period of time, the user must change his or her password using **kpasswd** or **kas setpassword**. The range of valid values is 0 through 254. Use 0 to indicate that the password never expires; use 1 through 254 to specify the number of days that the password will be valid. By default, a user's password will never expire.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

Step 7: Create an entry for the new user in the */etc/passwd* file of each machine that he or she will need to log into. Section 16.5.11 outlines a method for automating this step using the template "E" line and a shell script (assuming you want to set the UNIX UID equal to the AFS UID).

Even if you do not use the "E" line method, you should 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 **uss** output, but you can also learn it with **pts examine**.

% **pts examine** <*user or group name or id*>

16.7. Deleting Individual Accounts with uss delete

Unlike **uss add**, the **uss delete** command does not use a template file. Instead, it derives all of the information it needs to delete a user account from the information provided by the command line arguments and flags.

When you issue the **uss delete** command, the **uss** command interpreter

- Instructs the File Server to remove the user's home directory mount point from the file tree.
- Instructs the Volume Server to remove and delete the user's home volume and instructs the VL Server to delete the VLDB entry; if you do not want to delete the home volume and VLDB entry, specify the **-savevolume** flag. If you do intend to delete the volume, you may want to transfer its contents to tape for later restoration.
- Instructs the Authentication Server to delete the user's Authentication Database entry.
- Instructs the Protection Server to delete the user's Protection Database entry.

16.7.1. Issues to Consider Before Deleting an Account

Before you actually issue the **uss delete** command to delete a user account, there are several administrative tasks that you may want to consider:

- Will you need to restore the user's account and/or volume at a later date? If you will, you may want to maintain a record of the user's user name and UID for future reuse; you may also want to maintain a backup of the user's home volume for later restoration. Chapter 6 contains information on making backups.
- Does the user exclusively use a volume, other than his or her home volume (e.g., a backup volume or a project volume)? If he or she does, you will have to remove that volume manually. This removal is a separate task from the deletion of the user account. You may also want to maintain a backup of this volume for later restoration. Chapter 5 contains information on managing volumes.
- Does the user exclusively own any protection groups? If he or she does, you will need to reassign ownership of that group to another individual or group. If you delete a user and that user is still the exclusive owner, no one except a member of the group *system:administrators* will be able to administer that group. This type of group is referred to as an **orphaned** group. Chapter 18 contains information on administering the Protection Database.

Note: The **uss delete** command cannot automatically perform the tasks listed above; however, they can be automated using the **uss bulk** command. Although the **uss bulk** command is intended for the creation or deletion of multiple accounts, it can be used to create or delete individual accounts. And you may find this especially useful for delete operations because the bulk input file used by **uss bulk** can contain `exec` lines which provide the **uss bulk** command with functionality similar to the "X" line in a **uss add** template file. See section 16.8 for more information on **uss bulk**.

⇒ To delete an AFS account

Step 1: Verify that you have authenticated with AFS under an identity possessing all of the privileges necessary for performing the following steps. If necessary, issue the **klog** command. The required privileges and the commands used to confirm them are:

- Membership in *system:administrators*, necessary for deleting a Protection Database entry. If necessary, issue **pts membership** which is fully described in section 20.3.

  % **pts membership system:administrators**

- Inclusion in */usr/afs/etc/UserList*, necessary for removing and deleting a volume. If necessary, issue **bos listusers** which is fully described in section 20.5.

  % **bos listusers** <*machine name*>

- The `ADMIN` flag in your Authentication Database entry, necessary for deleting an Authentication Database entry. If necessary, issue **kas examine** which is fully described in section 20.4.

  % **kas examine** <*name of user*>
  `Password for` *name of user*`:` <*your password*>

- The DELETE right in the directory containing the user's volume, necessary for removing a mount point in that directory. If necessary, issue **fs listacl** which is fully described in section 19.3.

  % **fs listacl [**<*dir/file path*>**]**

  If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: **Required for users of Ultrix. Optional with other operating systems, except when placing elements in the** UNIX **file system.** Become "root" in the machine's UNIX file system, if you are not already, by issuing the **su** command. (Section 16.2.7 explains why this is necessary.

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

Step 3: Make sure that you have considered and resolved the issues listed in section 16.7.1 relating to the continued maintenance of a deleted user's account information, owned groups, and volumes.

Step 4: **(Optional.)** Run **uss delete** with the **-dryrun** flag to preview the deletion of the account. Note any error messages and correct the cause before re-issuing the command without **-dryrun**. The next step describes the **uss delete** command's syntax.

Step 5: Delete the account.

uss delete -user <*login name*>
-mountpoint <*mountpoint for user's volume*> **[-savevolume]**
[-verbose] [-cell <*cell name*>**]**
[-admin <*administrator to authenticate*>**] [-dryrun]**
[-skipauth] [-help]

where

d is an acceptable abbreviation of **delete**.

-user specifies the user name to remove from the Protection and Authentication Databases.

-mountpoint specifies the pathname to the user's home directory. The volume mounted at that location is deleted from the system. If the issuer does not provide a full pathname, it is interpreted relative to the working directory.

-savevolume prevents the user's volume and VLDB entry from being deleted from the system.

-verbose causes the command interpreter to produce on standard output (stdout) a more detailed trace of the actions it is executing. By default, only warnings and error messages appear.

-cell specifies the cell in which to run the command.

-admin names the user whom the Authentication Servers should authenticate for purposes of deleting the Authentication Database entry.

-dryrun indicates that the command interpreter should not actually execute the command, but should report all the actions it would perform if executing it.

-skipauth directs the command to bypass the Authentication Server, allowing a site using Kerberos to use the command with that form of authentication.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

Step 6: Delete the entry for the old user from the local password file (*/etc/passwd* file or equivalent) of each machine that he or she formerly logged into. If you intend to reactivate the user's account in the future, you may want to simply comment out the user entry or place an asterisk (*) in the password field.

16.8. Creating and Deleting Multiple Accounts with uss bulk

The **uss bulk** command allows you to create and delete many accounts at once. Before executing the command, you must

- Construct a template for all accounts that you intend to create with a single **uss bulk** command just as you would for accounts created individually with **uss add**. All of the accounts created with a single **uss bulk** command must use the same template.
- Construct a "bulk input file" that contains information will be used by **uss bulk** to create accounts, delete accounts, and execute programs and/or scripts (which are normally related to the creation or deletion of accounts).

16.8.1. Constructing a Bulk File for Account Creation and Deletion

Five types of command lines can appear in the bulk input file: `add`, `delete`, `exec`, `savevolume`, and `delvolume`. Each command line should have a carriage return only at the end, even though it may cover several lines on the screen.

The `add` ***line***

Each `add` line is the equivalent of a **uss add** command issued on the command line. Begin the line with "`add`" only, not "`uss add`", and provide the arguments in the same order they would appear on the **uss add** command line, separating each with a colon. Only the first argument, corresponding to the command line argument **-user**, is required. To omit a value for an argument (presumably because it is optional or the template specifies a constant value for it), type nothing between two colons. After the last argument provided, end the line with either a colon and carriage return, or a carriage return alone.

Fields nine through eighteen are for assigning values to the number variables, with the fields listed in increasing numerical order. The issuer must indicate either a value or an empty field (nothing between two colons) for every variable that precedes the last one being assigned an actual value. It is acceptable, but not necessary, to indicate empty fields for any number variables following the last one actually assigned a value.

The following representation uses the same identifiers for arguments as the **uss add** definition. The "`{ }`" notation indicates that each entry between vertical bars is one possible choice. Remember that an instruction like this would appear on a single line in the actual bulk input file.

`add` <*login name*>[:<*full name*>][:<*initial passwd*>][:<*password expires*>]
[:<*file server for home volume*>][:<*file server's disk partition for home volume*>]
[:<*home directory mount point*>][:<*uid to assign the user*>]
[:{<*var1*> | :<*var1*>:<*var2*> | :<*var1*>:<*var2*>:<*var3*> |
. . . . **et cetera through**. . . .
| :<*var1*>:<*var2*>:<*var3*>:<*var4*>:<*var5*>:<*var6*>:<*var7*>:<*var8*>:<*var9*>}]

Do not surround *full name* with double quotes in the bulk input file, even though its counterpart on the regular **uss add** command line must be so surrounded.

The EXAMPLES section below may help to clarify the format of the `add` line.

The `delete` *line*

Each `delete` line is the equivalent of a **uss delete** command issued on the command line. Begin the line with "`delete`" only, not "`uss delete`", and provide the arguments in the same order they would appear on the **uss delete** command line, separating each with a colon. Provide values for at least the first two arguments (corresponding to **-user** and **-mountpoint** on the command line). The third field, corresponding to **-savevolume**, is optional; there are three possible values:

- If the word `savevolume` appears, the volume and VLDB entry will not be removed
- If the word `delvolume` appears, the volume and VLDB entry will be removed
- If the field is blank, then the volume will be treated according to the prevailing default (see the following description of the `savevolume` and `delvolume` lines to learn how to set the default).

After the last argument provided, end the line with either a colon and carriage return or a carriage return alone.

The following representation uses the same identifiers for arguments as does the **uss delete** definition. The "`{ }`" notation indicates that each entry between a vertical line is one possible choice. Remember that an instruction like this would appear on a single line in the actual bulk input file. The EXAMPLES section illustrates use of this line.

`delete` <*account name*>:<*full mount point pathname*>
[:{`savevolume` | `delvolume` | }]

The `exec` *line*

This line causes the indicated command to be executed. The command can be a UNIX shell script or command, a program, or an AFS command. Its format is

`exec` <*command string to execute*>

The `savevolume` *and* `delvolume` *lines*

The `savevolume` and `delvolume` lines set the default treatment of volumes in `delete` lines that follow them in the bulk input file. The `savevolume` line prevents the removal of volume and VLDB entry for all subsequent `delete` lines. The `delvolume` line causes the removal of volume and VLDB entry for all subsequent `delete` lines. The default treatment if neither line appears in a bulk input file is to remove the volume and VLDB entry; `delete` lines that appear before the first `savevolume` line are also treated this way.

An explicit `savevolume` or `delvolume` instruction in the third field of an individual `delete` line overrides the default in effect at the time. If nothing appears in the third field, the default is obeyed.

Neither command takes arguments (the word `savevolume` or `delvolume` should appear by itself on the line). The effect of either line lasts until the end of the bulk input file, or until its opposite appears in the file. Multiple instances of each command can be used to toggle back and forth between removal and nonremoval of volumes.

16.8.2. Example Uses of the Bulk Input File

The following shows the proper format for an `add` line in the bulk file when only the first (required) argument is provided, and when the first three arguments are provided. In the first case, the user's "real name" is set to *anderson* and password to *changeme* (the defaults).

```
add anderson
add smith:John Smith:js_pswd
```

The following `add` line example supposes that the Transarc Corporation cell uses a template with the following "V" line in it:

```
V user.$USER $SERVER.transarc.com /vicep$PART 2000
       /afs/transarc.com/usr/$3/$USER $UID $USER all
```

To create an account for users John Smith from the Marketing Department and Pat Jones from the Finance Department, the appropriate **add** commands in the bulk input file might be:

```
add smith:John Smith:::fs1:a:::::marketing
add jones:Pat Jones:::fs3:c:::::finance
```

The effect would be to establish an account called *smith* in the Protection and Authentication Databases, with an initial password *changeme* and a value for `$UID` provided by the Protection Server. His volume, `user.smith`, would reside on partition "/vicepa" of file server machine "fs1.transarc.com" and would be mounted at */afs/transarc.com/usr/marketing/smith*. He would own his home directory and have full access to it. The account for *jones* would be similar, except that it would reside on partition "/vicepc" of file server machine "fs3.transarc.com" and would be mounted at */afs/transarc.com/usr/finance/jones*.

Notice that the fields corresponding to *<home directory mount point>*, *<uid to assign the user>*, *<var1>* and *<var2>* are empty (between `a` and `marketing` on the first example line) since their corresponding variables do not appear in the template file. The *<initial passwd>* field is also empty.

The issuer could, if he or she choose, specify values/empty fields for all nine number variables. In this case, the bulk input file lines shown above would look like:

```
add smith:John Smith:::fs1:a:::::marketing::::::
add jones:Pat Jones:::fs3:c:::::finance::::::
```

The following shows a complete bulk input file containing a set of `delete` lines combined with a `savevolume` line. Because *smith, pat,* and *rogers* appear before the `savevolume` command and their third field is blank, the volume will be removed. The volume for *terry* will not be removed, but *johnson*'s will be because the `delvolume` in the third field overrides the current default.

```
delete smith:/afs/transarc.com/usr/smith
delete pat:/afs/transarc.com/usr/pat
delete rogers:/afs/transarc.com/usr/rogers
savevolume
delete terry:/afs/transarc.com/usr/terry
delete johnson:/afs/transarc.com/usr/johnson:delvolume
```

The following `exec` example supposes that the bulk input file contains a set of `add` lines followed by `delete` lines. The operator places this line between the two sets to flag when the additions are finished and the deletions beginning.

```
exec echo "Additions completed; beginning deletions..."
```

⇒ To create and/or delete multiple AFS user accounts

Step 1: Verify that you have authenticated with AFS under an identity who possesses all of the privileges necessary for performing the following steps. If necessary, issue the **klog** command. The required privileges and the commands used to confirm them are

- Membership in *system:administrators*, necessary for creating and deleting a Protection Database entry, setting file and directory ownership, and setting access control lists. If necessary, issue **pts membership** which is fully described in section 20.3.

  % **pts membership system:administrators**

- Inclusion in */usr/afs/etc/UserList*, necessary for creating and deleting a volume. If necessary, issue **bos listusers** which is fully described in section 20.5.

  % **bos listusers** <*machine name*>

- The `ADMIN` flag in your Authentication Database entry, necessary for creating and deleting an Authentication Database entry. If necessary, issue **kas examine** which is fully described in section 20.4.

  % **kas examine** <*name of user*>
  `Password for` *name of user*`:` <*your password*>

- Either the INSERT and ADMINISTER rights in the directory where you are going to mount the user's volume, necessary for creating a mount point in that directory; or the DELETE rights in the directory from which you are going to remove the user's mount point. If necessary, issue **fs listacl** which is fully described in section 19.3.

  % **fs listacl [**<*dir/file path*>**]**

  If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: **Required for users of Ultrix. Optional with other operating systems, except when placing elements in the UNIX file system.** Become "root" in the machine's UNIX file system, if you are not already, by issuing the **su** command. (Section 16.2.7 explains why this is necessary for Ultrix users, and sections 16.5.9.3, 16.5.10.3, and 16.5.11.3 why it is necessary when creating elements in the UNIX file system.)

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

Step 3: If you are creating accounts with **uss bulk**, verify the location of the template file you wish to use, and that it functions as you expect. Section 16.5.4 explains where **uss** expects to find the template, though you can always provide an alternate pathname. You may wish to check the variables it contains, so you know which arguments you must provide on the lines in the bulk input file.

Step 4: Create a bulk input file that complies with the rules in section 16.8.1. For the sake of simplicity, you may wish to put the file in the same directory as the template you are going to use.

Step 5: **(Optional.)** Change into the directory where the bulk input file and/or template file reside. This will affect the type of pathnames you must type in step 6.

Step 6: Issue the **uss bulk** command.

% **uss bulk** <*bulk input file*> **[-template** <*pathname of template file*>**]**
[-verbose] **[-cell** <*cell name*>**]**
[-admin <*administrator to authenticate*>**]** **[-dryrun]**
[-skipauth] **[-overwrite]** **[-help]**

where

b is an acceptable abbreviation of **bulk**.

-file names the *bulk input file* containing the **uss** commands to execute. If you do not provide a pathname, the command interpreter assumes the file is in the working directory.

-template names the template file for any `add` commands that appear in the *bulk input file*. You may omit this argument if the desired template is called *uss.template* and resides in either the current working directory, */afs/**cellname**/common/uss* or */etc*. If the template has a different name, you can provide only its name if it resides in one of those directories. Otherwise, you must provide a full or partial pathname. If you do not provide a pathname, the command interpreter assumes the template file is in the working directory.

-verbose causes the command interpreter to produce (on stdout) a more detailed trace of the actions it is executing. By default, only warnings and error messages appear.

-cell specifies the cell in which to run the command.

-admin names the user whom the Authentication Servers should authenticate for purposes of creating an Authentication Database entry. If you do not provide it, the authentication is attempted for the user name under which you issuer are logged into the local machine's UNIX file system. You must use this argument if you are working on an Ultrix machine, or if the template creates elements in the UNIX file system rather than in AFS; see step 2.

-dryrun indicates that the command interpreter should not actually execute the command, but should report all the actions it would perform if executing it.

-skipauth directs the command to bypass the Authentication Server, allowing a site using Kerberos to use the command with that form of authentication.

-overwrite instructs the command interpreter (when `add` lines appear in the *bulk input file*) to overwrite any directories, files and links that exist in the file system related to the user for which it also finds definitions on template "D," "E," "F," "L," and "S" lines. If this flag is omitted, the command interpreter prompts, once for each `add` line in the *bulk input file*, for confirmation that the issuer really wants to overwrite those elements. This flag should not be used if the *bulk input file* does not contain `add` lines.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

Step 7: Create an entry for each new user in the local password file (*/etc/passwd* file or equivalent) of each machine that he or she will need to log into. Section 16.5.11 outlines a method for automating this step using the template "E" line and a shell script (assuming you want to set the UNIX UID equal to the AFS UID).

Note: Even if you do not use the "E" line method, you should set the user's UNIX UID to match the AFS UID assigned automatically by the Protection Server or by you in the <*uid to assign the user*> field. The new user's AFS UID appears in the **uss** output, but you can also learn it with **pts examine**.

% **pts examine** <*user or group name or id*>

If you are deleting users using **uss bulk**, delete the entry for each old user from the local password file (*/etc/passwd* file or equivalent) of each machine that into which he or she formerly logged. If you intend on reactivating the user's account in the future, you may want to simply comment out the user entry or place an asterisk (*) in the password field.

17. Administering User Accounts

This chapter explains how to create and maintain user accounts in your cell. It contains advice for how to configure accounts, as well as basic instructions, and has the following sections:

17.1	Summary of Commands	17-1
17.2	The Components of a Complete AFS User Account	17-2
17.3	About Creating */etc/password* Entries	17-3
17.4	About Converting Pre-Existing UNIX Accounts	17-5
17.5	Creating AFS User Accounts	17-7
17.6	Improving Password/Authentication Security	17-14
17.7	Changing Passwords	17-20
17.8	Listing and Setting Users' Volume Quota	17-21
17.9	Changing User Names	17-22
17.10	Removing User Accounts	17-25

17.1. Summary of 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**
Set volume quota	**fs setquota**
Change password in Authentication Database	**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**

17.2. The Components of a Complete AFS User Account

Because AFS and the UNIX file system are different, a complete AFS user account is not the same as a UNIX user account. This section explains the components you must create in order for an AFS user account to be complete.

The preferred method for creating user accounts is to use the **uss** program, which allows you to create multiple accounts with a single command. See chapter 16 for details. The instructions in section 17.5 tell you how to create each component individually. The components appear here in the same order as they are created, regardless of methods:

- An entry in the ***Protection Database*** (required). It defines the user's "user name" (the name the user types at login). It is mapped to the user's AFS UID, a number that uniquely identifies the user in AFS, just as the UNIX UID does in the UNIX file system. The AFS servers refer to users by AFS UID rather than user name. The Protection Database entry also records all the protection groups to which the user belongs. See chapter 18 for more about maintaining Protection Database entries.
- An entry in the ***Authentication Database*** (required). This entry records the user's password in a scrambled form suitable for use as an encryption key. When the user provides a password at login (or with the **klog** command), the login program scrambles it the same way so the Authentication Server can verify that the user has provided the correct password.
- A ***volume***. A volume is a "container" that keeps all the files in the user's home directory together on a single partition of a file server machine. See chapter 5 for more about volumes.
- A space ***quota*** on the volume that limits its maximum size. By default, AFS sets each volume's quota to 5 megabytes when it is created. This may or may not be an appropriate amount for your cell, so the instructions tell you how to alter it if you wish. For more information about volume quota, see section 5.12.
- A ***mount point*** that associates the user's volume with his or her home directory (that is, serves as the "root directory" of the volume). The mount point makes the contents of the volume visible and accessible in the file system tree. For more information about mount points, see section 5.2.6.
- Full ***ownership*** of and ***access rights*** on the home directory. In order to be able to manipulate all the file and subdirectories in the home directory, the user must own the home directory (in the standard UNIX sense) and have all the available access rights (as defined by AFS).
- An **entry in the local password file** (*/etc/passwd* or equivalent) of each machine the user will log into. Logging into a machine is necessary because AFS files are accessible only through a Cache Manager, which resides in the client machine kernel. See section 17.3.
- ***Standard files***. AFS does not require these files, but you may install them to help the user log on and off the system more easily, receive electronic mail, enable printing, etc.

17.3. About Creating */etc/passwd* Entries

In addition to AFS-specific components of an account, a user must also have an entry in the local password file (*/etc/passwd* or equivalent) of each client machine he or she will will log onto. This applies to all three types of accounts described in section 17.5.

17.3.1. Assigning AFS and UNIX UIDs

For simplicity and ease of administration, you should make the user's UNIX UID (defined in */etc/passwd* **or equivalent) and AFS UID (defined in the Protection Database) the same.** It is assumed throughout this guide that you have done so.

If your cell does not have existing UNIX accounts to convert into AFS accounts, you should allow the Protection Server to allocate AFS UIDs. You can let it start at its default, or use **pts setmax** to designate any starting point you wish. Although it is not recommended, you can also assign AFS UIDs explicitly by using the **-id** argument on **pts createuser**.

If you wish to convert existing UNIX accounts into AFS accounts (or create new AFS accounts in a cell where you have previously converted UNIX accounts), you must ensure that each user's AFS and UNIX UIDs match. See section 17.4 for details.

The **uss** command allows you to assign AFS UIDs. This makes it much more convenient to convert existing UNIX accounts using **uss**. See section 16.4.

17.3.2. Specifying Passwords in the Local Password File

If you do not use an AFS version of **login**, you must place an actual password in the local password file (*/etc/passwd* or equivalent) on each machine the user will log into. Non-AFS versions of **login** log issuers into the local machine's UNIX file system only. Users then need to issue **pagsh** and **klog** to authenticate with AFS. It is simplest if the passwords in */etc/passwd* and the Authentication Database are the same, but it is not required.

If you use the AFS version of **login**, the password field in a machine's password file is no longer the primary "gate" for access to your system. If the user provides the correct AFS password, then AFS **login** automatically authenticates the user with AFS and logs him or her into the local machine's UNIX file system, without consulting the local password file. However, you can still use the password field to control access, in the following manner:

- An asterisk (*) in this field prevents the user from either authenticating with AFS or logging into the machine's UNIX file system. This is useful mainly in emergencies, when you want to prevent a certain user from logging into the machine.
- A string of any length other than the standard thirteen characters means that if the user does not provide the correct AFS password, he or she will also not be allowed to log into the local machine's UNIX file system. This is appropriate if you want to ensure that only people with

AFS accounts in your cell can log into your machines. Putting a single "X" or other character is the simplest choice.

- A string of thirteen characters representing the encrypted form of a password set with **/bin/passwd** means that if the user does not provide the correct AFS password, he or she can still login to the local UNIX file system by providing the correct UNIX password.

For more details on the AFS version of **login**, see section 2.11.2 or the **login** page in the *AFS Command Reference Manual*.

17.4. About Converting Pre-Existing UNIX Accounts

Many cells installing AFS for the first time have pre-existing UNIX accounts which they wish to convert to AFS accounts. There are three main issues you need to address. Sections 17.4.1 through 17.4.3 discuss them in detail:

- Making UNIX and AFS UIDs match. See section 17.4.1.
- Setting the password field in the local password file appropriately. See section 17.4.2.
- Moving files into AFS. See section 17.4.3.

17.4.1. Making UNIX and AFS UIDs Match

As mentioned in section 17.3, AFS users retain an entry in the local password file on client machines, and administration is much simpler when a user's UNIX UID and AFS UID are the same. You have two alternatives when converting UNIX accounts: you can either retain the existing UNIX UIDs (setting the new AFS UIDs to match), or change the UNIX UIDs to match the AFS UIDs.

17.4.1.1. Alternative 1: Retain UNIX UIDs

The first alternative when converting UNIX accounts into AFS accounts is to retain each user's existing UNIX UID and make the AFS UID match. In this case, you need to take two steps.

1. You need to assign an AFS UID that matches the user's present UNIX UID, rather than allow the Protection Server to assign one. The **-id** argument available on **pts createuser** allows you to assign an AFS UID.

   Because the UNIX UID is not being changed, you may not need to alter the user's entry in the local password file. However, if you are using AFS **login**, you may want to change the entry's password field; see section 17.3.2 to learn how the contents of this field affect the behavior of AFS **login**.

2. You need to make sure that no new AFS UIDs overlap with existing UNIX UIDs. Therefore, before creating accounts for any new users (ones without existing UNIX accounts), you need to make sure that the `UserMax` counter is set higher than your cell's largest UNIX UID. You can examine the current value of `UserMax` with **pts listmax** and set it if necessary with **pts setmax**. Both commands are described in section 18.13.

17.4.1.2. Alternative 2: Change UNIX UIDs

The second alternative when converting existing accounts is to replace the current UNIX UIDs with new AFS UIDs, preferably allowing the Protection Server to allocate the AFS UIDs itself. You will need to alter any existing local password files appropriately.

There is one drawback to changing the UNIX UID: the owner of the original UNIX files and directories will no longer be computed correctly. However, because you will need to move these file and directories into AFS anyway, changing their owner at the same time may not be a problem. See section 17.4.3.

17.4.2. Setting the Password Field Appropriately

The user with an existing UNIX account already has an entry in the local password file, presumably with a (coded) password in the password field. What should appear in that field after the conversion to AFS depends on which version of **login** you are using.

- If you are not using the AFS version of **login**, an actual password must appear (in scrambled form) in the password field of the local password file on every machine the user will log into.
- If you are using AFS **login**, then what you enter in the password field of the local password files controls some aspects of **login**'s behavior. Section 17.3.2 describes the choices and their effects.

17.4.3. Moving UNIX files into AFS

Users with existing UNIX accounts may have created files and directories in their accounts, and you may wish to transfer these files into AFS. The easiest method is to move all relevant UNIX files and directories onto the local disk of an AFS client machine, and then move them from the local disk into the user's new AFS home directory (using **mv**). (An alternative is to move all the UNIX directories and files onto a single machine and then configure the machine as an AFS client, but you could run out of machines before running out of accounts.)

If you chose to change the user's UNIX UID to match a new AFS UID, you will need to change the ownership of the former UNIX files and directories. Once you move elements into AFS, only members of *system:administrators* can issue **chown**. Also, remember that in AFS only the "owner" mode bits are relevant, and their effect interacts with the access rights on the ACL of the file's parent directory. The setting of the "group" and "other" bits no longer protects the file from unauthorized access; instead the file needs to be placed in a directory with an equally protective ACL.

17.5. Creating AFS User Accounts

There are two methods for creating user accounts. The preferred method—using the **uss** commands—allows 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 user name). To learn more about **uss**, see chapter 16.

The second method requires issuing a separate command to create each component of the account. It is best suited to creation of a single user at a time, since some of the commands only allow one instance for their arguments. The instructions in this section describe the procedures for creating components; section 17.2 explains the function of each component.

17.5.1. The Three Types of User Accounts

Because different users require different levels of user account functionality, AFS allows you to provide or withhold certain account components. Of course, only some of the levels of functionality make sense. For instance, it is possible to stop the creation process before granting a user complete access rights to his or her own home directory, but doing so makes the user unable to create any files.

In general, there are three levels of account functionality commonly used and each level corresponds to a type of account. Note that all of them include an entry in the local password file of each machine where the user will login.

17.5.1.1. Level 1: The Authentication-Only Account

The minimal components of an AFS user account are an entry in the Protection Database and an entry in the Authentication Database. These components allow the user to authenticate in AFS and to issue commands (including privileged ones if the necessary permissions are granted). This level of functionality is of little use to regular users because without a volume or home directory, they cannot create any files of their own. It is useful for "dummy" administrative accounts, and to enable users from foreign cells to access information as locally authenticated users.

To achieve this minimum level, perform steps 1 through 4 and step 12 listed below. In step 1, you need to pick only a user name, AFS UID, and initial password.

17.5.1.2. Level 2: The Basic Account

The Basic Account contains a volume for the user, in addition to Authentication and Protection Database entries. You mount the volume in the file system at the user’s home directory, set a space quota on it, and define the initial entries on its access control list. The user can then create and store files in his or her own volume.

To achieve this level of functionality, perform steps 1 through 7 and 10 through 12 listed below.

17.5.1.3. Level 3: The Full Account

A Basic Account is completely functional, but may not be as convenient and useful as one that comes equipped with directories and configuration files that help provide some basic functions, such as logging in, printing, and mail delivery. Section 2.9.8 suggests some of the appropriate elements to create.

To create an account at this level of functionality, perform all steps indicated in the instructions below. Section 2.9.8 suggests some of the appropriate elements to create.

⇒ To create a single user account with individual commands

Step 1: Select the following:

- A user name. By convention, the names of many components of the user account incorporate this name. Section 2.9.4 discusses restrictions on user names and suggests some schemes for choosing them.
- An AFS UID, if you are not going to allow the Protection Server to allocate one itself. In most circumstances, you should let the Protection Server allocate an AFS UID automatically. Assign one yourself only when you are creating an AFS user account for someone who already has a UNIX user account and UID. In this case, the new AFS UID should match the UNIX UID. An error results if you specify an AFS UID that is already in use. See section 17.4 for more information.
- An initial dummy password. Advise the user to change this at the first login. The user can learn how to change passwords in the *AFS User's Guide*.
- A name for the user's volume. The recommended name is "`user.`*user name*". Example: `user.smith`.
- A site (disk partition on a file server machine) for the user's volume. If your cell designates certain machines/partitions for user volumes only, that will guide your decision. You may wish to put the volume on the emptiest partition that meets your other criteria. You can learn how much space is available on a file server machine's partitions (and their total size) with **vos partinfo**, fully described in section 5.3.
- The "usr" directory in which to mount the user's volume (that is, place the user's home directory). Normally, this is similar to */afs/**cell**/usr*. Example: */afs/transarc.com/usr*.
- The quota you wish the volume to have. This depends on the amount of disk space available. The default quota assigned to new volumes is 5 megabytes.
- The access rights you wish to grant on the home directory, if the default of ALL granted to *system:administrators* is not satisfactory.

Step 2: Verify that you have all the privileges necessary for performing all the steps:

- You belong to *system:administrators*, (necessary for steps 3, 7 and 11). If necessary, issue **pts membership**, which is fully described in section 20.3.

% **pts membership system:administrators**

- You are included in */usr/afs/etc/UserList*, (necessary for step 5). If necessary, issue **bos listusers**, which is fully described in section 20.5.

  % **bos listusers** <*machine name*>

- You have the `ADMIN` flag in your Authentication Database entry, (necessary for step 4). If necessary, issue **kas examine**, which is fully described in section 20.4.

  % **kas examine** <*name of user*>
  `Password for` *name of user*`:` <*your password*>

- You have the INSERT and ADMINISTER rights in the directory where you are going to mount the user's volume, (necessary for step 6). If necessary, issue **fs listacl**, which is fully described in section 19.3.

  % **fs listacl** <*dir/file path*>

  If you are a member of *system:administrators*, you have implicit ADMINISTER rights on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

- You know the "root" password on your local machine, (necessary for step 11 with some system types).

Step 3: Create an entry in the Protection Database. You can review section 17.3.1 for a discussion of the issues surrounding AFS UIDs. If you are converting an existing UNIX account into an AFS account, also see section 17.4.

% **pts createuser** <*user name*> **[**<*user id*>**]**

where

cu is an acceptable alias for **createuser**.

user name will become the user's user name, the character string typed when logging in. AFS allows any length up to 63 characters, but does not allow uppercase letters. You should keep user names shorter than 8 characters because many UNIX programs cannot use longer names. Also, do not include any characters considered special in UNIX, particularly the colon and period.

user id is optional and not generally recommended. Provide this only if the user already has a UNIX UID and you want the AFS UID to match. If you do not provide it, the Protection Server assigns one automatically (section 18.13 explains how). If the ID you specify is already in use, an error results.

Step 4: Create an entry in the Authentication Database. For security purposes, provide the user's temporary password as *initial password* at the prompts, rather than on the command line. It should not be more than 8 characters long, as some programs cannot use longer passwords.

You should instruct users to change *initial password* to a truly secret password when he or she logs in the first time (the *AFS User's Guide* provides instructions on changing passwords). Possible choices for initial passwords include the user's user name, a string of digits from his or her Social Security number, or a standard string such as "changeme."

```
% kas create <name of user>
Password for your user name: <your password>
initial_password: <initial password>
Verifying, please re-enter initial_password:
                                  <initial password>
```

where

cr is an acceptable abbreviation for **create**.

name of user is the same one you specified in step 3.

your user name is the issuer's user name.

Step 5: Create the user's volume.

% **vos create** <*machine name*> <*partition name*> <*volume name*>

where

machine name is the name of the file server to house the volume.

partition name specifies the partition on *machine name* that will house the volume.

volume name names the new volume. The recommended name for user volumes is "`user.`*user name*" where *user name* is the same as in step 3. Volume names cannot be longer than 22 characters, so they remain below the absolute limit of 31 characters if a *.readonly* extension is added.

Step 6: Mount the volume in the file tree. Because mount points look like directories, this step is equivalent to creating a home directory for the user.

% **fs mkmount** <*directory*> <*volume name*>

where

mk is an acceptable abbreviation for **mkmount**.

directory names the mount point you wish to create. Do not create a directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory; the usual choice is */afs/**cell**/usr*. The recommended name for the final element in the pathname (the "home directory" itself) is the *user name* assigned in step 3.

volume name is the name you assigned to the volume in step 5.

Step 7: Give the user full access rights to the home directory.

% **fs setacl** <*directory*> <*user name*> **all**

where

sa is an acceptable alias for **setacl**.

directory is the home directory name as assigned in step 6.

user name is the user name assigned in step 3.

all grants all possible access rights.

Step 8: **(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 may wish to move some files and directories into the user's new AFS home directory. See section 17.4 for direction.

Step 9: **(Optional.)** In the new *.login* or shell initialization file, define the user's `$PATH` variable to include the directories where AFS binaries are kept (for example, */usr/afsws/bin* and */usr/afsws/etc*).

Step 10: In steps 11 and 12, you must know the user's AFS UID. If you let the Protection Server assign it in step 3, you probably do not know it. If necessary, look up the AFS UID using **pts examine**.

% **pts examine** <*user or group name or id*>

where

e is an acceptable abbreviation of **examine**.

user or group name or id
is the user name assigned in step 3.

The first two entries on the first line of the output are the user name and AFS UID. See section 18.4 for an example of the output.

Step 11: Make the user the owner of the home directory and any files and subdirectories created or moved in step 8. Rather than a user name, specify the AFS UID you learned in step 10 as the new owner. If the user is new, this is necessary 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, there may be an entry but the UID in it may be incorrect. If you supplied a user name, the new owner is recorded as the UID currently mapped to that user name.

Note: Some operating systems (Ultrix, for example) allow only "root" to issue the **/etc/chown** command. If necessary, become "root" in the machine's UNIX file system by issuing the **su** command.

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

% **/etc/chown** <*new owner ID*> <*directory*>

where

new owner ID is the user's AFS UID, which you learned in step 10.

directory is the user's home directory you created in step 6, plus each subdirectory or file you created in step 8. Provide a complete pathname for each *directory* that is not in the working directory.

Step 12: Create or modify an entry for the new user in the local password file (*/etc/passwd* or equivalent) of each machine the user will log into. Remember to make the UNIX UID the same as the AFS UID you learned in step 10, and to fill the password field appropriately (see section 17.3.2). You may wish to make the change in a global version of this file and then use **package** to distribute the global version to client machines; see chapter 14 to learn about **package**.

17.5.2. Changing Default Quota and ACL After Volume Creation

The default quota on an AFS volume is 5 megabytes. If this is not satisfactory in your cell, you may change the quota after the volume is created.

Similarly, the default access control list (ACL) associated with the root directory of a new volume grants ALL rights to *system:administrators*. In step 7 above, you granted the user full access to the home directory. You may also remove *system:administrators* from the ACL, change the rights granted to it, and/or grant rights to other groups.

⇒ To change the quota after volume creation

Step 1: Verify that you have the privilege necessary to set volume quota (you belong to *system:administrators*). If necessary, issue the **pts membership** command described fully in section 20.3.

% **pts membership system:administrators**

Step 2: Set the volume's quota.

% **fs setquota** <*dir/file path*> <*max quota in kbytes*>

where

sq is an acceptable alias for **setquota**.

directory is the user's home directory as created in step 6 above.

max quota in kbytes
sets the volume's maximum size. Specify a number of kilobyte blocks; *1024* equals one megabyte. A value of *0* grants an unlimited quota.

⇒ To change the ACL after volume creation

Step 1: Verify that you have the privilege necessary to set all ACLs (you belong to *system:administrators*). If necessary, issue the **pts membership** command described fully in section 20.3.

% **pts membership system:administrators**

Step 2: Set the user home directory's ACL. Section 19.4 describes all the available options.

fs setacl <*directory*> <*access list entries*>+

where

sa is an acceptable alias for **setacl**.

directory names the user home directory. Provide a complete pathname unless *directory* is the working directory.

access list entries
specifies one or more pairings between a user/group name and a set of access rights, in that order. Use a space to separate the name from the rights, and the pairs from one another.

To remove *system:administrators* from the ACL, specify

system:administrators none

To grant *system:administrators* the LOOKUP right only, specify

system:administrators l

You may also provide other name/rights pairings, mixing user and group names if desired.

Each set of rights is either some combination of the seven letters `rlidwka`, in any order, or one of the four "shorthand" groupings of rights (**read**, **write**, **all**, and **none**); full definitions of these appear in section 19.2.3. It is valid to put both letters and groupings in the same command, but not within a single pair.

17.6. Improving Password/Authentication Security

There are several ways to make AFS more secure against unauthorized entry. These methods (all of which are optional) and the types of unauthorized entry that they are meant to thwart are discussed below:

Limiting the number of consecutive failed login attempts

One of the most common ways that an unauthorized user might try to gain entry to AFS is to guess the password of one or more users. This method of attack is made even more dangerous if the attacker is able to use many login processes in parallel or use the RPC interfaces to gain entry.

To protect against this type of unauthorized entry, you can limit the number of times that a user can consecutively fail to type in the correct password during an authentication attempt using either AFS **login** or **klog**. If this limit is exceeded, the Authentication Server will lock the corresponding Authentication Database entry (disallowing all authentication attempts) for a period of time that you define.

The **kas setfields** command is used to set the authentication failure limit and lockout time on a user-by-user basis. System administrators can unlock a locked user account by issuing the **kas unlock** command.

Note: The mechanism used to enforce the number of failed authentication attempts may cause a user account to be locked even though the number of failed attempts is less than the permitted limit. Here is why. The Authentication Database will store the limit on failed authentication attempts (*f*); however, each Authentication Server (*S*) will only know about and keep track of *fdivS* attempts (the sync site will know about and keep track of any remainder *fmodS*). In other words, each Authentication Server will permit only a portion of the configured total failed attempts before refusing to authenticate a user. An Authentication Server will continue to refuse to authenticate a user until the time between the most recent authentication attempt and the current time exceed the configured lockout time.

Under most circumstances, this mechanism will not present a problem. If an authentication attempt is refused by one Authentication Server, another server will be contacted until either all of the available servers have been contacted or a successful authentication is made.

However, if one or more of the Authentication Server processes are unavailable, the limit on failed authentication attempts will be effectively reduced. Furthermore, if you have specified an infinite lockout time, each time a user fails to properly authenticate, the failed authentication limit itself will be effectively reduced by one (since the Authentication Server will never reset its failure counter). And this will eventually result in account lockout.

For these reasons, you should follow the recommendations listed below when setting limits on failed authentication attempts and lockout times:

- Unless you have a compelling reason, do not set an infinite lockout time. This will result in continual lockouts that will have to be manually unlocked by the system administrators. It is strongly recommended that you do not set infinite lockout times on system administrator accounts; if all of your system administrator accounts become locked, the only way to remove these locks is to shut down the Authentication Servers and remove the files containing the failure information. (See the **kas** section of the *AFS Command Reference Manual* for information on this operation.)

- Use short lockout times. Remember, although guessing passwords is a common method of attack, it is by no means a sophisticated method. In most cases, you should set lockout times to discourage attackers; however, being overzealous in this regard will be more of a burden to your users than to potential attackers.
- Do not set the limit on failed authentication attempts too low. The loss of a server or two will reduce the limit even further.

For most cells, Transarc Corporation recommends setting the limit on authentication to 5 attempts and the lockout time to 25 minutes.

Defining password lifetimes

The older a password is, the more likely that it may be known by someone who should not have access to the corresponding account. And that opens up the possibility of unauthorized use of a user's account.

To protect against this type of unauthorized entry, you can set a lifetime limit on the age of a password. A user should change his or her password prior to the expiration of the password lifetime. If a user does not change his or her password before it expires, he or she will be unable to authenticate to AFS with that password. However, for up to 30 days after this password expiration, a user will be able to use his or her old password to change passwords. Beyond this 30 day grace period, the user will have to contact a system administrator to get his or her password changed.

Note: Upon a successful authentication using AFS **login** (but not **klog**), the global environment variable (PASSWORD_EXPIRES) will be set to the number of days until the password is scheduled to expire. Zero means that the password expires today and will not be valid tomorrow. If a user's password never expires, PASSWORD_EXPIRES will not be set. This environment variable permits system administrators to customize user login scripts to

- Display the password expiration date during user login (after authentication).
- Provide the user the option to change his or her password whenever the expiration date is close at hand.

Restricting password reuse

It does little good to force a user to change his or her password if he or she simply reuses the same password again. Users who reuse old passwords open up their accounts to invasion by anyone who may have learned that password some time in the past.

To prevent a user from reusing any of his or her last twenty passwords, use the **kas setfields** command. If a user attempts to create a password that is the same as or similar to a recently used password, the following error message will be displayed:

```
kpasswd:  Password was not changed because it seems like a reused password
```

Note: To prevent users from getting around this restriction by simply changing their passwords twenty times (or running a script which automatically does this), you can set a minimum time between password changes when **kaserver** is started. Setting a minimum time between password changes is not recommended. (See the *AFS Command Reference Manual* for more information on this command.)

If a user attempts to change a password before this minimum time has elapsed, the following error message will be displayed:

```
kpasswd:  Password was not changed because you changed it too recently; see
your systems administrator
```

Checking the quality of new passwords

Some passwords are better than others. For example, if a user utilizes a word as a password rather than a random string of numbers and letters, a skilled attacker is more likely to guess that password.

To ensure the quality of a password, the **kpasswd** and **kas setpassword** commands will hand off new passwords to a program called *kpwvalid* (if it exists). Whether you use the version of the program provided by Transarc or you create your own version, the *kpwvalid* program runs the new password through its quality check(s) and returns a valid or invalid signal to **kpasswd/kas setpassword**. If a password is invalid, the user must come up with a new one. The checks made by *kpwvalid* are applied to all new passwords, whether the password is created by a regular user or a system administrator.

The actual check(s) made by a custom *kpwvalid* program are left to the discretion of each cell. Below are some suggested checks:

- Check to make sure the password is at least *n* characters long.
- Check to make sure that the password is not a word.
- Check to make sure that the password contains both numbers and letters.

See the entry on **kpasswd** or **kas setpassword** in the *AFS Command Reference Manual* for information on the general specifications of *kpwvalid* and the example version provided by Transarc Corporation.

⇒ To limit the number of consecutive failed login attempts

Step 1: Verify that you have the privilege necessary to set this limit (you have the `ADMIN` flag in your Authentication Database entry). If necessary, issue **kas examine** which is fully described in section 20.4.

% **kas examine** <*name of user*>
`Password for` *name of user*: <*your password*>

Step 2: Issue the **kas setfields** command with the **-attempts** and **-locktime** arguments.

% **kas setfields** <*name of user*>
-attempts <*maximum successive failed login tries ([0..254])*>
-locktime <*failure penalty [hh:mm or minutes]*>

where

name of user specifies the Authentication Database entry to be affected.

-attempts specifies the maximum number of times that a user can consecutively fail to correctly type in his or her password during authentication (AFS **login**, **klog**, etc.,) before being locked out of his or her user account for the amount of time specified by the **-locktime** argument. The range of valid values is 0 through 254. Use 0 to indicate that there is no failed authentication limit; this is the default value.

-locktime specifies the amount of time that a user will be locked out of his or her user account if he or she exceeds the maximum number of failed authentication attempts specified by the **-attempts** argument.

You can specify a time in either hours and minutes (hh:mm) or minutes only (mm). If a 0 is specified, the account will be locked indefinitely (until a system administrator unlocks it using **kas unlock**). If a time is entered (other than 0), it will be rounded up to the next highest increment of 8.5 minutes; so all locked accounts will be locked for some multiple of 8.5 minutes. The maximum finite value is 36 hours; for example, if *99:99* is specified, the locktime will be 36 hours.

⇒ To unlock a locked user account

Step 1: Verify that you have the privilege necessary to unlock a user account (you have the `ADMIN` flag in your Authentication Database entry). If necessary, issue **kas examine** which is fully described in section 20.4.

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

Step 2: Verify that the user's account is in fact locked by issuing **kas examine** on the account. Look for the message `User is locked until` *day/date/time* in the entry.

Step 3: Issue the **kas unlock** command.

% **kas unlock** <*Authentication ID*>
<*admin principal to use for authentication*>

where

Authentication ID
specifies the user name associated with the account that you want to unlock.

admin principal to use for authentication
specifies the user name under which the issuer wishes to perform the command. If the issuer does not provide it, the current identity is used.

⇒ To set password lifetime

Step 1: Verify that you have the privilege necessary to set password lifetimes (you have the `ADMIN` flag in your Authentication Database entry). If necessary, issue **kas examine** which is fully described in section 20.4.

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

Step 2: Issue the **kas setfields** command with the **-pwexpires** argument.

% **kas setfields** <*name of user*>
-pwexpires <*number days password is valid ([0..254])*>

where

name of user specifies the Authentication Database entry to be affected.

-pwexpires defines the number of days for which *name of user*'s password will be valid. Beyond the specified number of days, the password will be invalid (that is, the user cannot use it to authenticate with his or her account). The range of valid values is 0 through 254. Use 0 to indicate that the password never expires; use 1 through 254 to specify the number of days until expiration. By default, a user's password will never expire.

Note: The password lifetime set with the **-pwexpires** argument is retroactive. For example, if a user changed his or her password 30 days ago, and you set the password lifetime to 30 or less, the password would expire in approximately 1 day (24 hours from the time that the password was changed). To avoid retroactive expiration, you should instruct users to change their passwords prior to setting a password lifetime.

⇒ To restrict reuse of passwords

Step 1: Verify that you have the privilege necessary to set this restriction (you have the `ADMIN` flag in your Authentication Database entry). If necessary, issue **kas examine** which is fully described in section 20.4.

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

Step 2: Issue the **kas setfields** command with the **-reuse** argument.

% **kas setfields** <*name of user*>
-reuse <*permit password reuse (yes/no)*>

where

name of user specifies the Authentication Database entry to be affected.

-reuse specifies whether or not the user can reuse any of his or her last twenty passwords. The valid values for this field are *yes* (the user can immediately reuse old passwords) and *no* (the user can only reuse old passwords after twenty password changes). By default, the user can reuse passwords (*yes*).

⇒ To check the quality of new passwords

Step 1: Create the program or shell script *kpwvalid* containing the quality checks that you want all new passwords to pass or use the version supplied by Transarc Corporation. For information on the specifications of *kpwvalid* and the example program provided by Transarc, see the entry for **kpasswd** or **kas setpassword** in the *AFS Command Reference Manual*.

Step 2: Verify that the executable version of *kpwvalid* is housed in the same directory as **kpasswd** and **kas setpassword**, that the directory is in AFS file space, and that the ACL for the directory gives write permission to the *system:administrators* group only.

Step 3: The *kpwvalid* program will execute automatically when either **kpasswd** or **kas setpassword** is run. If a new password passes all of the checks in *kpwvalid*, the password is valid; if not, the user will be forced to enter a new password.

17.7. Changing Passwords

After you set an initial password during account creation, you will normally not need to change user passwords, since users can do it themselves using **kpasswd**. The *AFS User's Guide* provides step-by-step instructions.

Infrequently, you may need to set a user password if the user has forgotten it or cannot log in. You should invent a new password in the Authentication Database using **kas setpassword**. You must have the `ADMIN` flag set in your Authentication Database entry to change other users' passwords.

This section tells you how to change a user password in the Authentication Database. If your cell also maintains */etc/passwd* to control local logins, remember to change passwords there, too, as necessary.

See section 17.6 for information on restrictions on password creation.

⇒ To change a password in the Authentication Database

Step 1: Verify that you have the privilege necessary to change other users' passwords (you have the `ADMIN` flag in your Authentication Database entry). If necessary, issue **kas examine**, which is fully described in section 20.4.

`%` **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

Step 2: Change the password.

`%` **kas setpassword** <*name of user*> **[**<*new password*>**]**
`Password for` *your user name*`:` <*your password*>

where

sp is an acceptable alias for **setpassword**.

name of user names the Authentication Database entry, which should match the user name.

new password becomes the password that *user name* types at login. It should be 8 characters of less, because some programs cannot use passwords longer than that. It is not usually typed on the command line because it is visible there. If you omit it, you will be prompted for it and it will not echo visibly:

`new_password:` <*new password*>
`Verifying, please re-enter new_password:` <*new password*>

17.8. Listing and Setting Users’ Volume Quota

As mentioned in section 17.5, AFS assigns each volume a maximum quota of 5 megabytes as it is created. Subsection 17.5.2 explains how to change that default if it is unsatisfactory.

User volumes are no different than any others with respect to quota. Three commands display a volume’s quota and related information, such as its current size:

- **fs quota**
- **fs listquota**
- **fs examine**

Two commands allow you to change quota:

- **fs setquota**
- **fs setvol**

Section 5.12 for complete descriptions about how to use these commands.

17.9. Changing User Names

By convention, many components of an account incorporate the user name, including the Protection and Authentication Database entries, the volume name and the home directory name. You should maintain this consistency when changing a user name. This requires almost as many steps as creating a new account.

Note: For the sake of convenience, the user name you change from is referred to as the "old (user) name" throughout these instructions. The user name you are changing to is referred to as the "new (user) name."

⇒ To change a user name

Step 1: Verify that you have all the privileges necessary for performing all the steps:

- You belong to *system:administrators*. If necessary, issue **pts membership**, which is fully described in section 20.3.

  % **pts membership system:administrators**

- You are included in */usr/afs/etc/UserList*. If necessary, issue **bos listusers**, which is fully described in section 20.5.

  % **bos listusers** <*machine name*>

- You have the `ADMIN` flag in your Authentication Database entry. If necessary, issue **kas examine**, which is fully described in section 20.4.

  % **kas examine** <*name of user*>
  `Password for` *name of user*`:` <*your password*>

- You have the INSERT, DELETE and ADMINISTER rights in the directory where you are going to remove the old mount point and create a new one. If necessary, issue **fs listacl**, which is fully described in section 19.3.

  % **fs listacl** <*dir/file path*>

  If you are a member of *system:administrators*, you have implicit ADMINISTER rights on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: **(Optional).** List and note the groups the user owns, so you can change their names after you change the user name in the Protection Database in step 3. This is necessary because group names do not change automatically when the user name changes. Section 18.4 fully describes the **pts listowned** command.

% **pts listowned** <*user or group name or id*>

Step 3: Change the user's name in the Protection Database.

% **pts rename** <*old name*> <*new name*>

where

ren is an acceptable abbreviation of **rename**.

old name is the old user name.

new name is the new user name.

Step 4: **(Optional).** Change the group names noted in step 2 to accurately reflect the owner's new name. Change the "*owner-field*" of each group name (the part before the colon) to match the *new name* you specified in step 3. This step restores consistency to the group names; omitting it may cause confusion, but does not cause errors.

Repeat this command for each group. It is described fully in step 3.

% **pts rename** <*old name*> <*new name*>

Step 5: Enter the Authentication Server's "interactive mode" to make the next few steps more convenient.

% **kas**
`Password for` *your user name*`:` <*your password*>

Step 6: Delete the user's current Authentication Database entry.

`ka>` **delete** <*name of user*>

where

del is an acceptable abbreviation for **delete**, or you can use the alias **rm**.

name of user is the user whose Authentication Database entry you want to delete.

Step 7: Create an Authentication Database entry under the new user name. Specify a dummy password as *initial password* at the prompts. The user should replace the dummy with a truly secret password at the next login. (You could retain the current password if the user tells it to you.)

`ka>` **create** <*name of user*>
`initial_password:` <*initial password*>
`Verifying, please re-enter initial_password:` <*initial password*>

where

cr is an acceptable abbreviation for **create**.

name of user is the new user name.

Step 8: Leave interactive mode.

`ka>` **quit**

Step 9: Change the name of the user's volume to make it consistent with the new user name.

% **vos rename** <*old volume name*> <*new volume name*>

where

ren is an acceptable abbreviation of **rename**.

old volume name
is the current name of the user's volume.

new volume name
is a name incorporating the new user name. Be sure it is no longer than 22 characters, so that the addition of a *.backup* extension does not violate the absolute length restriction of 31 characters.

Step 10: Remove the mount point for the volume's old name.

% **fs rmmount** <*directory*>

where

rmm is an acceptable abbreviation of **rmmount**.

directory is the mount point for the former volume name (that is, the user's former home directory). Provide a full pathname unless *directory* is in the working directory.

Step 11: Create a mount point for the volume's new name.

% **fs mkmount** <*directory*> <*volume name*>

where

mk is an acceptable abbreviation of **mkmount**.

directory names the mount point you wish to create. Do not create a directory of the same name beforehand. Provide a complete pathname unless you want to mount the volume in the working directory. The recommended name for the final element in the pathname (the "home directory" itself) is the *user name* assigned in step 3.

volume name matches the *new volume name* you specified in step 9.

17.10. Removing User Accounts

When removing an account, you should store the user's data on a permanent medium such as tape before removal. (You can also use the **uss delete** command to remove user accounts; see section 16.7 for information on this command.)

⇒ To remove a user account

Step 1: Verify that you have all the privileges necessary for performing all the steps:

- You belong to *system:administrators*. If necessary, issue **pts membership**, which is fully described in section 20.3.

  % **pts membership system:administrators**

- You are included in */usr/afs/etc/UserList*. If necessary, issue **bos listusers**, which is fully described in section 20.5.

  % **bos listusers** <*machine name*>

- You have the `ADMIN` flag in your Authentication Database entry. If necessary, issue **kas examine**, which is fully described in section 20.4.

  % **kas examine** <*name of user*>
  `Password for` *name of user*`:` <*your password*>

- You have the DELETE right in the user home directory's parent directory. If necessary, issue **fs listacl**, which is fully described in section 19.3.

  % **fs listacl** <*dir/file path*>

  If you are a member of *system:administrators*, you have implicit ADMINISTER rights on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: **(Optional).** If you think that you may want to restore the user's account someday, note the user name and AFS UID, possibly in a file designated for that purpose. This will allow you to restore the account with the same AFS UID.

Step 3: **(Optional).** Store the contents of the user's volume on tape. See chapter 6 on the AFS Backup System to learn how.

Step 4: **(Optional).** List the groups the user owns, (if you are going to remove them from the Protection Database, too). You may not wish to remove them because other users may have the groups on their own ACLs.

% **pts listowned** <*user or group name or id*>

where

listo is an acceptable abbreviation of **listowned**.

user or group name or id
is the user's user name.

Step 5: **(Optional).** Remove the groups the user owns, (if you have decided not to leave them in the Database). You may specify the groups in the list either by name or by AFS UID.

% **pts delete** <*user or group name or id*>$^{+}$

where

del is an acceptable abbreviation for **delete**.

user or group name or id
specifies the name or AFS UID of each group you listed in step 4.

Step 6: Remove the user's Authentication Database entry.

% **kas delete** <*name of user*>

where

d is an acceptable abbreviation for **delete**.

name of user is the name of the user whose Authentication Database you want to delete.

Step 7: Note the location of the user's volume in preparation for removing it. If you follow AFS naming conventions, the volume is called `user`.*user name*.

% **vos listvldb** <*volume name or ID*>

where

listvl is an acceptable abbreviation of **listvldb**.

volume name or ID
is the volume's name or volume ID number.

The user's volume is probably not replicated, so the last line of the output should be marked `RW`. It specifies the file server machine and partition where the volume resides. The volume ID number also appears in the output.

Step 8: Remove the user's volume.

% **vos remove** <*machine name*> <*partition name*> <*volume name or ID*>

where

remo is an acceptable abbreviation of **remove**.

machine name is the file server machine name you learned in step 7.

partition name specifies the partition you learned in step 7.

volume name or ID
is the name or ID number of the volume.

This command also removes the Backup copy of the volume, if any. If there are no ReadOnly sites, the entire VLDB entry disappears. If there are ReadOnly sites remaining, you need to repeat the **vos remove** command to remove each one in turn.

Step 9: Remove the mount point that referred to the volume. This will effectively remove the user's home directory from the file tree as well.

% **fs rmmount** <*directory*>

where

rm is an acceptable abbreviation of **rmmount**.

directory is the user's home directory.

Unmounting the ReadWrite volume is sufficient to unmount the Backup as well, if you mounted the latter as a subdirectory of the former. If you mounted the Backup at a completely separate directory, you need to repeat step 9 for it explicitly.

Step 10: Remove the user's Protection Database entry. You may specify the user either by name or by AFS UID. This command is described completely in step 5 above.

% **pts delete** <*user or group name or id*>

18. Administering the Protection Database

This chapter explains how to create and maintain user and group entries in the Protection Database. Much of the information is the same as in the *AFS User's Guide*, but some is specific to system administrators. The sections include:

18.1 Summary of Commands 18-1
18.2 About the Protection Database 18-2
18.3 Listing Information about Groups 18-10
18.4 Listing Group-Related Information about Users and Machine Entries 18-13
18.5 Creating User Entries 18-16
18.6 Creating Machine Entries 18-18
18.7 Creating Regular and Prefix-Less Groups 18-21
18.8 Adding Members to Groups and Removing Them from Groups 18-25
18.9 Deleting Entries from the Protection Database 18-27
18.10 Changing the Owner of an Entry 18-28
18.11 Changing the Name of an Entry 18-30
18.12 Changing Group-Creation Quota 18-32
18.13 Listing and Setting the AFS UID Counters 18-34
18.14 Listing and Setting Protections on Database Entries 18-36

18.1. Summary of Commands

Learn a user's, machines, or group's AFS UID	**pts examine**
Learn who owns an entry	**pts examine**
Learn who created an entry	**pts examine**
Learn how many users and/or machines belong to a group	**pts examine**
Learn how many groups a user or machine belongs to	**pts examine**
List the members of a group	**pts membership**
List the groups a user or machine belongs to	**pts membership**
List the groups a user or group owns	**pts listowned**
Create a user or machine entry	**pts createuser**
Create a group entry	**pts creategroup**
Add users and machines to groups	**pts adduser**
Remove users and machines from groups	**pts removeuser**
Remove an entry from the Database	**pts delete**
Change a group's owner	**pts chown**
Change an entry's name	**pts rename**
List group-creation quota	**pts examine**
Set group-creation quota	**pts setfields**
List AFS UID counters	**pts listmax**
Set AFS UID counters	**pts setmax**
Learn how an entry is protected	**pts examine**
Set protections on an entry	**pts setfields**

18.2. About the Protection Database

The Protection Database tracks information about users, machines, and groups so that access to directories can be allowed or restricted based on the name of the user attempting the access, the group memberships of the user attempting access, and the group memberships of the machine from which access is attempted. This access protection is created by putting users and groups on directory ***access control lists*** ACLs; see chapter 19 for more on ACLs.

Every **user** who requires authenticated access to an AFS cell must have a Protection Database entry for that cell. User entries can only be created by system administrators (that is, those with membership in the group *system:administrators*); see section 18.2.5 for information on the group *system:administrators*. When a user puts a user name on an access control list (ACL), he or she grants or restricts access to that directory for the specified user.

You should only create a **machine entry** in the Protection Database if you intend to allow or restrict access to one or more directories when the access originates from that machine. A machine entry cannot be placed directly on an ACL; instead, it must be made a member of a group which can then be put on the ACL. The ability to include a machine in a protection group allows a site to take advantage of AFS replication for binary files while still adhering to a machine-based license agreement associated with those binaries. Section 18.2.4 contains general information on this usage.

Like user entries, machine entries can only be created by system administrators (that is, member of *system:administrators*). Unlike user entries, there need not be a one-to-one correspondence between a single machine and a machine entry in the Protection Database; if desired, machines with consecutive IP addresses can be "bundled" into one machine entry by using a wildcarding scheme. (For information on creating machine entries, see section 18.6.)

Groups are lists of individual users and/or machines defined by users themselves. There are also three "system-defined" groups (see subsection 18.2.5 below) that exist in every cell. When a user puts a group on an ACL, he or she grants or restricts access to that directory based on the user and/or machine membership of the group. The real benefit of groups is that they make it possible to add a user and/or machine to many ACLs at once, simply by adding the user/machine to groups that appear on that ACL; removal is just as easy.

The Protection Database is also used to

- Map each user, machine, and group entry to its corresponding AFS UID. AFS uses AFS UIDs in internal processing; however, human users prefer to use names rather than numbers. The Protection Database provides the translation mechanism.
- Record the list of groups to which each user and machine belongs, and the members of each group. The File Server uses information about group membership when it determines if users are entitled to the type of access they have requested to a file or directory.

18.2.1. Protection Database Entries

Each Protection Database entry contain the following information for users, machines, and groups:

- A ***name***. For users, this is the character string typed when logging in.

  For machines, the name is the IP address (or range of IP addresses specified with wild cards) corresponding to the machine (or group of machines).

  For groups, there are two different styles of names: ***regular*** and ***prefix-less***. Regular group names have two parts, separated by a colon. The first part, the *owner-name* field, specifies the owner of the group. The second part, the *group-name* field, is the name of the group, as chosen by the creator. In a more graphic form:

  owner-name:group-name

  Prefix-less group do not have the *owner-name* field or the colon. Only members of the *system:administrators* group are allowed to create prefix-less groups. To learn why prefix-less groups are useful, see 18.2.3.

  To learn more about the constraints on group names and how to create regular and prefix-less groups, see section 18.7. To learn how to change names, see section 18.11.

- An ***AFS UID***. This is a unique identification number that the AFS server processes use internally. It is similar in function to a UNIX UID, but operates in the AFS file system rather than the UNIX file system. Users and machines have positive integer AFS UIDs, and groups have negative integer AFS UIDs.

  Normally, the Protection Server assigns AFS UIDs automatically when creating entries. As a member of the *system:administrators* group, you can specify the AFS UID for a user, machine, or group entry as you create it, and can manipulate the counters used in automatic allocation. See sections 18.5, 18.6, and 18.7 for information on entry creation and section 18.13 for information on the counters.

- An ***owner***. This is the user or group that owns the entry (is allowed to make changes to various aspects of an entry like its ownership, name, list of groups/members, etc.,) or delete it entirely. The owner cannot necessarily change the group-creation quota, unless he or she is a member of *system:administrators*.

  As a member of the *system:administrators* group, you have total administrative control over every entry in the Protection Database, even if you are not the owner. See section 18.10 to learn how to change this field.

- A ***creator***. The name of the user who originally created the entry. This field is mainly useful as an audit trail, and cannot be changed.

  The user *anonymous* may appear here if 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. See chapter 8 for more information about "no authentication" mode.

- A ***membership count***. For users and machines, this indicates how many groups the entry belongs to. For groups, it indicates how many members belong to the group. This number cannot be set explicitly.

- A set of ***privacy flags***. This field indicates who is allowed to list certain information about the entry or change it in certain ways. As a member of *system:administrators*, you possess all the access rights on every entry in the Protection Database, no matter how they are protected. See section 18.14 for more information.

- A ***group-creation quota***. This field indicates how many groups a user is allowed to create. It is set to `20` at the time each user or machine entry is created. The creation quota is meaningless for groups and machines because it is not possible authenticate as a group or machine.

  Only members of *system:administrators* may reset a group-creation quota. The most common reason is to grant additional quota to a user who has exhausted the original quota of 20, but you may also use it to set a lower quota. See section 18.12 for details.

- A ***membership list***. For users and machines, this lists all the groups they belong to. The system-defined groups *system:anyuser* and *system:authuser* do not appear, for the reasons discussed in section 18.2.5.

  For groups, this lists all the members of the group. Groups may not be members of other groups. This information is not available for the system-defined groups *system:anyuser* and *system:authuser*, for the reasons discussed in section 18.2.5.

- A list of the ***groups owned***. This records all the group entries that this user or group owns. A group may own another group. A machine may theoretically own a group, but this provides no additional functionality and is not recommended.

The **pts examine** command lists all the information described here except the *membership list* and *groups owned*. The **pts membership** command displays the *membership list* for users, machines, and groups. The **pts listowned** command displays the *groups owned* for both users and groups. These commands are described in sections 18.3 and 18.4.

A Note About Machine Entries: In most ways, a machine entry in the Protection Database is the same as a user entry. However, the following exceptions should be kept in mind:

- Although a machine is given a group creation quota in the Protection Database, there is no way to use this quota because there is no way to authenticate as a machine and thus no way to create a group as a machine.

- Although a machine can be given ownership of a group, there is no functional purpose for this ownership because there is no way to authenticate as a machine and thus no way to administer the group as a machine.

18.2.2. Using Groups Effectively

There are three basic ways to use groups, each of which is best suited to a particular purpose. All users may find the suggested uses helpful; but system administrators may find the suggestions in section 18.2.2.3 particularly useful. Keep in mind that these are only suggestions—you are free to use groups in other ways too.

18.2.2.1. Private Use

Using a group ***privately*** means that you create a group for your own convenience and put it on your own ACLs, without necessarily informing the group's members that they belong to it. You will probably do this more often as a regular user than as a system administrator. The only thing the members notice is that they are given or denied certain kinds of access to a directory. 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. Anyone can learn that a group exists when they see it on an ACL. A user can restrict who can list the members of a group, but a member of *system:administrators* always have this right; see section 18.14.

18.2.2.2. Shared Use

In the case of ***shared use***, you inform the group's members that you have formed a group with them in it, but you still retain sole control over its membership. Users can add the group to their own ACLs, since it is not necessary to own a group to add it on an ACL (it is necessary to have the ADMINISTER right on the directory). Again, this type of use applies more to you as a regular user than as a system administrator.

When you add a group owned by someone else on your ACLs, keep in mind that the owner can change the group membership without informing you. Thus, someone new could gain (or be denied) access in a way you did not intend and without your knowledge. As a member of *system:administrators*, you can always check group membership using **pts membership**.

For example, assume a manager creates a group consisting of all the employees in the department. The employees can put the group on the directories that store department-related files, so that everyone in the department can look at them. The manager retains control over the membership of the group.

18.2.2.3. Group Use

Group use has two variations, both of them equally useful to you as a system administrator and a regular user.

- ***Group-owned***: one group owns another group, so that all members of the "owner" group can administer the "owned" group; the members of the owned group have no administer rights themselves.
- ***Self-owned***: a group owns itself, so that all the members are allowed to administer the group.

Both variations are possible because after a user initially creates a group, he or she can use the **pts chown** command to change the owner to a group. In the case of the self-owned groups, the owner group is also the owned group (see section 18.10).

Group-owned and self-owned groups make it possible to share responsibility for administering a group among several people (the members of the owner group, which in the case of a self-owned group are also the members of the owned group). A single person does not have to keep track of adding and deleting members, and if the original creator leaves the group, ownership does not have to be transferred.

Keep in mind that everyone in an owner group has the power to make changes that affect others negatively: "locking" people out of the group by removing them from the list of members, adding members that should not belong, or changing ownership back to themselves exclusively. These problems could 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.

18.2.3. Using Prefix-Less Groups

As a member of *system:administrators*, you are entitled to create a special kind of group particularly suited to group use: the ***prefix-less group***. As its name implies, a prefix-less group name does not include an *owner-name* field or colon.

The main motivation for prefix-less groups is that many organizations want to define groups of people allowed to perform certain administrative tasks. It usually seems inappropriate that these groups include an *owner-name* field, as the following example shows.

Suppose that the manager of the Accounting Department in a company, user name *smith*, creates a group to include all the employees in the department. (This group could then be placed on the ACLs of directories storing departmental records, for example). If this were a regular group, its name would be something like *smith:accting*, since regular group names have to have an *owner-name* field to indicate who owns the group. There are a couple of drawbacks to this name:

- The fact that *smith* created and owns the group is irrelevant to its purpose. What is relevant is that the group represents a department, so a prefix-less name like *accting* seems more appropriate.
- If *jones* ever replaces *smith* as manager of the Accounting Department, *jones* needs to become the new owner of *smith:accting*. Two things make this inconvenient.
  - *smith* might forget to change the owner to *jones* before leaving. If *smith* forgets, only a member of *system:administrators* will able to make the change.
  - The name of the group itself will change from *smith:accting* to *jones:accting*. However, the names of regular groups owned by *smith:accting* do not change automatically—their "*owner-name*" field remains *smith*. To keep names consistent, someone would have to change all of their names individually, replacing *smith* with *jones*.

One solution to this problem would be to create a fictional user called *accting* so that groups owned by it would have *accting* as a prefix rather than a person's name. Members of the Accounting Department would authenticate as *accting* when they need to administer these groups. However, the number of people who need to know the password for *accting* may be a superset of the people who should be making changes to groups owned by *accting*. Also, it becomes impossible to determine who made a change to a group because anyone who happens to know the password for *accting* can do so.

Prefix-less groups solve the problem of inappropriate owner names. The manager of the Accounting Department can create a group called *accting* rather than a full-fledged account complete with password. The groups owned by this group have *accting* in their owner field, which better reflects their membership than having the manager's name there.

Prefix-less groups are better than dummy accounts in terms of accountability, too. Belonging to the group gives individuals all the rights the group has on ACLs, but people continue to perform tasks under their own names rather than as the dummy user name. Even if the group owns itself, then only a finite number of people can administer the group entry. If a dummy user owned the group instead, then anyone who happened to learn the password could administer the group.

Section 18.7 describes how to create prefix-less groups.

18.2.4. Placing Machines in Groups

Although you may find other uses for "machine" groups (that is, groups containing machine entries), their main purpose is to allow replication of binaries while still enforcing machine-based license restrictions. This is how it works:

- To ensure that the binaries can only be accessed from selected client machines, Protection Database entries are made for those client machines (grouping machines into single entries where possible). The machine entries are then placed in a group and the group is placed on the ACL(s)of the directory(ies) containing the binaries (at the appropriate level of protection).
- To increase the availability of the binary files, the volumes that contain them are then replicated in the normal manner and the ReadOnly copies are placed on as many file servers as desired.

Although it is technically possible to mix machine entries with user entries in the same group, it is not recommended. In general, it is easier to use and administer non-mixed groups.

Warning: You should exercise caution when adding a machine to a group, because the access rights granted when that group is placed on an ACL applies to all users logged into that machine. This is equivalent to having *system:anyuser* on the ACL for the user(s) accessing the directory via a machine in the group on that ACL.

18.2.5. The Standard System Groups

In addition to the groups that users and administrators can create, AFS defines three standard groups. You do not need to create these groups yourself; the Protection Server does so automatically when it builds the first version of a cell's Protection Database, assigning them AFS UIDs automatically.

The three groups are

system:anyuser This group is unusual in that it has no stable membership. Instead, its membership changes as people attempt to access files in the cell’s file tree. It includes anyone from a foreign cell who is running AFS and accesses the local file tree, and people who telnet to the local cell. It also includes everyone from the local cell, authenticated or not, including people who have logged into a machine as "root" or who have telnetted.

system:authuser This group also lacks a stable membership—its membership changes as people authenticate and unauthenticate. It includes everyone who is currently authenticated (has a valid token) in the cell.

system:administrators This group includes the few users authorized to administer the cell. Members of this group have implicit ADMINISTER rights on the access control list of every directory in the system. Only members of *system:administrators* can issue privileged **pts** commands and the **fs** commands that set quota. Because of the extensive power granted to members of this group, it is recommended that you include only a few high-level administrators in this group (and not, for instance, every operator).

18.2.5.1. Determining the Membership of the System Groups

As mentioned above, the *system:anyuser* and *system:authuser* groups are unusual because they do not have a stable or user-controlled membership. It is not possible to add or remove members explicitly, nor is there anyone ever listed in their Protection Database entry’s *membership list* field. Instead, users become "members" when the Protection Server constructs a ***current protection subgroup*** or ***CPS*** for them.

When a user requests access to a file, the File Server contacts the Protection Server for a CPS, which lists all the groups to which the user belongs. In addition to user-invented groups, the Protection Server always includes *system:anyuser* on the CPS, and includes *system:authuser* if the user is authenticated. The File Server then compares the CPS to the entries on the ACL, to determine if the user, or a group to which the user belongs, is authorized to access files in the requested way.

The *system:administrators* group is similar to user-defined groups, in that it has a stable membership. Only current members of *system:administrators* can add members to it. The person who first configures a cell’s file server machine(s) should add himself or herself to this group while the Protection Server is still running in "noauthentication" mode; see the *AFS Installation Guide* for details.

18.2.5.2. Using the System Groups on ACLs

You can place system groups on access control lists just like any other groups. This provides a convenient shorthand for providing all users (or all authenticated users) with certain rights on a directory. As each AFS volume is created, the ACL on its root directory is set to grant ALL rights to *system:administrators*.

It is possible to remove *system:administrators* from access control lists, but keep in mind that members of the group have implicit ADMINISTER rights on the access control list of every directory in the file system, so they can grant themselves other rights explicitly when necessary.

18.2.6. A Note on Reusing AFS UIDs

Do not reuse the AFS UIDs of users who have left your cell permanently (to avoid the apparent waste of IDs). This is not recommended because when users and groups are removed from the Protection Database, their AFS UIDs appear on access control lists where their names used to be. If you reuse an AFS UID, it switches back to a name on access control lists; this time it is the name of the user or group newly associated with the ID. This grants the new user or group the same rights as the former possessor of the AFS UID; in most cases, you do not want this to happen.

It is possible to remove obsolete AFS UIDs from access control lists with the **fs cleanacl** command. See section 19.10.

18.3. Listing Information about Groups

You can learn several things about a group from its entry in the Protection Database by using the following commands:

- Its AFS UID or name, given the other; use the **pts examine** command.
- How it is protected; use the **pts examine** command.
- Who owns it; use the **pts examine** command.
- Who created it; use the **pts examine** command.
- How many users and/or machines belong to it; use the **pts examine** command.
- What user and machines belong to it; use the **pts membership** command.
- What groups it owns; use the **pts listowned** command.

The standard system groups *system:anyuser* and *system:authuser* do not appear in the output from these commands.

You can also examine group-related information about particular users, such as what groups they own. See section 18.4.

⇒ To examine a Protection Database group entry

Step 1: Verify that you have the privilege necessary to examine a group's entry no matter how the first ("`s`") privacy flag associated with it is set (that is, you belong to *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in section 18.4.

`%` **pts membership** <*user or group name or id*>

Step 2: Examine the group's Protection Database entry. You may specify an entry either by name or AFS UID.

`%` **pts examine** <*user or group name or id*>

where

e is an acceptable abbreviation of **examine**.

user or group name or id
specifies the name or AFS UID of each group entry you want to examine.

The output has several fields. Section 18.2.1 explains them in more detail. An example below shows fields filled in.

- `Name`. Lists the group's name, confirming that the group does exist in the Protection Database.

- `id`. Specifies the group's AFS UID. It should be negative because this is a group entry. (If it is positive this a user or machine entry).
- `owner`. Indicates which user or group owns the entry. The user or all members of the group may administer the entry. Section 18.10 explains how to change this field.
- `creator`. Indicates which user created the entry in the first place. This information is useful mostly as an audit trail. If this group is ever deleted from the Protection Database, the creator's group creation quota will increase by one, even if he or she no longer owns the group. See section 18.12 for more information.
- `membership`. Indicates how many users and/or machines belong to the group.
- `flags`. Indicates how the entry is protected. The default setting is `S-M--`. Section 18.14 explains the meaning of all the letters and symbols in this field.
- `group quota.` This field has no meaning for groups, since they cannot create other groups. It should always be `0` for them.

An example for the group called *terry:friends*:

```
% pts examine terry:friends
Name: terry:friends, id: -567, owner:terry, creator: terry,
  membership: 12, flags: SOm--, group quota: 0
```

⇒ To list the members of a group

Step 1: Verify that you have the privilege necessary to list the members of a group no matter how the third ("`m`") privacy flag associated with the group is set (that is, you belong to *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in the next step and in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: List the group's members. You may specify each group either by name or by AFS UID.

% **pts membership** <*user or group name or id*>

where

m is an acceptable abbreviation of **membership**.

user or group name or id
specifies the name or AFS UID of each group whose membership you wish to list.

An example for the group *terry:friends*:

```
% pts mem terry:friends
Members of terry:friends (id: -567) are:
  pat
  smith
  johnson
```

⇒ To list the groups that a group owns

Step 1: Verify that you have the privilege necessary to list the groups a group owns, no matter how the second ("`o`") privacy flag associated with the group is set (that is, you belong to *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: List the groups owned by each group. You may specify each group by either its name or its AFS UID.

% **pts listowned** <*user or group name or id*>**+{+}**

where

listo is an acceptable abbreviation of **listowned**.

user or group name or id
specifies the name or AFS UID of each group whose owned groups you want to list.

An example for the group *terry:friends*:

```
% pts listo terry:friends
Groups owned by terry:friends (id: -567) are:
  terry:pals
  terry:buddies
```

18.4. Listing Group-Related Information about Users and Machine Entries

You can learn several things about a user or machine from its entry in the Protection Database by using the following commands:

- The name (IP address for machines) or AFS UID, given one of the two; use the **pts examine** command.
- The number of groups the user/machine belongs to; use the **pts examine** command.
- How the entry is protected; use the **pts examine** command.
- The number of additional groups that the user can create (not possible for a machine); use the **pts examine** command.
- The groups the user/machine belongs to; use the **pts membership** command.
- The groups the user owns (not recommended for a machine); use the **pts listowned** command.

⇒ To examine a Protection Database entry for a user or machine

Step 1: Verify that you have the privilege necessary to examine a user entry no matter how the first (`"s"`) privacy flag associated with the entry is set (that is, you belong to *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described in the next instruction.

`%` **pts membership** <*user or group name or id*>

Step 2: Examine the Protection Database entry. You may specify an entry either by name (IP address for machines) or AFS UID.

`%` **pts examine** <*user or group name or id*>

where

e is an acceptable abbreviation of **examine**.

user or group name or id
specifies the name (IP address for machines) or AFS UID of each user or machine entry you want to examine.

The output has several fields. The following mentions only the relevant ones; section 18.2.1 explains all of them in more detail.

- `Name`. Lists the user's name or the machine's IP address, confirming that the user/machine does exist in the Protection Database.
- `id`. Specifies the user's/machine's AFS UID. It should be positive because this is a user/machine entry. (If it is negative, this is a group entry.)

- `membership`. Indicates how many groups the user/machine belongs to.
- `flags`. Indicates how the entry is protected. The default setting is `S----`. Section 18.14 explains the meaning of all the letters and symbols in this field.
- `group quota.` Indicates how many more groups the user may create; this field has no meaning for machines. Its default setting as the entry is created is `20`. See section 18.12 to learn more.

The following is an example for the fictional user *terry*:

```
% pts ex terry
Name: terry, id: 5347, owner: system:administrators, creator: anonymous,
  membership: 7, flags: S----, group quota: 14.
```

The following is an example for the machine Sebastian in the cell *transarc.com* with the IP address 158.98.4.10:

```
% pts ex 158.98.4.10
Name: 158.98.4.10, id: 5151, owner: system:administrators, creator: admin,
  membership: 1, flags: S----, group quota: 20.
```

⇒ To list the groups a user or machine belongs to

Step 1: Verify that you have the privilege necessary to check which groups a user belongs to no matter how the third ("`m`") privacy flag associated with the entry is set (that is, you belong to *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in the next step.

% **pts membership** <*user or group name or id*>

Step 2: List the groups to which the user or machine belongs. You may specify each user or machine entry either by name (IP address for machines) or by AFS UID.

% **pts membership** <*user or group name or id*>

m is an acceptable abbreviation of **membership**.

user or group name or id
specifies the name (IP address for machines) or AFS UID of each user or machine whose membership you wish to list.

An example for the fictional user *terry*:

```
% pts mem terry
Groups terry (id: 5347) is a member of:
  pat:friends
  sales
  accting:general
```

⇒ To list the groups that a user owns

Step 1: Verify that you have the privilege necessary to list which groups a user owns no matter how the second ("`o`") privacy flag associated with the entry is set (that is, you belong to *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in the previous instruction.

% **pts membership** <*user or group name or id*>

Step 2: List the groups owned by each user. You may specify each user either by name or by AFS UID.

% **pts listowned** <*user or group name or id*>

where

listo is an acceptable abbreviation of **listowned**.

user or group name or id
specifies the name or AFS UID of each user whose owned groups you want to list.

An example for the fictional user *terry*:

```
% pts listo terry
Groups owned by terry (id: 5347) are:
  terry:friends
  terry:co-workers
```

18.5. Creating User Entries

An entry in the Protection Database is the first component you create for an AFS user account. You can create one or more user entries at a time. As you create the entry, the Protection Server assigns the user an AFS UID (or you can specify one yourself). It is normally best to let the Protection Server assign the next available AFS UID, which is one greater than the current value of the "`UserMax`" counter. If you do specify an AFS UID, be aware that doing so can change the value of `UserMax`; see section 18.13, page 18-34.

In addition to the AFS UID, several other fields are set automatically in a new user entry:

- *system:administrators* is recorded in the owner field, indicating its members administer the entry.
- The user name of the creator of the entry is recorded in the creator field. This is useful mainly as an audit trail.
- The membership field is set to `0`, because the new user does not yet belong to any groups.
- The entry's privacy flags are set to `S----`. See section 18.14 for an explanation.
- The group creation quota is set to `20`, meaning that the user may create 20 groups. As a system administrator, you may increase or decrease this number at any time. See section 18.12.

Note: It is recommended that you create Protection Database user entries only in the context of user creation, as described in section 17.5. This section appears here only for the sake of completeness. Without the other components of an account (an Authentication Database entry, in particular), the user will not be able to do any work anyway.

⇒ To create a single user entry in the Protection Database

Step 1: Verify that you have the privilege necessary for creating user entries (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Create the user entry.

% **pts createuser** <*user name*> **[**<*user id*>**]**

where

cu is an acceptable alias for **createuser**.

user name will become the user's user name. AFS allows any length up to 63 characters, but not the inclusion of uppercase letters. It is recommended that you keep user names shorter than 8 characters because many UNIX programs cannot use longer names. Also, do not include any characters considered special in UNIX, particularly the colon and period.

user id is optional and not recommended. If provided, it becomes the user's AFS UID. If not provided, the Protection Server assigns the next available one. See section 18.13 for more information.

⇒ To create multiple user entries simultaneously

Step 1: Verify that you have the privilege necessary for creating user entries (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Create the user entries. Include the indicated switches if you specify both *user name*s and *user id*s.

% **pts createuser -name** <*user name*>$^+$ **[-id** <*user id*>$^+$**]**

where

cu is an acceptable alias for **createuser**.

user name specify each user's user name, the character string typed when logging in. AFS allows any length up to 63 characters, but not the inclusion of uppercase letters. It is recommended that you keep user names shorter than 8 characters because many UNIX programs cannot use longer names. Also, do not include any characters considered special in UNIX, particularly the colon and period.

user id is optional and not recommended. If you do provide *user id*s, they are matched respectively with *user name*s (the first *user name* gets the first *user id* and so on). It is best to provide an equal number of each, but if you provide more *user name*s than *user id*s, then the Protection Server allocates AFS UIDs to the *user name*s at the end of the list. In the reverse situation, the excess *user id*s are ignored.

18.6. Creating Machine Entries

You should create a machine entry in the Protection Database only if you intend to put that machine entry in a group to allow or restrict access to a directory based on the machine origin of an access request. You cannot place a machine's IP address directly on an ACL in order to allow or restrict access to the directory from that machine; and you cannot place the machine's name on an ACL, either. Instead, you must create an entry for the machine's IP address in the Protection Database and then make that IP address a member of a group. Then, by putting the group on an ACL, you can allow or restrict access to the directory from that machine.

There does not need to be a one-to-one correspondence between a single machine and a machine entry in the Protection Database. If desired, you can group machines into a single machine entry in the Protection Database (as long as the machines have consecutive IP addresses).

AFS provides a convenient way to "bundle" multiple IP addresses into a single Protection Database entry. The number **0** included in an entry for an IP address in the Protection Database acts as a wildcard by matching all possible numbers in the IP address for that part of the address in which it is placed. Thus, an entry can be defined and included in a group to represent all machines from a network or subnet.

The following list describes the use of the wildcard in an entry for an IP address. An **X** is used to indicate a specific number from an IP address:

- **X.X.X.0** matches all machines whose IP addresses match the first three numbers in the specified address. For example, the address **142.2.50.0** matches both **142.2.50.119** and **142.2.50.120**, but does not match **142.2.55.86**.
- **X.X.0.0** matches all machines whose IP addresses match the first two numbers in the specified address. For example, the address **142.2.0.0** matches both **142.2.50.119** and **142.2.55.86**, but does not match **142.4.30.95**.
- **X.0.0.0** matches all machines whose IP addresses match the first number in the specified address. For example, the address **142.0.0.0** matches both **142.2.50.119** and **142.4.30.95**, but does not match **158.98.4.10**.

You cannot define an entry in the Protection Database that consists of four wildcards **0.0.0.0**.

It is normally best to let the Protection Server assign the next available AFS UID to a machine entry, which is one greater than the current value of the "`UserMax`" counter. If you do specify an AFS UID, be aware that doing so can change the value of `UserMax`; see section 18.13.

In addition to the AFS UID, several other fields are set automatically in a new machine entry:

- *system:administrators* is recorded in the owner field, indicating its members administer the entry.
- The user name of the creator of the entry is recorded in the creator field. This is useful mainly as an audit trail.

- The membership field is set to `0`, because the new machine entry does not yet belong to any groups.
- The entry's privacy flags are set to `S----`. See section 18.14 for an explanation.
- The group creation quota is set to `20`; however, this has no meaning because it is not possible to authenticate as a machine and therefore it is not possible to create groups as a machine. See section 18.12.

⇒ To create a single machine entry in the Protection Database

Step 1: Verify that you have the privilege necessary for creating machine entries (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Create the machine entry.

% **pts createuser** <*user name*> **[**<*user id*>**]**

where

cu	is an acceptable alias for **createuser**.
user name	specifies the IP address (or range of IP addresses specified using a wildcard) corresponding to the machine (group of machines) for the entry you want to create.
user id	is optional and not recommended. If provided, it becomes the machine's AFS UID. If not provided, the Protection Server assigns the next available one. See section 18.13 for more information.

The following creates a machine entry for some of the machines in the Transarc Corporation subnet:

% **pts cu 158.98.0.0**

⇒ To create multiple machine entries simultaneously

Step 1: Verify that you have the privilege necessary for creating user entries (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Create the machine entries

% **pts createuser -name** <*user name*>$^{+}$ **[-id** <*user id*>$^{+}$**]**

where

cu	is an acceptable alias for **createuser**.

user name specifies the IP address (or range of IP addresses specified using a wildcard) corresponding to the machine (group of machines) for entry you want to create.

user id is optional and not recommended. If provided, it becomes the machine's AFS UID. If not provided, the Protection Server assigns the next available one. If you do provide AFS UIDs, they are matched respectively with each machine IP address (the first AFS UID gets the first machine IP address, and so on). It is best to provide an equal number of each, but if you provide more IP addresses than AFS UIDs, the Protection Server allocates AFS UIDs to the IP addresses at the end of the list. In the reverse situation, the excess AFS UIDs are ignored.

18.7. Creating Regular and Prefix-Less Groups

Before you can add members to a group, you must create the group entry itself. The instructions in this section tell you how to create both regular groups and prefix-less groups. Only members of *system:administrators* are allowed to create the latter type. Section 18.2.3 discusses why prefix-less groups are administratively useful. You may create either one group at a time or many. Instructions for both methods appear below.

18.7.1. Rules for Group Names and Ownership

There are a few constraints on the form of group names and on who may own groups. Violation of the constraints results in an error message and failure to create the group.

Group names may contain lowercase letters, numbers, and punctuation, but not uppercase letters or a colon, which is reserved for separation of the two parts of regular group names (see the next rule). Names may be up to 63 characters long.

Specific naming constraints include:

- ***Regular group names*** have two parts separated by a colon, an owner field and a name field:

  owner-name:group-name

  When a regular group is created, the *owner-name* must match the actual owner of the group. If the owner is an individual user, *owner-name* must be his or her user name. If the owner is a regular group, *owner-name* must match the owner field of the owner group's name. (For example, if the group *terry:associates* owns any groups, their owner fields must also be *terry*.) If the owner is a prefix-less group, *owner-name* must be the group's name.

  The *group-name* field may include any lowercase letters, numbers, and punctuation except the colon, which is reserved as a separator between this field and *owner-name*. Short *group-name*s are preferable because they must always be typed in full, but they should also give some indication of the nature of the group. Use meaningful names— it is difficult to remember which group is which if they all have names like *terry:1*, *terry:2*, and so on.

  Regular group names should not contain UNIX special characters other than the colon.

- ***Prefix-less group names***, as their name suggests, have only

  group-name

  It should obey the rules given above for the *group-name* part of regular group names. Prefix-less group names should not contain UNIX special characters, particularly the colon and period.

AFS UIDs are normally allocated automatically by the Protection Server, but you can choose them yourself. Group AFS UIDs are negative numbers, so the next available ID will be one less (more negative) than the current value of the "`GroupMax`" counter. If you do specify an AFS UID, be aware that doing so can change the value of `GroupMax`; see section 18.13.

Group owners may be individual users, regular groups, and prefix-less groups. When you create a group, you become its owner automatically unless you specify a different owner with the optional **-owner** argument. Any user may use the **-owner** argument, but the `creator` field in the group's entry indicates who actually created the group. A group must already have members in order to become the owner of another group, so you cannot create a self-owned group in one step; see the complete instructions below.

Creators of groups are recorded automatically and cannot be changed. The purpose of this field is to provide an audit trail. The record always shows who issued the creation command, even if they specified a different owner at the time.

⇒ To create a single regular group entry

Step 1: Verify that you have the privilege necessary for setting AFS UIDs (membership in *system:administrators*), if you plan to do so. If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

`%` **pts membership** <*user or group name or id*>

Step 2: Create the group.

`%` **pts creategroup** <*group name*> **[**<*owner of the group*>**]**
[<*id (negated) for the group*>**]**

where

cg is an acceptable alias for **creategroup**.

group name specifies the group to be created. It must have the form *owner-name:group-name* to reflect the ownership of the group. The exact form of *owner-name* depends on whether a user or group is to own this group. See the rules for group names outlined in section 18.7.1 above.

owner of the group
is optional. It is necessary only if the owner will be someone other than the issuer of the command. If provided, it determines the form of the *owner-name* field in the *group name*. It may be any user or any group that already has members. If you want the group to own itself, see the instructions below.

id (negated) for the group
is optional and not recommended. If provided, it becomes the group's AFS UID. Precede it with a hyphen (-) to indicate that it is negative. Providing this argument may affect the value of the `GroupMax` counter; see section 18.13. You must belong to *system:administrators* to use this argument.

⇒ To create a single prefix-less group entry

Step 1: Verify that you have the privilege necessary for creating prefix-less groups or setting AFS UIDs (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Create the group.

% **pts creategroup** <*group name*> **[**<*owner of the group*>**]**
[<*id (negated) for the group*>**]**

where

cg is an acceptable alias for **creategroup**.

group name specifies the prefix-less group to be created. See the rules for group names outlined in section 18.7.1 above.

owner of the group
is optional. It is necessary only if the owner will be someone other than the issuer of the command. It may be any user or any group that already has members. If you want the group to own itself, see the instructions below.

id (negated) for the group
is optional and not recommended. If provided, it becomes the group's AFS UID. Precede it with a hyphen (-) to indicate that it is negative. Providing this argument may affect the value of the `GroupMax` counter; see section 18.13.

⇒ To create multiple groups at one time (regular or prefix-less)

Step 1: Verify that you have the privilege necessary for creating prefix-less groups or setting AFS UIDs (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Create the groups. Include the arguments shown. All of the groups will have the same owner.

% **pts creategroup -name** <*group name*>$^+$ **[-owner** <*owner of the group*>**]**
[-id <*id (negated) for the group*>$^+$**]**

where

cg is an acceptable alias for **creategroup**.

group name is the name of each group you wish to create. If any are regular groups, then the *owner-name* field of each must reflect the true owner. See the rules about group names in section 18.7.1.

owner of the group
is optional. It is necessary only if the owner will be someone other than the issuer of the command. It may be any user or any group that already has members. All the groups will have the same owner. If you want any of the groups to own themselves, see the instructions below.

id (negated) for the group
is optional and not recommended. If you provide any *id (negated) for the group*s, they are matched respectively with *group name*s (the first *group name* gets the first *id (negated) for the group* and so on). It is best to provide an equal number of each, but if you provide more *group name*s than *id (negated) for the group*s, then the Protection Server allocates AFS UIDs to the *group name*s at the end of the list. In the reverse situation, the excess *id (negated) for the group*s are ignored.

⇒ To make a new group own itself

Step 1: Create the group using one of the three methods described above in this section. Do not specify an alternate owner unless it is a group to which you already belong. You must initially own this group in order to make it own itself.

Step 2: Add members to the group (remembering that groups may not belong to other groups). Section 18.8 describes the **pts adduser** command fully. The **-user** and **-group** switches are necessary if you specify more than one *user name*.

% **pts adduser -user** <*user name*>$^+$ **-group** <*group name*>$^+$

Step 3: Make the group own itself. Section 18.10 describes the **pts chown** command fully.

% **pts chown** <*group name*> <*new owner*>

18.8. Adding Members to Groups and Removing Them from Groups

Newly created groups have no members (users and machines) in them. The **pts adduser** command described in this section adds users and machines to groups and the **pts removeuser** command removes them from groups. When adding members to a group, remember that groups cannot be members of other groups.

There are four addition/removal scenarios; you can

- Add/remove a single user/machine to/from a single group.
- Add/remove a single user/machine to/from many groups.
- Add/remove many users/machines to/from a single group.
- Add/remove many users/machines to/from many groups.

If you specify multiple users or machines, they all become members of each group you specify. If you specify multiple groups, then each user or machine you specify becomes a member of all the groups. You may find it less confusing to specify only one user/machine or group at a time.

Note: Although it is technically possible to mix machine entries with user entries in the same group, it is not recommended. In general, it is easier to use and administer non-mixed groups.

⇒ To add users and machines to groups

Step 1: Verify that you are entitled to add members to each group. The fourth ("`a`") privacy flag associated with the entry determines who may add members. If necessary, examine the flags with the **pts examine** command described fully in section 18.3.

% **pts examine** <*user or group name or id*>+

The owner of the entry (also visible with **pts examine**) and members of *system:administrators* always have the necessary privilege.

Step 2: Add one or more members to one or more groups. The **-user** and **-group** switches are necessary if you specify more than one *user name*.

% **pts adduser -user** <*user name*>+ **-group** <*group name*>+

where

add is an acceptable abbreviation for **adduser**.

user name specifies each user and/or each machine IP address to become a member of each *group name*. Remember that groups may not belong to other groups. (To add bundled machine entries to a group, use the wildcarded IP addresses that have entries in the Protection Database; see section 18.6 for information on creating these entries.)

group name specifies each group to which new members should be added.

⇒ To remove users and machines from groups

Step 1: Verify that you are entitled to remove members from each group. The fifth ("r") privacy flag associated with the entry determines who may remove members. If necessary, examine the flags with the **pts examine** command described fully in section 18.3.

% **pts examine** <*user or group name or id*>$^+$

The owner of the entry (also visible with **pts examine**) and members of *system:administrators* always have the necessary privilege.

Step 2: Remove one or more members from the one or more groups. The **-user** and **-group** switches are necessary if you specify more than one *user name*. At the command shell prompt, type

% **pts removeuser -user** <*user name*>$^+$ **-group** <*group name*>$^+$

where

rem	is an acceptable abbreviation for **removeuser**.
user name	specifies each user or each machine IP address you wish to remove from each *group name*. (To remove bundled machine entries from a group, use the wildcarded IP addresses that have entries in the Protection Database.)
group name	specifies each group from which the indicated members should be removed.

18.9. Deleting Entries from the Protection Database

You use the **pts delete** command to delete user, machine, and group entries from the Protection Database.

- When a user entry is deleted, its name on ACLs is replaced by its AFS UIDs, but its name is removed from group memberships. The user will be unable to perform even basic functions, even if he or she can still log into the system, because the servers can no longer translate the user name found in the server ticket into the AFS UID used internally.
- When a machine entry is deleted, its IP address is removed from group memberships.
- When a group entry is deleted, its name on ACLs is replaced by its AFS UID. The group-creation quota of the user who originally created the entry increases by one, even if he or she no longer owns the group.

Leftover AFS UIDs on ACLs can be removed by using the **fs cleanacl** command; see section 19.10.

⇒ To delete user and group entries

Step 1: Verify that you have the privilege necessary for removing any kind of entry from the Protection Database (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is fully described in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Delete each user, machine, or group entry from the Protection Database. Specify each entry either by name (IP address for machines) or by AFS UID.

% **pts delete** <*user or group name or id*>$^{+}$

where

del is an acceptable abbreviation for **delete**.

user or group name or id
specifies the name (IP address for machines) or AFS UID of each user, machine, or group entry you want to remove from the Protection Database.

18.10. Changing the Owner of an Entry

The owner of a Protection Database entry is the only one who may administer it, so the main reason to change entry ownership is to change responsibility for it. The information in this section applies mostly to groups, because it is not recommended that you change a user or machine entry's owner from the default *system:administrators* assigned at the time of creation. Section 18.2.2 explains that both users and groups may own groups, and the consequences of each type of ownership.

18.10.1. The Name-Changing Implications of Changing Ownership

Section 18.7.1 specifies that the *owner-name* prefix of a regular group's name (the part before the colon) must accurately reflect its owner:

- If the owner is an individual, *owner-name* must be the user name.
- If the owner is a regular group, *owner-name* must match the *owner name* of the owner group.
- If the owner is a prefix-less group, *owner-name* must be the owner group's name.

When you change a regular group's owner, the Protection Server automatically changes the *owner-name* field of its name appropriately. Changing a regular group's owner may have the effect of changing the group's name, too. For example, when the owner of *terry:friends* changes from *terry* to *pat*, its name also changes to *pat:friends*. When group names change in this way, they change automatically on ACLs.

The name-changing effect does not "trickle down" to any groups that the group may own, however. Their *owner-name* fields remain the same as before. If you want their names changed to reflect the change in their owner's name, you must use the **pts rename** command described in section 18.11.

Suppose the group *terry:friends* owns the group *terry:pals*. *terry:friends* changed to *pat:friends* (because *pat* became the new owner), *terry:pals*'s kept its original name. If you want *terry:pals* to reflect its true ownership, you could use **pts rename** to change it to *pat:pals*.

To learn how to change the "name field" of a regular group's name, or a prefix-less name, see section 18.11.

⇒ To change the owner of a group entry

Step 1: Verify that you have the privilege necessary for changing the owner of any entry (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: **(Optional.)** If you are making one group own another (or itself) and want to retain administrative privilege on the owned group, verify that you belong to the owner group. If necessary, issue **pts membership**, fully described in section 18.3.

% **pts membership** <*user or group name or id*>

If you do not, add yourself with **pts adduser**, fully described in section 18.8.

% **pts adduser** <*user name*> <*group name*>

Step 3: Change the group's owner.

% **pts chown** <*group name*> <*new owner*>

where

group name is the present name of the group.

new owner is the user name or complete group name of the new owner.

18.11. Changing the Name of an Entry

It is possible to change the name of both user and group entries in the Protection Database with the **pts rename** command. Since a machine IP address maps directly to an actual network address, it cannot be renamed. You will probably not change user names often, as it requires several corresponding name changes—to the user's Authentication Database entry, the name of his or her home directory, volume, and mount point (if you want those consistent with the new name), etc. Section 17.9 outlines all the necessary changes.

The information in this section applies mostly to groups. Section 18.10 mentioned one reason you might want to change a group's name: changing a regular group's owner also changes its *owner-name* prefix, but does not change the *owner-name* field of any groups owned by that group. To make their names reflect their true ownership, you should change them using **pts rename**.

You must own a group or be a member of *system:administrators* to change a group name. Keep the following in mind as you change group names with **pts rename**:

- **Regular groups**: The *owner-name* part of the new name must continue to reflect the true ownership of the group.

  To continue the example from section 18.10: the group *terry:pals* is now owned by the group called *pat:friends* (*pat:friends* became its name when the owner of *terry:friends* changed to *pat*). For consistency's sake, you could change *terry:pals* to *pat:pals* with the **pts rename** command. You could not, however, change *terry:pals* to *smith:pals*, because *smith* does not own *pat:friends*. Nor could you change *terry:pals* to *terry:buddies*, because that would not reflect the true owner of the group (which is now *pat:friends*).

  As a member of *system:administrators*, you can also change a regular group name to a prefix-less one. As with name changes resulting from owner changes to regular groups, this type of change does not "trickle down" to regular groups owned by this group.

- **Prefix-less groups**: Members of *system:administrators* can change the name of a prefix-less group to another prefix-less name, or to a regular name whose *owner-name* field is consistent with the actual owner. As with other name changes, the *owner-name* field of any regular groups owned by the changed group does not change.

  As a member of *system:administrators*, you can also change a prefix-less group name to a regular one. The *owner-name* field of the new name must accurately reflect the ownership.

⇒ To change the name of a group entry

Step 1: Verify that you have the privilege necessary for changing the name of any entry (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Change the group's name.

% **pts rename** <*old name*> <*new name*>

where

ren	is an acceptable abbreviation of **rename**.
old name	is the current name of the group.
new name	is the new name. If it is a regular group name, the *owner-name* field must match the actual owner.

18.12. Changing Group-Creation Quota

AFS limits the number of groups a user may create, to prevent abuse of system resources. The default "group quota" granted to a user when his or her Protection Database entry is created is `20`. This number indicates how many more groups the user may create. As a member of *system:administrators*, you may increase or decrease this number as necessary. The group creation quota for groups and machines is meaningless because it is not possible to authenticate as a group or machine.

⇒ To examine group-creation quota

Step 1: Examine the Protection Database entry for each user. You may specify each entry either by name or by AFS UID.

% **pts examine** <*user or group name or id*>

where

user or group name or id
specifies the name or AFS UID of each user entry you want to examine.

The quota appears in the last field in the output, following the `group quota:` label. For instance, the following output indicates that *terry* can create 14 more groups:

```
% pts examine terry
Name: terry, id: 5347, owner: system:administrators, creator: anonymous
  membership: 7, flags: S----, group quota: 14.
```

⇒ To change group-creation quota

Step 1: Verify that you have the privilege necessary for changing group creation quota (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Set the group-creation quota to the number of groups each user may create. You may identify users either by name or AFS UID. The same quota applies to each user.

% **pts setfields -nameorid** <*user or group name or id*>+
[-groupquota <*set limit on group creation*>**]**

where

setf is an acceptable abbreviation for **setfields**.

user or group name or id
specifies the name or AFS UID of each user whose group creation quota you are changing.

set limit on group creation
specifies how many more groups each user may create. Note that it replaces, not increments, the current value.

18.13. Listing and Setting the AFS UID Counters

The sections on creating user, machine, and group entries (18.5 18.6, and 18.7) mentioned that the Protection Server normally allocates AFS UIDs automatically for both users and groups, keeping track of the next available ID by means of two counters: `UserMax` and `GroupMax`. They also mentioned that members of *system:administrators* may assign any AFS UID not already in use to a new entry (with the **-id** argument on the creation commands).

One reason to specify AFS UIDs as you create user and group entries is that you want to set up separate "ID spaces" that correspond to real-world divisions among people. (For example, IDs from 300-399 for one department in a company, 400-499 for another department, etc.)

Explicitly assigning AFS UIDs can affect the two counters, as follows (recall that `UserMax` counter is a positive number and the `GroupMax` counter a negative one):

- If the assigned AFS UID for a user or machine is larger than `UserMax`, then `UserMax` is changed to the assigned AFS UID. The Protection Server never automatically assigns the IDs between the old value of the counter and the newly adjusted one unless the administrator moves the counter back to its old value. There is no change in the counter in the case where the explicitly assigned value is smaller than `UserMax`.
- If the assigned AFS UID for a group is more negative than `GroupMax`, then `GroupMax` is changed to the assigned AFS UID. The Protection Server never automatically assigns the IDs between the old value of the counter and the newly adjusted one unless the administrator moves the counter back to its old value. There is no change in the counter in the case where the explicitly assigned value is less negative than `GroupMax`.

You can examine the present value of the two counters with **pts listmax**. You can adjust them to a desired value with **pts setmax**.

Note: Both the `UserMax` and `GroupMax` counters indicate the last AFS UID assigned. The next user or machine entry created will get the ID one greater than `UserMax`, and the next group entry created will get the ID one less (one more negative) than `GroupMax`.

⇒ To list the `UserMax` and `GroupMax` counters

Step 1: List the counters.

`%` **pts listmax**

where **listm** is an acceptable abbreviation of **listmax**.

An example output:

```
% pts listmax
Max user id is 5438 and max group id is -468.
```

In this case, the next automatically assigned IDs would be 5439 for users and -469 for groups.

⇒ To set `UserMax` and/or `GroupMax`

Step 1: Verify that you have the privilege necessary for changing the counters (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Set either one counter or both by choosing the appropriate argument(s).

% **pts setmax [-group** <*group max*>**] [-user** <*user max*>**]**

where

setm	is an acceptable abbreviation for **setmax**.
group max	is one less negative than the AFS UID you wish to be assigned to the next group entry created. Precede the number with a hyphen to show that it is negative.
user max	is one less than the AFS UID you wish to be assigned to the next user or machine entry created.

18.14. Listing and Setting Protections on Database Entries

It is possible to limit the kind of information that users obtain about entries in the Protection Database, by setting the privacy flags associated with each entry. Regular users control access to the group entries they own and to their own entry. Members of *system:administrators* may set privacy flags on any entry in the Database.

There are five privacy flags, each with a corresponding letter. Each flag is used to allow or restrict the performance of specific actions on the entry as follows:

STATUS **(s)**: list status information about an entry using **pts examine**.

OWNED **(o)**: list the groups a user or group owns using **pts listowned**.

MEMBERSHIP **(m)**: list the groups a user or machine belongs to, or which users or machines belong to a group, using **pts membership**.

ADD **(a)**: add a user or machine to a group using **pts adduser**. Although this flag applies sensibly only to group entries, it must also be set for user and machine entries.

REMOVE **(r)**: remove a user or machine from a group using **pts removeuser**. Although this flag applies sensibly only to group entries, it must also be set for user and machine entries.

There are 5 "slots" in the list of rights, one for each letter. Three variations of the appropriate letter may appear in each slot, each representing a different set of people who possess the right:

- A ***hyphen*** in the slot indicates that only the owner and members of *system:administrators* have the access right. For user entries, the user also has the access right.
- A ***lowercase letter*** in the slot indicates that members of the group have the access right. This applies meaningfully to group entries only.
- An ***uppercase letter*** in the slot indicates that everyone has the access right. This is equivalent to *system:anyuser* on standard directory access control lists.

Consider an example list "`SOmar`" associated with a group. The capital `S` and `O` mean that anyone can examine the group's entry and list the groups that it owns. The lowercase `mar` mean that only members of the group may list, add, or remove its members, respectively.

18.14.1. Default Rights

The default rights for user and machine entries are "`S----`," meaning that all users can obtain status information about the entry. However, only the user and members of *system:administrators* can perform other functions on a user entry; and only members of *system:administrators* can perform other functions on a machine entry.

The default rights for group entries are "`S-M--`," meaning that all users can obtain status information and list the members of the group.

18.14.2. Constraints on Privacy Flags

There are several constraints on how you set the flags:

- It is always necessary to specify a value for all five flags.
- Do not set the flags in user or machine entries to lowercase letters (other than the first flag); they have no sensible interpretation because users and machines do not have members. The hyphen is equivalent to lowercase letters for user and machine entries. The hyphen is equivalent to lowercase letters for self-owned group entries, too, since all the members of a self-owned group are owners of the group itself.
- Legal values for the first flag ("check status") are lowercase `s` and uppercase `S`. It is not possible to set the first flag to the hyphen because the command interpreter cannot distinguish an initial hyphen from the indicator of a switch or flag. For user and machine entries, the lowercase *s* is equivalent to the hyphen.
- Legal values for the second flag ("list owned") are the hyphen and uppercase `O`. For groups, the lowercase *o* is equivalent to the hyphen, because members of a group are automatically the owners of any groups owned by the group.
- All three values are legal for the third ("membership") flag.
- The fourth flag ("add members") has no sensible interpretation for user and machine entries. It must be set nonetheless, preferably to the hyphen.
- Legal values for the fifth flag ("remove members") are the hyphen and lowercase `r`. This flag has no sensible interpretation for user and machine entries. It must be set nonetheless, preferably to the hyphen.

⇒ To examine the privacy flags on an entry

Step 1: Examine the rights. You may specify an entry either by name or AFS UID.

`%` **pts examine** <*user or group name or id*>

where

e is an acceptable abbreviation of **examine**

user or group name or id
specifies the name (IP address for machines) or AFS UID of each user, machine, or group entry you want to examine.

An example:

```
% pts examine terry:friends
Name: terry:friends, id: -567, owner:terry,  creator: terry,
  membership: 12, flags: SOm--, group quota: 0
```

The relevant field is labeled **flags:**. The `SO` indicates that everyone can examine this entry and find out what other groups it owns, and the `m` that only members of *terry:friends* may find out who else belongs to it. The two hyphens at the end indicate that only *terry* (the owner) and members of *system:administrators* may add and remove members.

⇒ To set the privacy flags on an entry

Step 1: Verify that you have the privilege necessary for setting the privacy flags on any entry (membership in *system:administrators*). If necessary, issue **pts membership** on your own entry; you always have the right to do this. This command is described fully in section 18.4.

% **pts membership** <*user or group name or id*>

Step 2: Set the privacy flags. You may specify the entries to alter either by name (IP address for machines) or AFS UID.

% **pts setfields** <*user or group name or id*> <*set privacy flags*>

where

setf is an acceptable abbreviation for **setfields**.

user or group name or id
specifies the name (IP address for machines) or AFS UID of each user, machine, or group entry for which you want to set the privacy flags.

set privacy flags
specifies the set of privacy flags to associate with this entry. Remember to provide something in all five slots, and obey the constraints detailed in section 18.14.2.

19. Managing Access Control Lists

AFS augments and refines the UNIX file system's mechanisms for controlling access to files and directories. By setting ***access control lists*** at the level of directories rather than individual files, AFS allows the user to grant a more precise set of rights to 20 separate users or groups of users. The groups may include ones that users themselves have defined (chapter 18 explains how to invent groups).

The *AFS User's Guide* includes a chapter corresponding to this one, but this chapter is more comprehensive. This chapter has the following sections:

19.1 Summary of Commands 19-1
19.2 About File Protection in AFS 19-2
19.3 Examining ACLs 19-9
19.4 About Changing ACLs 19-11
19.5 Granting Access 19-13
19.6 Denying Access 19-14
19.7 Restoring Access After Denying It 19-16
19.8 Completely Replacing an ACL 19-18
19.9 Copying ACLs Between Directories 19-19
19.10 Removing Obsolete AFS UIDs from ACLs 19-21
19.11 How the UNIX Mode Bits Work in AFS 19-23

19.1. Summary of Commands

Examine access control list . . . . **fs listacl**
Grant access . . . . **fs setacl**
Deny access . . . . **fs setacl** with **none** shorthand or
fs setacl with **-negative** flag
Restore access . . . . **fs setacl** or
fs setacl with **-negative** flag and **none** shorthand
Replace an ACL . . . . **fs setacl** with **-clear** flag
Copy an ACL . . . . **fs copyacl**
Remove obsolete AFS UIDs . . . . **fs cleanacl**

19.2. About File Protection in AFS

AFS augments and refines the UNIX file system's method for controlling access to files and directories. AFS associates an ***access control list*** (***ACL***) with each directory in the file system as the main means of protection, rather than relying exclusively on mode bits associated with each file (as the UNIX file system does).

The entries on a directory's ACL determine who may access the files within the directory and in what manner they may do so. In other words, the protections defined on a directory's ACL control access to all the files in that directory.

19.2.1. Differences Between UNIX and AFS File Protection

This section elaborates on the differences between the AFS and UNIX protection schemes to demonstrate how AFS provides more-refined protection in several ways.

19.2.1.1. Directory-Level versus File-Level Protection

The UNIX file system associates a set of "mode bits" with each file or directory element. Under this scheme, the bits associated with a directory element control access to the files in the directory; it is also possible to further restrict access to individual files. For instance, if a directory's "read" bits are "off," then none of the files in it are accessible. Even if the directory's "read" bits are "on," it is still possible to protect an individual file more closely by turning off its "read" bits.

ACLs do not make it possible to protect an individual file more restrictively than the other files in a directory. (It is possible to set the mode bits for this purpose, but the setting interacts with the entries on the ACL; see section 19.11.) If a certain file is more sensitive than others, you should store it in a directory with a more restrictive ACL.

19.2.1.2. Number of Access Rights

UNIX defines three access rights—READ, WRITE, and EXECUTE. AFS defines seven access rights—READ, WRITE, LIST, INSERT, DELETE, LOCK and ADMINISTER—to make control more precise. Section 19.2.3 describes AFS's seven rights.

19.2.1.3. Number of Individuals for Whom Access is Defined

The UNIX file system distinguishes among only three user types by providing a set of mode bits for each: the "user" (the owner of the file), a single "group" definable by the user, and "other," which means everyone who has an account on the system.

AFS, in contrast, allows you to place up to 20 different entries on an ACL, granting a different set of access rights to each entry. Each entry may be either a user or a group. The result is that AFS allows you to protect the contents of a directory very precisely.

Users may define their own groups (by default, up to 20) which can be composed on users and client machines. Users can place both their own groups and those created and owned by other users on ACLs. See chapter 18 to learn how to define groups.

When a user appears on an ACL, that user is given all of the rights to that directory as specified on the ACL. When a group appears on an ACL, the members of that group are given the rights specified on the ACL as follows:

- If a user is a member of a group, he or she is given all of the rights of that group specified by the ACL.
- When a user is logged into a machine that is a member of a group, he or she is given all of the rights of that group specified by the ACL.

AFS also allows you to grant access to everyone on the system, if desired, when you place the *system:anyuser* group on an ACL. Or you can use the *system:authuser* group to grant access only to users authenticated in your cell. To learn more about groups on access control lists, see section 19.2.5.

19.2.1.4. How the Mode Bits Are Interpreted

In the UNIX file system, mode bits are the only means of protection. AFS allows you to set the mode bits on a file in addition to the ACL on its parent directory, but it interprets them differently than the UNIX file system does. See section 19.11 to learn more.

19.2.2. The Implications of Directory-Level Protection

Defining access at the directory level has important consequences on your work with files and directories:

- The rights on a directory's ACL apply to all of the files in the directory. If you move a file to a new directory, the access rights that apply to the file change to those of its new directory. To protect the file as in the original directory, make sure that the new directory has an equally restrictive ACL. In addition, changing a directory's ACL changes the protection on all the files in it.
- When you create a subdirectory, it inherits the access rights that its parent directory has at the time. From then on, you may set the subdirectory's ACL independently of its parent's. However, the parent's ACL continues to affect access to the subdirectory in the following way: if the ACL on a parent directory is more restrictive than on a subdirectory, users may be prevented from exercising the rights that the subdirectory's ACL grants them. In particular, the parent's ACL must grant the LOOKUP right to anyone who has rights on the subdirectory. Lack of the LOOKUP right prevents the person from even accessing the subdirectory as an element in the parent directory, let alone accessing its contents.

  In general, you should assign fairly liberal access rights to high-level directories (including user home directories). If certain files require more protection, define more restrictive rights on the ACL of a subdirectory. See section 19.2.5 for more information about granting the LOOKUP right to a wide range of people.

19.2.3. The Access Rights Defined

Seven access rights are available on ACLs. They are associated with a directory and apply to all elements in it. Each right has an accepted single-character abbreviation, which appears after the complete name in the definitions below.

Logically, the access rights are divided into two groups. One group of rights is useful only when applied to the directory itself; the other group of rights is useful only when applied to the files in the directory.

19.2.3.1. The Four Rights that Apply to the Directory Itself

The four rights in this group are useful when applied to the directory itself. For example, the INSERT right does not mean that the possessor can "insert" data into a file (that requires the WRITE right described below), but rather that he or she can create or copy a file into the directory.

The* LOOKUP *right

The LOOKUP (**l**) right allows the possessor to issue

- **ls** to list the names of the files and subdirectories in the directory.
- **ls -ld** to "stat" the directory element itself.
- **fs listacl** with the directory name as the argument to examine the directory's ACL.

With respect to a file in the directory, this right does <u>not</u> allow the possessor to read the contents of the file or to issue **ls -l** or **fs listacl** with the file as an argument. To do any of these things, the possessor must also have the READ right on the ACL.

With respect to a subdirectory in the directory, this right does <u>not</u> allow the possessor to issue **ls**, **ls -l**, **ls -ld**, or **fs listacl** with the subdirectory as an argument. To do any of these things, the possessor must have the LOOKUP right on the subdirectory itself.

This is the least powerful right a user can possess. However, **the user must have it in order to exercise any other rights on the directory.** Lack of the LOOKUP right on a parent directory prevents the user from accessing anything in its subdirectories, even if the user has extensive rights on the subdirectories.

The* INSERT *right

The INSERT (**i**) right allows the possessor 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 DELETE *right*

The DELETE (**d**) right allows the possessor to remove files and subdirectories from the directory or move them into other directories (where the user must have the INSERT right).

The ADMINISTER *right*

The ADMINISTER (**a**) right allows the possessor to change the ACL of the directory. Members of *system:administrators* have this right on every directory, even when they do not appear on the ACL. A user always has this right on their home directory and sub-directory descendants of the home directory, even if they inadvertently remove themselves from the ACL.

19.2.3.2. The Three Rights that Apply to Files in the Directory

The three rights in this group are useful when applied to files in the directory, not to the directory itself or to its subdirectories.

The READ *right*

The READ (**r**) right allows the possessor to read the contents of files in the directory. It also allows the user to issue **ls -l** to "stat" the file elements.

The WRITE *right*

The WRITE (**w**) right allows the possessor to modify the contents of files in the directory and to change their UNIX mode bits with **chmod**.

The LOCK *right*

The LOCK (**k**) right allows the possessor to run programs that need to "flock" files in the directory.

19.2.3.3. Shorthand Forms for Common Combinations of Rights

You can combine the seven rights 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 type either the individual letters corresponding to the rights or the shorthand forms, which are

write all rights except ADMINISTER (that is, **rlidwk**).

read READ and LOOKUP rights (**rl**).

all all seven rights (**rlidwka**).

none no rights. This removes the user's entry from the ACL entirely.

19.2.3.4. Undefined Access Rights

Eight additional, undefined access rights are also available for use on an ACL. The additional rights are denoted by the uppercase letters *A* through *H* (`A`, `B`, `C`, `D`, `E`, `F`, `G`, and `H`). The letters must be uppercase if used.

By default, the additional rights have no meaning to AFS server processes (they are undefined). However, administrators and application programs can assign meanings to them and place them on ACLs to control access to directories' contents in new ways. For example, they can be defined to control access to printer daemons (which must then be programmed to recognize and interpret the rights). The additional rights are recognized by the **fs listacl** and **fs setacl** commands.

19.2.4. Normal versus Negative Rights

AFS allows you not only to grant access, but also to deny it. The distinction between granting and denying corresponds to an ACL's two subparts, the `Normal rights` list and the `Negative rights` list.

- To grant rights to a particular user/group, you use the **fs setacl** command to create an entry on the `Normal rights` part of the ACL. See section 19.5 to learn how to grant access.
- To deny rights to a certain user/group, you create an entry on the `Negative rights` part of the ACL by adding the **-negative** flag to the **fs setacl** command. The `Negative rights` list is interpreted after the `Normal rights` list, so it cancels out corresponding rights the user/group may have gained from the `Normal rights` list. See section 19.6 to learn how to deny access and section 19.7 to learn how to restore access previously denied.

Warning: AFS provides negative rights for maximum flexibility, but setting them is usually unnecessary and is not recommended. Simply omitting someone from the `Normal rights` list is generally enough to prevent undesired access, <u>provided the ACL does not grant the rights to a group to which the user belongs</u>.

If you use negative rights, keep the following in mind:

- You are reversing the meaning of the rights by putting them on the `Negative rights` list. In particular, the **none** shorthand constitutes a "double negative." It removes the entry from the `Negative rights` list, making it possible for the person to gain access via the `Normal rights` list. The **all** grouping means the person or group is explicitly forbidden all rights.
- It is useless to deny rights if the `Normal rights` part of the same ACL grants those rights to *system:anyuser*. As explained more fully in section 19.2.5, *system:anyuser* includes anyone logged into a machine in the cell. If the denied person uses the **unlog** command, they replace their "true" identity with *anonymous*. The File Server considers *anonymous* a member of *system:anyuser* and so grants the person *system:anyuser*'s rights. It does not recognize the person as someone on the `Negative Rights` list, and so cannot deny them the specified rights.

19.2.5. Groups on ACLs Can Extend Access to Many People

You can place a group on an ACL in the same way that a user is placed on an ACL; however, a group has the potential to significantly extend access depending upon the membership of the group:

- Every user member of a group is granted or denied the rights specified for the group on the ACL.
- Every user logged into a machine that is a member of a group is granted or denied the rights specified for the group on the ACL. (Unlike users, machine entries cannot be placed directly on an ACL; instead, the machine IP addresses must be defined as members of a group, and then the group can be put on the ACL.)

Note: Although it is technically possible to mix machine entries with user entries in the same group, it is not recommended. In general, it is easier to use and administer non-mixed groups.

By default, each user may create up to 20 groups; of course, system administrators may increase or decrease this quota. System administrators can create an unlimited number of groups. The Protection Database keeps track of group memberships.

Users can put any groups on an ACL, not just those they themselves create or own. Chapter 18 discusses the possible types of group ownership and the most effective ways to use different group types on ACLs.

19.2.5.1. Two system-defined groups

In addition to putting user defined groups on ACLs, it is also possible to place two system-defined groups on ACLs to grant access to a very wide range of people at once. The groups are ***system:anyuser*** and ***system:authuser***, which are defined fully in section 18.2.5.

These two groups are unusual in that they have no stable membership. Instead, the membership of *system:anyuser* changes as people attempt to access files in the system, and the membership of *system:authuser* changes as people authenticate and unauthenticate.

- Anyone who can access the cell's file tree is considered a member of *system:anyuser*. This includes people who have logged in as "root" on a certain machine, have telnetted in from outside the cell, or are using AFS in a foreign cell. Thus, this group includes people who have not authenticated and have the AFS UID for *anonymous*.
- Only people who are currently authenticated (have a valid token) in the cell belong to *system:authuser*. Such people have proved their identity to the Authentication Server by executing the `klog` command and providing a password, either at login or after.

Including these two groups on ACLs allows you to grant or deny access to a wide array of people at one time, as discussed in the next section.

19.2.5.2. How to Use the System-Defined Groups on ACLs

The system-defined groups are a useful way to provide the most basic access to a wide range of people. In particular, it is useful to grant the LOOKUP right to *system:anyuser* on the ACL of most directories in the file system. Possessing the LOOKUP right allows a user to do no more than learn the names of files and subdirectories in a directory. On the other hand, lack of this right prevents the user (and the user's Cache Manager) from accessing any of those elements on its way to a pathname to the desired file. Lack of the LOOKUP right thus prevents the user from exercising any rights on a subdirectory, no matter how extensive.

A slightly more restrictive alternative to granting LOOKUP to *system:anyuser* is to grant it to *system:authuser*. If you feel this is not restrictive enough, you can grant LOOKUP only to individual users or groups. However, only 20 entries may appear on an ACL.

Another reason to grant *system:anyuser* certain rights on directories is that process daemons that provide services such as printing and mail delivery often require certain rights in order to work correctly. For example, in addition to LOOKUP, a printing daemon may need READ rights on certain directories to print out the contents of files, and a mail daemon may require INSERT rights to deliver new pieces of mail. As a system administrator, you will probably grant rights like these as you create directories, particularly user home directories.

Note: The only way to allow users from foreign cells (who do not have accounts in the local cell) to access a directory is to grant *system:anyuser* rights on that directory's ACL. There is currently no way to grant access to specific users from foreign cells, except by giving them accounts in the local cell.

To ease administration, AFS grants ALL rights to *system:administrators* to each root directory's of each volume. You can remove these rights; however, a member of *system:administrators* always has implicit `administer` rights on every ACL. So a member of *system:administrators* can always grant himself or herself any rights on a directory.

19.3. Examining ACLs

Before changing an ACL, you may wish to check its current state. The **fs listacl** command outputs the ACL for a directory. It requires that the issuer already have the LOOKUP right on the directory.

It is possible to issue **fs listacl** with a filename argument if you have READ and LOOKUP rights on the directory. The list shown is associated with the file's parent directory.

⇒ To list a directory's ACL

Step 1: Issue the **fs listacl** command.

% **fs listacl [**<*dir/file path*>**]**

where

la is an acceptable alias for *listacl*.

dir/file path names the directory whose ACL you want to see. If you specify a filename, the ACL of the file's parent directory is displayed. If you omit this argument, the ACL of the current directory is displayed.

If you receive the error message

`fs: You don't have the required access rights on '`*dir/file path*`'`

and *dir/file path* is a directory name, this indicates you do not possess the LOOKUP right on the directory's ACL. If *dir/file path* is a file name, you do not possess the LOOKUP and READ rights on the ACL associated with the file's parent directory. If you are a member of *system:administrators*, you may grant yourself that right, as you have implicit ADMINISTER rights on every directory. See section 19.5 to learn how to grant access rights using **fs setacl**.

The output of the **fs la** command pairs user or group names with a list of access rights. The rights appear as the single letters defined in 19.2.3, and always in the order `rlidwka`. The outputs lists normal rights and negative rights separately; if no negative rights are defined, that part of the output does not appear. The following example shows the home directory of the fictional Transarc Corporation employee *terry*:

% **fs la /afs/transarc.com/usr/terry**

```
Access list for /afs/transarc.com/usr/terry is
Normal rights:
   system:authuser rl
   pat rlw
   terry rlidwka
Negative rights:
   terry:other-dept rl
   jones rl
```

where *pat*, *terry*, and *jones* are individual users; *system:authuser* is a system-defined group; and *terry:other-dept* is a group owned by *terry*. The list under `Normal rights` indicates that *terry* has all seven access rights; *pat* has READ, LOOKUP, and WRITE rights; and anyone belonging to the *system:authuser* group has READ and LOOKUP rights.

The list under `Negative rights` denies READ and LOOKUP rights to *jones* and the members of the group *terry:other-dept*. The end result is that *jones* and members of *terry:other-dept* cannot access this directory in any way, because they do not appear at all on the `Normal rights` list, and the `Negative rights` list explicitly denies them the READ and LOOKUP rights they might gain by belonging to *system:authuser*.

See section 19.6 to learn more about the interaction between normal and negative rights. See chapter 18 to learn more about groups.

Note: The output from the following command shows the ACL of every subdirectory and file in the current directory. Although the output makes it appear that files have their own ACLs, ACLs do not protect files individually. The ACL shown for a file is the same as the parent directory's and cannot be changed independently.

% **fs listacl ***

19.4. About Changing ACLs

The **fs setacl** command allows you to change ACLs in several different ways in order to

- Grant access (see section 19.5).
- Deny access (see section 19.6).
- Restore access after denying it (see section 19.7).
- Erase an ACL, replacing it with a completely new one (see section 19.8).

You must have the ADMINISTER right on an ACL to change it. Members of *system:administrators* have implicit ADMINISTER right on the ACL of every directory. The owner of a directory also has the implicit ADMINISTER right.

For your convenience, the complete syntax of the **fs setacl** command appears here. The following sections define how to use the command for specific purposes.

fs setacl -dir <*directory*>+ **-acl** <*access list entries*>+
[-clear] [-negative] [-help]

where

sa is an acceptable alias for **setacl**.

directory names each directory you wish to change. Unlike the **fs listacl** command, you must indicate a directory name even when changing only the working directory's ACL. See the **Hint** below.

access list entries
specifies one or more pairings between a user/group name and a set of access rights, in that order. You may mix user and group names in one command. Use spaces to separate a name from its associated rights and the pairs of names and rights from one another. See the **Hint** below.

Each set of rights is either some combination of the seven letters `rlidwka`, in any order, or one of the four shorthand groupings of rights (**read**, **write**, **all**, or **none**); full definitions of these appear in section 19.2.3. It is legal to put both letters and groupings in the same command, but not within a single pair.

-clear indicates that the ACL should be erased and completely replaced with the indicated *access list entries*.

-negative indicates that all the *pair*s should go on the `Negative rights` list.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one.

Note: If an ACL already grants certain rights to a user or group, and you want to specify a new right for that user or group, the rights you specify replace the existing rights; they are not added to them. In other words, if you want to extend additional rights to user or group that already has rights on an ACL, include all of the rights, new and old, on the set of rights that you specify.

Hint: This command is easiest to use if you specify only one directory, in which case you may omit the **-dir** and **-acl** switches. You may provide as many *access list entries* as you wish even when you do not provide the switches, as long as they appear after the *directory* name.

To set the ACL for all of the subdirectories in a parent directory at once, issue **fs setacl** in the parent directory, specifying the asterisk (`*`) for *directory*. The parent directory's ACL is unaffected.

19.5. Granting Access

To grant access to a directory, you set rights on the `Normal rights` part of the ACL. The absence of the **-negative** or **-clear** flag on the command line indicates that you want to grant rights, not take them away.

⇒ To grant access on an ACL

Step 1: Verify that you have ADMINISTER rights on each directory for which you wish to grant rights. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Grant rights on the ACL.

% **fs setacl -dir** <*directory*>$^+$ **-acl** <*access list entries*>$^+$

For example, to give *pat* READ and LOOKUP access to your **notes** subdirectory, you could type

% **fs sa ~/notes pat rl**

which adds the following entry to the ACL of the **notes** subdirectory:

```
pat rl
```

However, because **read** is the shorthand grouping for READ and LOOKUP, you could also type

% **fs sa ~/notes pat read**

to add the same entry to the ACL. Both entries grant READ and LOOKUP access to *pat* on the **notes** subdirectory.

19.6. Denying Access

There are two ways to deny someone access to a directory:

- Remove the user or group from the `Normal rights` list by placing **none** after the user name in the plain **fs setacl** command. This works because those who don't appear on the `Normal rights` list have no access, unless they belong to a group that does appear there. Be mindful of putting *system:anyuser* on ACLs, because everyone on the system receives the rights granted to it on the ACL.
- Put the user or group on the `Negative rights` part of the ACL by adding the **-negative** flag to **fs setacl**. Since people can gain access by virtue of group membership, assigning negative rights is the only way to be absolutely sure that someone cannot access a directory. The File Server computes negative rights after normal rights, so an entry on the `Negative rights` list overrides any rights someone may gain by belonging to a group that appears under `Normal rights`.

  Note: You must make sure that *system:anyuser* does not have the corresponding rights on the `Normal rights` list. Someone on the `Negative rights` list only has to **unlog** to gain the rights granted to *system:anyuser*.

⇒ To remove an entry from the `Normal rights` list

Step 1: Verify that you have the ADMINISTER right on each directory on which you wish to deny rights. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Deny the specified user access by removing them from the `Normal rights` list.

% **fs setacl -dir** <*directory*>$^+$ **-acl** <*name*> **none**

where

name **none** indicates that you wish to remove the user or group *name* from the `Normal rights` list. You may specify **none** for more than one user/group at a time. In addition, you may grant rights to other users/groups in the same command.

⇒ To put an entry on the `Negative rights` list

Step 1: Verify that you have ADMINISTER right on each directory for which you wish to deny rights. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Add an entry for the specified user to the `Negative rights` list.

% fs setacl -dir <*directory*>$^+$ **-acl** <*access list entries*>$^+$ **-negative**

where

-negative indicates that the specified *access list entries* are to go on the `Negative rights` part of the ACL of each *directory*.

Note: If a user/group already appears on the `Negative rights` list, the rights you specify when issuing **fs setacl** with the **-negative** flag replace the existing rights; they are not added to them. In other words, you should specify the complete list of rights you want to deny a user who is already on the `Negative rights` list.

19.7. Restoring Access After Denying It

It is possible to restore rights that you have taken away using the methods described in section 19.6. The procedure for restoring rights depends on how you denied access in the first place:

- If you denied access simply by using the **none** shorthand to remove the user/group from the `Normal rights` list, no entry for the user/group appears in the output from **fs listacl**. To restore access, use the regular **fs setacl** command described in section 19.5 to give the user or group the rights you wish them to have.
- If you put the user/group on the `Negative rights` list, as confirmed by the **fs listacl** command, to restore access you must remove the user/group from the `Negative rights` list before you can give them any rights on the `Normal rights` list. If you forget to remove their negative rights, they will still be locked out, because negative rights override normal rights. You combine the **-negative** flag with the **none** shorthand grouping to remove someone's negative rights.

⇒ To restore access after using the none shorthand

Step 1: Verify that you have ADMINISTER right on each directory for which you wish to grant rights. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Use the regular **fs setacl** command to grant rights.

% **fs setacl -dir** <*directory*>+ **-acl** <*access list entries*>+

⇒ To restore access after putting an entry on the `Negative rights` list

Step 1: Verify that you have ADMINISTER right on each directory for which you wish to change the ACL. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Remove the entry from the `Negative rights` list.

% fs setacl -dir <*directory*>+ **-acl** <*name*> **none -negative**

where

-negative indicates that the specified entry is to be removed from the `Negative rights` part of the ACL of each *directory*.

name **none** indicates that you wish to remove the user or group *name* from the `Negative rights` list. You may specify **none** for more than one user/group at a time. You may also deny rights to other users/groups in the same command (that is, all entries provided in the command go on the `Negative rights` list).

Step 3: Grant the desired rights on the `Normal rights` part of the ACL, using the regular **fs setacl** command.

% fs setacl -dir <*directory*>+ **-acl** <*access list entries*>+

19.8. Completely Replacing an ACL

It is sometimes desirable to clear an ACL completely before defining new rights on it. For instance, there may be such a mix of normal and negative rights on an ACL that you are no longer sure about the effects of their interaction. The **-clear** flag on the **fs setacl** command erases both the `Normal rights` and `Negative rights` parts of an ACL. The rights you specify on the same command line then go onto the newly cleared list. It is possible to define only one type of rights (normal or negative) with this single command.

Note: Make sure to grant the owner of the directory all seven rights when you issue this command. Without the LOOKUP right in particular, the owner will be unable to do anything in that directory that requires interpreting the "`.`" or "`..`" notation, including commands such as **fs la .** or **cd ..**

Members of *system:administrators* implicitly have the ADMINISTER right on every ACL, so they can help users who may have locked themselves out of a directory. Also, users always retain the ADMINISTER rights on their home directories, so they can always **cd** to their home directories and grant themselves necessary rights in any subdirectories.

⇒ To replace an ACL completely

Step 1: Verify that you have ADMINISTER right on each ACL you wish to replace. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Clear the ACL and set either normal or negative rights. Since you need to grant the owner of the directory all rights, it is better in most cases to set normal rights at this point.

% **fs setacl -dir** <*directory*>+ **-acl** <*access list entries*>+
-clear [-negative]

where

-clear indicates that you want to clear the ACL before placing the indicated *access list entries* on it.

-negative indicates that the specified *access list entries* are to go on the `Negative rights` part of the ACL. Omit this flag to place the entries on the `Normal rights` part.

Step 3: Use the **fs setacl** command to set additional rights, if you wish. If you set normal rights in step 2 (as recommended), you may use the command to set negative rights (using the **-negative** flag) or additional normal rights.

19.9. Copying ACLs Between Directories

The **fs copyacl** command allows you to copy entries from the ACL of one directory to the ACLs of one or more other directories. The command requires you to specify the name of the *source directory* from which the ACL is to be copied and the name of each *destination directory* to which the *source directory*'s ACL is to be copied. You must have the LOOKUP right on the *source directory* and the ADMINISTER right on each *destination directory*. You can specify a filename for *source directory*, in which case the ACL of the file's parent directory, on which you must have both READ and LOOKUP rights, is used.

The command never affects entries on the ACL of the *source directory*. By default, it also does not affect entries on the ACL of a *destination directory* that are not present on the ACL of the *source directory*. However, it comes with an optional **-clear** flag that allows you to completely replace the ACL of each *destination directory* with the ACL of the *source directory*.

The command affects entries on the ACL of each *destination directory* as follows:

- If an entry is on the ACL of the *source directory* but not on the ACL of the *destination directory*, the command copies it to the ACL of the *destination directory*.
- If an entry is on the ACLs of both the *source directory* and the *destination directory*, the command changes it on the ACL of the *destination directory* to match the rights granted on the ACL of the *source directory*.
- If an entry is on the ACL of the *destination directory* but not on the ACL of the *source directory*, the command does not affect the entry (unless the **-clear** flag is being used).

⇒ To copy an ACL between directories

Step 1: Verify that you have LOOKUP right on the *source directory* from which the ACL is to be copied and ADMINISTER right on each *destination directory* to which the ACL is to be copied. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Issue the **fs copyacl** command to copy the ACL from the *source directory* to each *destination directory*.

% **fs copyacl -fromdir** <*source directory*> **-todir** < *destination directory*>+
[-clear]

where

co is an acceptable abbreviation for **copyacl**.

source directory
names the *source directory* whose ACL is to be copied to each *destination directory*. You can specify a directory or a filename. If you specify a filename, the ACL of the file's parent directory is used.

destination directory
names each *destination directory* to receive the ACL of the *source directory*. You cannot provide a filename for this argument.

-clear indicates that you want to completely replace the ACL of each *destination directory* with the ACL of the *source directory*. All entries are removed from the ACL of each *destination directory* before the ACL of the *source directory* is copied.

For example, suppose you want the ACL of the subdirectory *plans* to contain all of the entries on the subdirectory *notes*, and you want those entries to grant the same rights. However, you want *plans* to retain any additional entries it does not share with *notes*. The ACLs of the two directories are currently

```
% fs la notes plans
Access list for notes is
Normal rights:
   terry rlidwka
   smith rl
   jones rl

Access list for plans is
Normal rights:
   terry rlidwk
   pat rlidwk
```

The **fs copyacl** command can be used to copy the ACL from *notes* to *plans*. Because the **-clear** flag is omitted from the command, the entry for *pat* remains on the ACL of *plans*.

```
% fs co notes plans

% fs la notes plans
Access list for notes is
Normal rights:
   terry rlidwka
   smith rl
   jones rl

Access list for plans is
Normal rights:
   terry rlidwka
   pat rlidwk
   smith rl
   jones rl
```

19.10. Removing Obsolete AFS UIDs from ACLs

When you remove a user or group entry from the Protection Database, the AFS UID of the former user or group appears on every ACL on which the name once appeared. Consider the following example where a fictional Transarc Corporation employee *terry* has granted the group *terry:friends* (AFS UID -567) extra rights on his home directory, and then has removed the group from the Protection Database:

```
% fs listacl /afs/transarc.com/usr/terry
Access list for /afs/transarc.com/usr/terry is
Normal rights:
  terry:friends rlik
  system:anyuser l
  terry rlidwka

% pts delete terry:friends

% fs listacl /afs/transarc.com/usr/terry
Access list for /afs/transarc.com/usr/terry is
Normal rights:
  -567 rlik
  system:anyuser l
  terry rlidwka
```

There are two drawbacks to having AFS UIDs appear on ACLs:

- It clutters ACLs with irrelevant information, making them harder to read.
- If the system administrator ever reuses an old AFS UID, the new possessor of the ID may end up gaining access that the owner of the directory actually intended only for the old possessor of the ID (section 18.2.6 advises against reusing AFS UIDs for this reason).

Periodically, you may wish to remove AFS UIDs from ACLs using the **fs cleanacl** command.

⇒ To clean obsolete AFS UIDs from an ACL

Step 1: Verify that you have ADMINISTER right on each directory for which you wish to clean the ACL. If necessary, issue the **fs listacl** command, described fully in section 19.3.

% **fs listacl [**<*dir/file path*>**]**

If you are a member of *system:administrators*, you have the ADMINISTER right on every ACL and can use **fs setacl** to grant yourself other rights as necessary.

Step 2: Clean the ACL of obsolete AFS UIDs.

% fs cleanacl [<*dir/file path***>]**

where

cl is an acceptable abbreviation for **cleanacl**.

dir/file path names each directory whose ACL you wish to clean. If you specify a filename, the ACL of the file's parent directory is cleaned. Omit this argument to clean the ACL of the working directory.

You can use the asterisk for *dir/file path* to clean the working directory and all it subdirectories simultaneously.

19.11. How the UNIX Mode Bits Work in AFS

The introduction to this chapter explained that AFS does not base protection solely on the nine mode bits associated with each file, as the UNIX file system does. However, AFS does not ignore the mode bits entirely. Specifically, AFS pays attention to the bits as follows:

- It still pays attention to the "type indicator" that appears first in the output of **ls -l** (for example, `-` for a file, `d` for a directory).
- AFS looks at the first set of `rwx` bits—those associated with "user" (the owner of the file). AFS ignores the mode bits for "group" and "other," although all of them still change when you issue the appropriate **chmod** command. All nine mode bits still apply to any files kept on the local disk of a workstation.
- The "user" bits on a directory are ignored.
- The "user" bits on a file interact with ACL rights in the following ways:
  - If the `r` mode bit is present ("turned on"), anyone with the READ and LOOKUP rights on the ACL of the file's parent directory is allowed to read the file. If the `r` bit is absent, no one (including the owner) can read the file, no matter what permissions they have on the ACL.
  - If the `w` mode bit is present, anyone with the WRITE and LOOKUP rights on the ACL of the file's parent directory can modify the file; if the `w` bit is off, no one can modify the file, not even its owner.
  - Although there is no ACL right directly corresponding to `x`, if the `x` bit is on, anyone with READ and LOOKUP rights on the ACL of the file's parent directory can read the file to execute it.

For files in AFS, you may turn the `r`, `w` and `x` bits on and off with the UNIX **chmod** command. To change a file's mode bits, the required permission is the WRITE right on the ACL of the directory that houses the file. To change a directory element's mode bits, the required ACL rights are INSERT, LOOKUP, and DELETE.

20. Managing Privileged Users

This chapter explains how to grant privileged status to system administrators and operators so that they will be able to perform system operation tasks. It has the following sections:

20.1 Summary of Commands 20-1
20.2 The Different Types of Privilege 20-2
20.3 Granting Privilege for **pts** and **fs**: the *system:administrators* Group 20-4
20.4 Granting Privilege for **kas**: the ADMIN Flag 20-5
20.5 Granting Privilege for **bos**, **vos**, and **backup**: the *UserList* 20-7
20.6 Granting All System Administrator Privileges 20-10

20.1. Summary of Commands

List members of *system:administrators*	**pts membership**
Add a user to *system:administrators*	**pts adduser**
Remove a user from *system:administrators*	**pts removeuser**
Check for `ADMIN` flag in Authentication Database entry	**kas examine**
Add the `ADMIN` flag to an Authentication Database entry	**kas setfields**
Remove the `ADMIN` flag from an Authentication Database entry	**kas setfields**
List users in *UserList*	**bos listusers**
Add a user to *UserList*	**bos adduser**
Remove a user from *UserList*	**bos removeuser**

20.2. The Different Types of Privilege

An AFS system administrator is privileged in three different ways. He or she

- Belongs to the *system:administrators* group.
- Has the `ADMIN` flag in his or her Authentication Database entry.
- Is included in the file */usr/afs/etc/UserList* on the local disk of each file server and database server machine.

This section describes these three privileges and explains why more than one privilege is necessary.

20.2.1. Membership in *system:administrators*

The first type of privilege is membership in the *system:administrators* group in the Protection Database. The resulting privileges are:

- The right to issue all **pts** commands. These allow the administrator to create and administer all Protection Database entries, including user entries and group entries owned by others. The administrator can also set the counters that control the Protection Server's allocation of AFS UIDs.
- Implicit ADMINISTER right on the access control list of every directory in the file system. This means that members of *system:administrators* may always grant themselves the rights necessary to perform other actions, with the **fs setacl** command.
- The right to issue the **fs setvol** and **fs setquota** commands, which sets the space quota on volumes.

To learn how to grant this privilege, see section 20.3.

20.2.2. The ADMIN Flag in the Authentication Database

The second kind of privilege is having the `ADMIN` flag in one's Authentication Database entry. With this privilege, the user has the right to issue all **kas** commands. These allow the administrator to manage Authentication Database entries.

To learn how to grant this type of privilege, see section 20.4.

20.2.3. Inclusion in */usr/afs/etc/UserList*

The third type of privilege is inclusion in the file */usr/afs/etc/UserList* on the local disk of each file server machine and each database server machine. The resulting privileges include

- The right to issue all **bos** commands. These allow the administrator to control the status of server processes, install new binaries, and manage the global configuration files that define the cell's database server machines, server encryption keys and privileged users.

- The right to issue all **vos** commands. These allow the administrator to manage volumes and make changes to the VLDB.
- The right to issue all **backup** commands. These allow the administrator to use the AFS Backup System to make changes to the Backup Database and to back up data to tape.

To learn how to grant this type of privilege, see section 20.5.

20.2.4. The Reason for Separate Privileges

Using three separate types of privilege allows AFS sites to grant different privileges to different personnel, which means that different levels of system administrators can be empowered for different tasks. You will probably want to grant all three types to at least some of your cell's administrators; but you may also find that some system administrators require only some of the privileges described in this section. Section 20.6 summarizes the procedures for granting or revoking all three types of privilege.

The *system:administrators* 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 from membership in *system:administrators* because of the extreme sensitivity of the information in the Authentication Database, especially the server encryption key(s). Separating it allows you to grant this privilege to only a subset of system administrators, if desired. 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 privilege determined by */usr/afs/etc/UserList* is separate from the others; the record of it is stored on the local disk of each file server and database server machine, rather than in a database. In case of serious server or network problems, its presence on disk makes it possible for administrators to log in at the console of a server machine and still issue the privileged **bos** and **vos** commands, which can help solve the problem.

20.2.5. A Warning about *anonymous*

The AFS server processes recognize any unauthenticated user as *anonymous*. Granting *anonymous* any sort of privilege allows everyone who can reach your cell to issue privileged commands.

20.3. Granting Privilege for pts and fs: the *system:administrators* Group

By making someone a member of the *system:administrators* group in the Protection Database, you grant them the right to issue all **pts** commands, to set volume quotas with the **fs setvol** and **fs setquota** commands, and implicitly give them the ADMINISTER right on the access control list of every directory in the file system.

⇒ To list the members of *system:administrators*

Step 1: List the members of *system:administrators*.

% **pts membership system:administrators**

where **m** is an acceptable abbreviation of **membership**.

⇒ To add users to *system:administrators*

Step 1: Verify that you have the privilege necessary to add users to *system:administrators* (you belong to this group yourself). If necessary, issue the **pts membership** command.

% **pts membership system:administrators**

Step 2: Add each user to *system:administrators*.

% **pts adduser -user** <*user name*>$^+$ **-group system:administrators**

where

ad	is an acceptable abbreviation of **adduser**.
user name	names each user you want to add to *system:administrators*.

⇒ To remove users from *system:administrators*

Step 1: Verify that you have the privilege necessary to remove users from *system:administrators* (you belong to this group). If necessary, issue the **pts membership** command.

% **pts membership system:administrators**

Step 2: Remove each user from *system:administrators*.

% **pts removeuser -user** <*user name*>$^+$ **-group system:administrators**

where

rem	is an acceptable abbreviation of **removeuser**.
user name	names each user you want to remove from *system:administrators*. *user name*.

20.4. Granting Privilege for kas: the ADMIN Flag

You grant or deny an administrator the right to issue all **kas** commands by adding or removing the `ADMIN` flag in his or her Authentication Database entry.

⇒ To see if the `ADMIN` flag is turned on

Step 1: Verify that you have the privilege necessary to examine Authentication Database entries (that the `ADMIN` flag is set on your Authentication Database entry). If necessary, issue the **kas examine** command on your own entry; every user is allowed to do this. The **kas examine** command is fully described in the next step.

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

Step 2: Examine the Authentication Database entry, looking for the `ADMIN` flag on the first line. You will be prompted for your password.

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

where

e is an acceptable abbreviation of **examine**.

name of user is the entry you wish to look at.

If the `ADMIN` flag is in the entry, it appears on the first line of the output. An example:

```
% kas ex terry
User data for terry (ADMIN)
  key version is 0, etc...
```

⇒ To set the `ADMIN` flag

Step 1: Verify that you have the privilege necessary to add the `ADMIN` flag (that the `ADMIN` flag is set on your Authentication Database entry). If necessary, issue the **kas examine** command. You will be prompted for your password.

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

Step 2: Add the `ADMIN` flag to the Database entry. You will be prompted for your password.

% **kas setfields** <*name of user*> **ADMIN**
`Password for` *your user name*`:` <*your password*>

where

sf is an acceptable alias for **setfields**.

name of user is the entry in which you wish to set the `ADMIN` flag.

⇒ To remove the `ADMIN` flag

Step 1: Verify that you have the privilege necessary to remove the `ADMIN` flag (it appears in your own Authentication Database entry). If necessary, issue the **kas examine** command described above. You will be prompted for your password.

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

Step 2: Remove the `ADMIN` flag from the Database entry. You will be prompted for your password.

% **kas setfields** <*name of user*> **NOADMIN**
`Password for` *your user name*`:` <*your password*>

where

sf is an acceptable alias for **setfields**.

name of user is the entry from which you want to remove the `ADMIN` flag.

20.5. Granting Privilege for bos, vos, and backup: the *UserList*

You grant someone the right to issue all privileged **bos**, **vos**, and **backup** commands by adding that person's user name to the file */usr/afs/etc/UserList*. Similarly, you deny that person the right to issue such commands by removing the user name from the file. Each file server machine and each database server machine stores a copy of the file on its local disk.

To guarantee that all file server and database server machines consider the same users privileged, all server machines use the same *UserList* file. To keep the *UserList* files the same on all server machines, never edit a *UserList* file directly. Instead, use the **bos** commands described in this section to make the desired changes.

For administrators in cells using the United States edition of AFS, the easiest way to update the *UserList* is to direct the appropriate **bos** command to your cell's system control machine. If you have forgotten which machine is the system control machine, consult section 4.4. Make sure all other file server and database server machines are running the client portion of the Update Server to reference the system control machine.

Administrators in cells using the international edition of AFS must direct the relevant **bos** command to each file server and database server machine in turn because international cells should not use the Update Server to distribute the contents of */usr/afs/etc*.

⇒ To list the users in *UserList*

Step 1: List the contents of */usr/afs/etc/UserList*.

% **bos listusers** <*machine name*>

where

listu is an acceptable abbreviation of **listusers**.

machine name may be any of the cell's file server or database server machines, since they all should share the same copy of *UserList*. You can check the *UserList* on a certain machine if you suspect it is not the same as the others.

⇒ To add users to *UserList*

Step 1: Verify you have the privilege necessary to add people to *UserList* (you appear in it yourself). If necessary, issue the **bos listusers** command described fully above.

% **bos listusers** <*machine name*>

Step 2: If using the United States edition of AFS, add each user to the copy of *UserList* on the system control machine. If using the international edition of AFS, repeat the command once for each file server and database server machine, substituting its name for *machine name*.

% **bos adduser** <*machine name*> <*user names*>$^+$

where

addu is an acceptable abbreviation of **adduser**.

machine name should name the system control machine in cells using the United States edition of AFS. If *machine name* is not the system control machine but is running a client portion of the Update Server, the changes you make are overwritten the next time the client portion copies the system control machine's copy of *UserList* (by default, within 5 minutes).

In cells using the international edition of AFS, *machine name* should name each file server or database server machine in turn.

user names includes the user name of each administrator you wish to add to *UserList*.

Note: If you are using the United States edition of AFS and have changed *UserList* on the system control machine only, it will take a few minutes for the Update Server to distribute the new *UserList* to all file server and database server machines (by default, up to 5 minutes). The newly added users need to wait that amount of time before issuing commands that require this type of privilege.

⇒ To remove users from *UserList*

Step 1: Verify you have the privilege necessary to remove people from *UserList* (you appear in it yourself). If necessary, issue the **bos listusers** command described fully above.

% **bos listusers** <*machine name*>

Step 2: If using the United States edition of AFS, remove each user from the copy of *UserList* on the system control machine. If using the international edition of AFS, repeat the command once for each file server and database server machine, substituting its name for *machine name*.

% **bos removeuser** <*machine name*> <*user names*>$^+$

where

removeu is an acceptable abbreviation of **removeuser**.

machine name should name the system control machine in cells using the United States edition of AFS. If *machine name* is not the system control machine but is running a client portion of the Update Server, the changes you make are overwritten the next time the client portion copies the system control machine's copy of *UserList* (by default, within 5 minutes).

In cells using the international edition of AFS, *machine name* should name each file server or database server machine in turn.

user names includes the user name of each administrator you wish to remove from *UserList*.

Note: If you are using the United States edition of AFS and have changed *UserList* on the system control machine only, it will take a few minutes for the Update Server to distribute the changed file to all file server and database server machines (by default, up to 5 minutes). The newly removed users will be able to issue commands that require this type of privilege until that amount of time passes.

20.6. Granting All System Administrator Privileges

Granting someone all three types of privilege requires issuing three separate commands, as does revoking all privileges. The previous three sections described the commands fully. This section summarizes the procedures for quick reference.

Because some of the commands allow you to grant privileges to only one user per issuance, the instructions assume you will specify only one user name. They do not show the switches you would have to include if granting privileges to more than one person per issuance of the command.

⇒ To grant complete system administrator privileges

Step 1: Verify that you have the three types of privileges necessary to grant all privileges. If necessary, issue **pts membership** (described fully in section 20.3, **kas examine** (section 20.4) and **bos listusers** (section 20.5).

`%` **pts membership system:administrators**

`%` **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

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

Step 2: Add the user to *system:administrators*.

`%` **pts adduser** <*user name*> **system:administrators**

where

ad is an acceptable abbreviation of **adduser**.

user name is the user you want to add to *system:administrators*.

Step 3: Add the `ADMIN` flag to the Authentication Database entry. You will be prompted for your password.

`%` **kas setfields** <*name of user*> **ADMIN**
`Password for` *your user name*`:` <*your password*>

where

sf is an acceptable alias for **setfields**.

name of user is the entry to which you want to add the `ADMIN` flag.

Step 4: Add the user to *UserList*.

% **bos adduser** <*machine name*> <*user name*>

where

addu is an acceptable abbreviation of **adduser**.

machine name should name the system control machine in cells using the United States edition of AFS. If *machine name* is not the system control machine but is running a client portion of the Update Server, the changes you make are overwritten the next time the client portion copies the system control machine's copy of *UserList* (by default, within 5 minutes).

In cells using the international edition of AFS, *machine name* should name each file server or database server machine in turn.

user names is the user name you wish to add to *UserList*.

Note: If you are using the United States edition of AFS and have changed *UserList* on the system control machine only, it will take a few minutes for the Update Server to distribute the changed file to all file server and database server machines (by default, up to 5 minutes). The newly added users need to wait that amount of time before issuing commands that require this type of privilege.

⇒ To revoke complete system administrator privileges

Step 1: Verify that you have the three types of privilege necessary to revoke all privileges. If necessary, issue **pts membership** (described fully in section 20.3, **kas examine** (section 20.4) and **bos listusers** (section 20.5).

% **pts membership system:administrators**

% **kas examine** <*name of user*>
`Password for` *name of user*`:` <*your password*>

% **bos listusers** <*machine name*>

Step 2: Remove the user to *system:administrators*.

% **pts removeuser** <*user name*> **system:administrators**

where

rem is an acceptable abbreviation of **removeuser**.

user name is the user you want to remove from *system:administrators*.

Step 3: Remove the `ADMIN` flag from the Authentication Database entry. You will be prompted for your password.

% **kas setfields** <*name of user*> **NOADMIN**
`Password for` *your user name*`:` <*your password*>

where

sf	is an acceptable alias for **setfields**.
name of user	is the entry from which you want to remove the ADMIN flag.

Step 4: Remove the user from *UserList*.

% **bos removeuser** <*machine name*> <*user names*>

where

removeu	is an acceptable abbreviation of **adduser**.
machine name	should name the system control machine in cells using the United States edition of AFS. If *machine name* is not the system control machine but is running a client portion of the Update Server, the changes you make are overwritten the next time the client portion copies the system control machine's copy of *UserList* (by default, within 5 minutes). In cells using the international edition of AFS, *machine name* should name each file server or database server machine in turn.
user names	is the user name you wish to remove from *UserList*.

Note: If you are using the United States edition of AFS and have changed *UserList* on the system control machine only, it will take a few minutes for the Update Server to distribute the changed file to all file server and database server machines (by default, up to 5 minutes). The newly removed users will be able to issue commands that require this type of privilege until that amount of time passes.

Appendixes

Appendix A
AFSMonitor Statistics

This Appendix provides a complete listing of the statistics fields available with AFSMonitor including the groups and sections into which the fields are organized. A short description of each field, group, and section is provided. See Chapter 7 for instructions on using **AFSMonitor**.

A.1 The Cache Manager Statistics A-2
A.2 The File Server Statistics A-24

A.1. 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.

A.1.1. Performance Statistics Section (PerfStats_section)

A.1.1.1. 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.
- ProtServerAddr: Address of Protection Server used.
- 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.

A.1.1.2. 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.

A.1.2. Server Up/Down Statistics Section (Server_UpDown_section)

A.1.2.1. 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 **AFSMonitor** 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.

A.1.2.2. 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.

A.1.2.3. 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.

A.1.2.4. 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.

A.1.3. RPC Operation Measurements Section (RPCop_section)

A.1.3.1. 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.

A.1.3.2. 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.

A.1.3.3. 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.

A.1.3.4. 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.
- 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.
- 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.
- 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.
- 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.
- 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.
- 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.

A.1.4. Authentication and Replicated File Access Section (Auth_Access_section)

A.1.4.1. 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.

A.1.4.2. 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.

A.2. 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.

A.2.1. Performance Statistics Section (PerfStats_section)

A.2.1.1. 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)

A.2.1.2. 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.

A.2.1.3. 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.

A.2.1.4. 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.

A.2.1.5. Miscellaneous Variables Group (misc_group)

- numPerfCalls: Number of performance calls received.

A.2.2. RPC Operations Section (RPCop_section)

A.2.2.1. 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.

A.2.2.2. 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 B
AIX Audit Events

This Appendix provides a complete listing of the AFS events that can be audited on AIX file server machines. See Chapter 7 for instructions on auditing AFS events on AIX file server machines.

B.1 Introduction B-2
B.2 Audit-Specific Events. B-3
B.3 Volume Server Events B-4
B.4 Backup Server Events B-7
B.5 Protection Server Events B-10
B.6 Authentication Events B-12
B.7 File Server and Cache Manager Interface Events B-14
B.8 BOS Server Events. B-16
B.9 Volume Location Server Events. B-19

B.1. 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 may 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

B.2. Audit-Specific Events

Event\ Description	Class	Parameters
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.

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).

B.3. Volume Server Events

Event\ Description	Class	Parameters
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. The transaction number specifies the volume ID.		
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.		
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.		

Event\ Description	Class	Parameters
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.		

Event\ Description	Class	Parameters
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.		
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.		

B.4. Backup Server Events

Event\ Description	Class	Parameters
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.		
AFS_BUDB_GetTxt	P	ECode AFSName HostID Type
BUDB_GetText - Get the information about hosts/volume-sets/dump-hierarchy.		

Event\ Description	Class	Parameters
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.		

Event\ Description	Class	Parameters
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.		

B.5. Protection Server Events

Event\ Description	Class	Parameters
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 name-to-ID translation(s).		
AFS_PTS_IdToNm	None	ECode AFSName HostID GroupId
PR_IDToName - Perform ID-to-name translation(s).		
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, and/or owner.		
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.		

Event\ Description	Class	Parameters
AFS_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.		
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?		

B.6. Authentication Events

Event\ Description	Class	Parameters
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.		
AFS_KAM_GetEnt	S	ECode AFSName HostID name instance
KAM_GetEntry - The RPC made by **kas examine** 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.		

Event\ Description	Class	Parameters
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.

B.7. File Server and Cache Manager Interface Events

Event\ Description	Class	Parameters
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.		

Event\ Description	Class	Parameters
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.		
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.		

B.8. BOS Server Events

Event\ Description	Class	Parameters
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.		
AFS_BOS_AddSUsr	S P	ECode AFSName HostID
BOZO_AddSUser - Add a user to the UserList.		

Event\ Description	Class	Parameters
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.		

Event\ Description	Class	Parameters
AFS_BOS_DoExec	P C	ECode exec
The **bosserver** process was restarted.		
AFS_BOS_StpProc	P C	ECode cmd
An RPC to stop any process controlled by the BOS Server.		

B.9. Volume Location Server Events

Event\ Description	Class	Parameters
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

ContentsEnv tc0

.BAK version of binary file
  checking compilation date on 11-11
  created by bos install command 11-3
  removing obsolete 11-12
  used by bos uninstall command 11-9
.OLD version of binary file
  checking compilation date on 11-11
  created by bos install command 11-3
  removing obsolete 11-12
  used by bos uninstall command 11-9

/afs directory 2-9, 2-13
/afs/cell_name/common/etc/CellServDB
  as central update source for clients 15-3
/afs/cell_name/service/etc/CellServDB.local 2-10
/afs/cell_name/usr
  grouping home directories under 2-33
/afs/transarc.com/service/etc/CellServDB.export
  as global update source 2-10, 15-3
/etc/fstab file
  adding new file server machine disk 11-14
  removing file server machine disk 11-17
/etc/package 14-22
/etc/passwd
  See also Local password file
/etc/rc
  starting BOS Server in 4-30
/usr/afs/backup
  assigning ownership 6-15
  creating 6-14
/usr/afs/backup/tapeconfig 6-14
  assigning ownership 6-15
  creating 6-14
/usr/afs/bin 3-10, 11-2
  removing obsolete .BAK and .OLD files 11-12
/usr/afs/bin/backup 3-10
/usr/afs/bin/bos 3-10
/usr/afs/bin/bosserver 3-4, 3-10
/usr/afs/bin/buserver 3-8, 3-10
/usr/afs/bin/fileserver 3-3, 3-11
/usr/afs/bin/kas 3-11
/usr/afs/bin/kaserver 3-5, 3-11
/usr/afs/bin/ntpd 3-11
/usr/afs/bin/ntpdc 3-11
/usr/afs/bin/pts 3-11
/usr/afs/bin/ptserver 3-5, 3-11
/usr/afs/bin/runntp 3-9, 3-11
/usr/afs/bin/salvager 3-3, 3-11
/usr/afs/bin/udebug 3-11
/usr/afs/bin/upclient 3-7, 3-11
/usr/afs/bin/upserver 3-7, 3-11
/usr/afs/bin/vlserver 3-6, 3-11
/usr/afs/bin/volserver 3-3, 3-11
/usr/afs/bin/vos 3-11
/usr/afs/db directory 3-15
/usr/afs/db/bdb.DB0 3-15
/usr/afs/db/bdb.DBSYS1 3-15
/usr/afs/db/kaserver.DB0 3-15
/usr/afs/db/kaserver.DBSYS1 3-15
/usr/afs/db/prdb.DB0 3-15
/usr/afs/db/prdb.DBSYS1 3-15
/usr/afs/db/vldb.DB0 3-15
/usr/afs/db/vldb.DBSYS1 3-15
/usr/afs/etc directory 3-12
/usr/afs/etc/CellServDB (server) 1-18
  See also CellServDB file (server)
/usr/afs/etc/KeyFile
  See also KeyFile file
/usr/afs/etc/ThisCell (server)
  See also ThisCell file (server)
/usr/afs/etc/UserList
  See also UserList file
/usr/afs/local directory 3-14
/usr/afs/local/BosConfig 3-14
  See also BosConfig file
/usr/afs/local/NoAuth 3-14, 8-3
/usr/afs/local/SALVAGE.fs 3-14
/usr/afs/local/salvage.lock 3-14
/usr/afs/logs 3-16
  removing core files 11-12
/usr/afs/logs/AuthLog 3-16
  examining 4-28
/usr/afs/logs/BackupLog 3-16
  examining 4-28
/usr/afs/logs/BosLog 3-16
  examining 4-28
/usr/afs/logs/FileLog 3-16
  examining 4-28
/usr/afs/logs/SalvageLog 3-16
  examining 4-28
/usr/afs/logs/VolserLog 3-16
  examining 4-28
/usr/vice/cache directory 12-5
/usr/vice/cache/AFSLog 12-5
/usr/vice/cache/CacheItems
  See also CacheItems file
/usr/vice/cache/V-digits
  See also V file
/usr/vice/cache/VolumeItems
  See also VolumeItems file
/usr/vice/etc directory 12-3
/usr/vice/etc/afsd
  See also afsd
/usr/vice/etc/AFSLog 13-22
/usr/vice/etc/cacheinfo
  See also cacheinfo file
/usr/vice/etc/CellServDB (client)
  See also CellServDB file (client)
/usr/vice/etc/ThisCell (client)
  See also ThisCell file (client)

/vicepx directories 3-17
/vicepx/FORCESALVAGE 3-17

@sys variable in pathnames 2-29
UNIX UID
  matching AFS UID 17-3

A access right (on ACL) 19-5
Access
  transparent 1-7, 5-4
Access control list
  See also ACL
Access rights on ACLs 19-4
  See also ACL, Index entries for individual rights
Accesses
  of volume, counter of 5-7
Account
  See also User account
ACL 1-13, 19-2
  access rights defined 19-4
  changing default assigned when creating user account 17-12
  changing entries, about 19-11
  cleaning 19-21
  clearing 19-18, 19-19
  default rights assigned when volume created 18-8
  default set by vos create 5-10
  denying access on 19-14
  granting access on 19-13
  listing entries on 19-9
  normal vs. negative rights 19-6
  placing foreign users on 19-8
  removing obsolete AFS UIDs from 19-21
  replacing all entries 19-18, 19-19
  restoring access after denying it 19-16
  setting for directory with uss 16-27
  setting for home directory with uss 16-24
  shorthand groupings for defining entries 19-5
  undefined access rights 19-6
  using groups on 19-7
  using system-defined groups on 18-8, 19-7
Active clients statistic in Scout 7-4
Add privacy flag on Protection Database entry 18-36
Adding
  ADMIN flag to Authentication Database entry 20-5
  database server machine to CellServDB file (client) 15-10
  database server machine to CellServDB file (server) 9-6
  database server machine to client's kernel list 15-8
  disk to file server machine 11-13
  entry to ACL 19-13
  members to groups 18-25
  members to system:administrators group 20-4
  ReadOnly site definition in VLDB 5-18
  server encryption key to KeyFile file 10-9
  users to UserList file 20-7
ADMIN flag in Authentication Database entry
  adding 20-5
  checking for 20-5
  privileges of 20-2
  removing 20-6
Administer access right (on ACL) 19-5
AFS
  accessing through NFS 2-59
  authentication separate from UNIX 2-38
  differences from UNIX, summarized 2-2
  global name space 1-7, 2-9
  list of server processes used in 1-11
  list of supported system types 2-22
  reducing traffic in 1-10
  security features 2-47
  server encryption key (see entry: *Server encryption key*) 10-2
afs entry in Authentication Database 10-4
AFS events
  auditing on AIX File Servers 7-29
AFS UID
  assigning 17-3
  assigning for group 18-21
  assigning in uss 16-8
  assigning with pts 18-16, 18-18, 18-21
  defined 18-3
  functional difference from UNIX UID 1-14
  learning, given group name 18-10
  learning, given machine IP address 18-13
  learning, given user name 18-13
  listing counters for automatic allocation 18-34
  matching AFS UID 2-32
  matching with existing UNIX UID 17-3
  matching with existing UNIX UID 16-8
  removing obsolete from ACL 19-21
  reserved for anonymous 2-32
  reserved for system-defined groups 2-36
  reusing, about 18-9
  setting counters for automatic allocation 18-35
AFSCELL Environment Variable 6-28
Afsd 12-3
Afsd command
  configuration scripts 13-12
AFSLog file 12-5, 13-22
AFSMonitor
  available statistics A-1
  cache manager statistics A-2
  command syntax 7-20
  creating an output file for 7-24
  creating configuration files for 7-21
  features summarized 7-14
  file server statistics A-24
  requirements for running 7-14
  screen layout 7-15
  setting terminal type 7-14
  starting 7-19
  stopping 7-21
Afsmonitor command 7-20
AIX
  auditing AFS events in 7-29
AIX auditing
  AFS events listed B-1
All grouping of access rights
  using with fs setacl command 19-5
All-or-nothing release of ReadOnly volumes 5-14
Anonymous user
  AFS UID reserved for 2-32
  identity assigned to unauthenticated user 8-2
Apropos commands xxvi
Argument
  defined xviii
  list-taking xviii
Assigning

AFS UID 17-3
AFS UID for group 18-21
AFS UID with pts 18-16, 18-18
AFS UID with uss 16-8
Attention highlighting levels in Scout 7-5
Auditing AFS Events
on AIX File Servers 7-29
Authenticated identity
acquiring with klog 2-44
Authentication
AFS compared to UNIX 2-3
AFS separate from UNIX 2-38
compared to authorization checking 2-49, 8-2
consequences of multiple failures 2-41
improving security of 17-14
See also Authorization checking, Mutual authentication
Authentication Database
afs entry in 10-4
changing passwords 2-42
creating entry with uss 16-39
creating entry without uss 17-8
deleting entry with uss 16-45
listing server encryption key from 10-7
removing entry 17-26
renaming entry 17-23
restricting reuse of passwords 2-43
setting lifetime of 2-42
setting password, instructions 17-20
site for server encryption key 10-4
Authentication Server
about starting and stopping 4-5
as kaserver process 3-4
basic description 1-13
examining log file 4-28
on database server machine 1-21
restarting after changes to CellServDB file (server) 9-6, 9-8
when to contact 3-5
AuthLog file 3-16
examining 4-28
Authorization checking
and restarting processes 8-4
compared to authentication 2-49, 8-2
controlling cell-wide 8-3
defined 8-2
turning off 8-4
turning on 8-4
Automatic daily restart for new binaries
default time for 11-7
listing time for 4-25, 11-7
setting time for 4-27, 11-8
Automatic updates
in Ubik 2-52
Automatic weekly restart of BOS Server
about 4-25
default time for 4-25
listing time for 4-25
setting time for 4-26
Automating
creation of Backup volumes 5-21
Availability of data
interrupted by dumping 5-58

Backing up
Backup Database to tape 6-59
data 6-43
dump ID numbers 6-45
full dumps 6-43
in a foreign cell 6-28
incremental dumps 6-43
tapes 6-45
Backup adddump command 6-23
Backup addhost command 6-16, 6-18
Backup addvolentry command 6-21
Backup addvolset command 6-20
Backup command
to enter interactive mode 6-29
Backup commands 3-8, 6-1
granting privilege for 20-7
Backup Database 1-15, 6-11
adding dump levels 6-23
adding Tape Coordinators 6-16
adding volume entries 6-21
adding volume sets 6-20
administering 6-59
Backup System 1-15
changing expiration dates 6-24
deleting dump levels 6-25
deleting volume entries 6-22
deleting volume sets 6-21
displaying backup information 6-37
displaying dump hierarchy 6-36
displaying dump history of a volume 6-40
displaying dump ID numbers 6-37, 6-40
displaying dump levels 6-36
displaying dump sets 6-37
displaying port offset numbers 6-39
expiration dates 6-23
listing information 6-35
listing volume entries 6-36
listing volume sets 6-36
recovering backup information from dump tapes 6-60
restoring from Backup Database tapes 6-60
restoring from tape 6-59
saving to tape 6-59
setting expiration dates 6-23
verifying 6-35
Backup dbverify command 6-35
Backup deldump command 6-25
Backup deletedump command 6-65
Backup delhost command 6-18
Backup delvolentry command 6-22
Backup delvolset command 6-21
Backup diskrestore command 6-58
Backup dump command 6-47
Backup dumpinfo command 6-37
Backup jobs command 6-31
Backup kill command 6-32
Backup labeltape command 6-26
Backup listdumps command 6-36
Backup listhosts command 6-39
Backup listvolsets command 6-36
Backup quit command 6-30
Backup readlabel command 6-27
Backup restoredb 6-62
Backup savedb 6-60, 6-64
Backup scantape command 6-41, 6-63
Backup Server
about starting and stopping 4-5

as buserver process 3-8
Backup System 1-15
basic description 1-15
examining log file 4-28
restarting after changes to CellServDB file (server) 9-6, 9-8
when to contact 3-8
Backup setexp command 6-24
Backup status command 6-33
Backup System 1-15
/usr/afs/etc/UserList file 6-15
about 2-57
automatic scheduling of dumps 6-51
backing up data 6-43
backing up the Backup Database 6-59
basic description 1-15
canceling operations in interactive mode 6-30
configuring 6-12
creating /usr/afs/backup 6-14
creating /usr/afs/backup/tapeconfig 6-14
date-specific restores 6-55
deleting backup information 6-63
disabling initial tape prompt 6-50
displaying dump ID numbers 6-37, 6-40
displaying job ID numbers 6-30
dump 6-4
dump hierarchy 6-4, 6-22
dump ID numbers 6-6, 6-45
dump labels 6-10
dump level 6-4
dump levels 6-22
dump names 6-6
dump set 6-4
dump sets 6-43
dumping data 6-43
EOF markers 6-10, 6-12
expiration dates 6-6
full dumps 6-43
full restores 6-55
further automation of 6-50
incremental dumps 6-43
interactive mode 6-28
interfaces 6-28
introduction 6-3
job ID numbers 6-29, 6-30
labeling tapes 6-26
listing information 6-35
listing operations in interactive mode 6-30
multiple dumps on a dump set 6-47
overview 6-44
port offset numbers 6-11
reading tape labels 6-27
recovering backup information from dump tape 6-60
recovering data 6-55
reducing operator intervention in 6-50
restoring backed up data 6-9
restoring backup data 6-55
restoring data 6-55
restoring the Backup Database 6-59, 6-60
scanning tapes 6-41
starting Tape Coordinators 6-32
status of Tape Coordinators 6-33
stopping Tape Coordinators 6-33
tape archiving 6-8
Tape Coordinator machines 6-11, 6-13
Tape Coordinators 6-17, 6-32
tape labels 6-10, 6-25
tape names 6-6
tape recycling schedules 6-7
tape size 6-12
tapes 6-12
task ID numbers 6-29
the Backup Database 6-11
useCounts of tapes 6-8
volume entries 6-3, 6-18
volume sets 6-3, 6-18
warnings about backups 6-43
Backup volinfo command 6-40
Backup volrestore command 6-56
Backup volume
about 2-57
automating creation of 5-21
changing name of 5-63
creating 5-20
creating multiple at once 5-20, 5-23
defined 5-3
dumping 5-58
ID number in VLDB 5-5
ID number in volume header 5-6
mounting 5-21
moving 5-40
removed by ReadWrite move 5-40
removed by ReadWrite removal 5-53
removing, effect of 5-54
removing, how to 5-57
space-saving nature of 5-13
suggested schedule for creation of 5-21
BackupLog file 3-16
examining 4-28
BAK version of binary file
See also .BAK (at beginning of Index)
Banner line on Scout screen 7-3
Basenames in Scout 7-3
Basic OverSeer Server
See also BOS Server
Bdb.DB0 file 3-15
Bdb.DBSYS1 file 3-15
Binaries
See also Command suite binaries, Server process binaries
Binary distribution machine
defined 1-19, 3-10
identifying with bos status 4-11
need for one of each CPU type 11-2
Binary file
See also Command suite binaries, Server process binaries
Bos addhost command 9-6
Bos addkey command 10-10
when handling key emergency 10-18
Bos adduser command 20-7
Bos commands 3-4, 3-7
granting privilege for 20-7
preventing mutual authentication for 8-5
Bos create command
for cron-type process 4-16
for fs-type process 4-16
for simple-type process 4-14
Bos delete command 4-18
Bos exec command

for rebooting file server machine 4-31
Bos getdate command 11-11
Bos getlog command 4-28
Bos getrestart command 4-25
Bos install command 11-4
Bos listhosts command 4-11, 9-5
Bos listkeys command 10-7
Bos listusers command 20-7
Bos prune command 11-12
Bos removehost command 9-8
Bos removekey command 10-12
Bos removeuser command 20-8
Bos restart command
after changes to CellServDB file (server) 9-7, 9-9
excluding BOS Server 4-24
for selected processes 4-24
including BOS Server 4-23
using -bosserver flag with 4-23
when handling key emergency 10-17
Bos salvage command
for all volumes on file server machine 5-47
for all volumes on partition 5-47
for single volume 5-48
BOS Server
as bosserver process 3-4
as maintainer of server process binaries 11-2
automatic weekly restart 4-25
basic description 1-12
default times for restart 4-25
examining log file 4-28
memory state 4-5
monitoring server processes 4-1
restart times 4-25
starting up in /etc/rc 4-30
use of BosConfig file 4-5
when to contact 3-4
Bos setauth command 8-3, 8-4
Bos setrestart command 4-26, 4-27
Bos shutdown command
adding new file server machine disk 11-14
preparing for local reboot 4-30
preparing for remote reboot 4-31
removing file server machine disk 11-17
Bos status command 4-7
output explained 4-8, 4-9
Bos stop command 4-18, 4-19
Bos uninstall command 11-10
BosConfig file 3-14
information in 4-3
listing entries in 4-8
restart times in 4-25
BosLog file 3-16
examining 4-28
Bosserver process 3-4
See also BOS Server
Bulk mode in uss 16-48
Buserver process 3-8
See also Backup Server
Butc command 6-32

Cache
See also Data cache
Cache files (client) 12-5
Cache Manager
as interpreter of mount points 5-8
changing data cache size temporarily 13-6
collecting data about 2-26
collecting data with xstat 7-25
configuration scripts 13-12
controlling running of setuid programs 13-15
customizable aspects of 13-2
debugging trace 13-21
described 1-16, 12-2
displaying file server machine preferences 13-26
file server machine preferences 2-29
flushing cache 13-24
listing cache size from cacheinfo file 13-4
listing current data cache size 13-6
logging and disabling messages from 13-19
monitoring performance 13-21
need to know about database server machines 15-2
rc.afsd.large script 13-12
rc.afsd.med script 13-12
rc.afsd.small script 13-12
resetting disk cache size to default value 13-6
setting disk cache location 13-4
setting file server machine preferences 13-26
setting home cell 13-18
use of CellServDB file (client) 15-2
using configuration scripts 13-13
xstat data collections 7-25
xstat example commands 7-26
xstat libraries 7-25
xstat_cm_test example command 7-27
Cacheinfo file 12-3
listing contents 13-4, 13-6
resetting disk cache to size specified in 13-6
setting disk cache location 13-4
CacheItems file 12-5
Caching 1-9, 12-2
Callback 1-10, 12-2
Cell 1-7
advice for setting up file tree 2-13
changing client's kernel list of 15-8
choosing name for 2-6
foreign 1-7
granting local access to foreign users 2-11
local 1-7
making foreign visible to local 2-10
making local visible to foreign 2-10
name at second level in file tree 2-9, 2-13
setting home cell for client machine 13-18
setting name of 2-7
CellServDB file (client)
about 2-11, 12-3, 15-2
central update source 15-13
changing 15-10
copied into kernel list 15-2
correct format 15-3
effect of wrong information in 15-4
global source at Transarc 15-3
listing contents of 15-6
maintaining 15-3
updating from central source without package 15-11
updating with package 15-10
CellServDB file (server)
about 1-18, 2-56, 3-12, 9-2
adding database server machine to 9-6
effect of wrong information in 9-2
listing contents of 9-5

removing database server machine from 9-8
rules for maintaining 9-3
Ubik's use of 9-2
Cellular mount point 5-26
Central update source
for *CellServDB* file (client) 15-13
Changing
data cache size, temporarily 13-6
default ACL assigned when creating user account 17-12
default quota assigned when creating user account 17-12
disk cache location, in cacheinfo file 13-4
disk cache size to default value 13-6
entries on ACL, about 19-11
group ownership to self-owned 18-24
group-creation quota 18-32
mount point when renaming user 17-24
name of Protection Database entry 18-30
owner of Protection Database entry 18-28
user name 17-22
volume name 5-63
volume name when renaming user 17-23
Checking
time stamp on binary file 11-11
Checksum 10-6
Chgrp command
AFS compared to UNIX 2-3
Choosing
cell name 2-6
name for user volume mount point 2-33
user name, advice on 2-31
user volume name, advice on 2-32
Chown command
AFS compared to UNIX 2-3
Client
configuring local disk with package 14-1
modifying to run package 14-22
Client machine
advice for configuring 2-27
basic configuration requirements 12-1
cache files in /usr/vice/etc 12-5
changing CellServDB file 15-10
changing data cache size temporarily 13-6
changing kernel list of cells 15-8
configuration files in /usr/vice/etc 12-3
controlling access to foreign cell 2-28
controlling running of setuid programs 13-15
defined 1-6
files required on local disk 2-27
flushing data cache 13-24
listing current data cache size 13-6
listing data cache size from cacheinfo file 13-4
listing kernel list of cells 15-6
logging and disabling messages from 13-19
making cell's file tree visible on 2-11
making cell's file tree visible to 15-2
monitoring performance 13-21
need to know about database server machines 15-2
resetting disk cache size to default value 13-6
setting disk cache location 13-4
setting home cell 13-18
setting home cell for 2-7
setting the File Server probe interval 13-17
using configuration scripts 13-12
Client machines statistic in Scout 7-4
Client portion
See also Update Server
Client, defined 1-6
Clone 1-9, 5-14
forcing creation of new 5-15
Cloning
defined 5-13
for backup 5-20
for replication 5-14
See also Replication
Collecting
data with xstat 7-25
Command
executing from uss template file 16-37
Command interpreters
use of CellServDB file (client) 15-2
Command parameters
in BosConfig file 4-3
Command suite
binaries, checking time stamp 11-11
binaries, installing 11-3
binaries, removing obsolete 11-12
binaries, reverting to old version 11-9
binaries, uninstalling 11-9
defined xviii
Commands
afsmonitor 7-20
argument defined xviii
backup (to enter interactive mode) 6-29
backup adddump 6-23
backup addhost 6-16, 6-18
backup addvolentry 6-21
backup addvolset 6-20
backup dbverify 6-35
backup deldump 6-25
backup deletedump 6-65
backup delhost 6-18
backup delvolentry 6-22
backup delvolset 6-21
backup diskrestore 6-58
backup dump 6-47
backup dumpinfo 6-37
backup jobs 6-31
backup kill 6-32
backup labeltape 6-26
backup listdumps 6-36
backup listhosts 6-39
backup listvolsets 6-36
backup quit 6-30
backup readlabel 6-27
backup restoredb 6-62
backup savedb 6-60, 6-64
backup scantape 6-41, 6-63
backup setexp 6-24
backup status 6-33
backup volinfo 6-40
backup volrestore 6-56
backupsys 5-23
bos addhost 9-6
bos addkey 10-10
bos adduser 20-7
bos create, for cron-type process 4-16
bos create, for fs-type process 4-16
bos create, for simple-type process 4-14

bos delete 4-18
bos exec 4-31
bos getdate 11-11
bos getlog 4-28
bos getrestart 4-25
bos install 11-4
bos listhosts 4-11, 9-5
bos listkeys 10-7
bos listusers 20-7
bos prune 11-12
bos removehost 9-8
bos removekey 10-12
bos removeuser 20-8
bos restart, excluding BOS Server 4-24
bos restart, for selected processes 4-24
bos restart, including BOS Server 4-23
bos salvage 5-47
bos setauth 8-3, 8-4
bos setrestart 4-26, 4-27
bos shutdown 4-30, 4-31, 11-14, 11-17
bos status 4-7
bos stop 4-18, 4-19
bos uninstall 11-10
butc 6-32
chgrp (AFS compared to UNIX) 2-3
chown (AFS compared to UNIX) 2-3
fms 6-12
fs checkservers 13-17
fs cleanacl 19-22
fs debug 13-21
fs examine 5-38, 5-51
fs flush 13-24
fs flushvolume 13-25
fs getcacheparms 13-6
fs getcellstatus 13-15
fs getserverprefs 13-27
fs listacl 19-9
fs listcells 15-6
fs listquota 5-37, 5-50
fs lsmount 5-29
fs messages 13-19, 13-20
fs mkmount 5-11, 5-22, 5-28, 5-62, 5-64, 17-10, 17-24
fs monitor 13-23
fs newcell 15-8
fs quota 5-50
fs rmmount 5-29, 5-56, 5-57, 5-63, 17-24, 17-26
fs setacl 17-12, 19-11
fs setcachesize 13-6
fs setcell 13-15
fs setquota 5-49, 17-12
fs setserverprefs 13-28
fs setvol 5-50
fs whereis 5-39
fsck (AFS compared to UNIX) 2-4
fsck (AFS version) 2-4
ftp (AFS compared to UNIX) 2-58
ftpd (AFS compared to UNIX) 2-3, 2-58
groups (AFS compared to UNIX) 2-3
help (online) for xxv
inetd (AFS compared to UNIX) 2-3, 2-58
instance defined xviii
kas create 17-10, 17-23
kas delete 17-23, 17-26
kas examine 10-7
kas interactive 8-5
kas setfields 2-41, 2-42, 2-43, 17-16, 17-18
kas setpassword 2-42, 10-11, 17-20
kas unlock 17-17
keywords for looking up xxvi
klog 2-44
klog.krb 2-46
kpasswd 2-42, 2-43
ln (AFS compared to UNIX) 2-3
login (AFS version) 2-40
login.krb 2-46
man xxvii
operation code defined xviii
pagsh 2-43
pagsh.krb 2-46
privileged 8-2
pts adduser 18-25
pts chown 18-29
pts createuser 17-9, 18-16, 18-19
pts delete 17-27, 18-27
pts examine (for group) 18-10
pts examine (for machine) 18-13
pts examine (for user) 18-13
pts listmax 18-34
pts listowned (for group) 18-12
pts listowned (for user) 18-15
pts membership (for group) 18-11
pts membership (for user) 18-14
pts removeuser 18-26
pts rename 17-22, 18-31
pts setfields 18-32, 18-38
pts setmax 18-35
rcp (AFS compared to UNIX) 2-3, 2-58
rlogind (AFS compared to UNIX) 2-3, 2-58
rsh (AFS compared to UNIX) 2-3, 2-58
scout 7-8
suite defined xviii
switch defined xviii
tokens 2-44
tokens.krb 2-46
umount 11-17
unlog 2-45
uss add 16-41
uss bulk 16-52
uss delete 16-46
vos addsite 5-18
vos backup 5-22
vos create 5-11, 17-10
vos delentry 5-55
vos dump 5-59
vos examine 5-37
vos listpart 11-14
vos listvldb 5-31
vos listvol 5-33
vos lock 5-66
vos move 5-41, 11-17
vos partinfo 5-10, 5-17
vos release 5-15, 5-18
vos remove 5-55, 5-56, 5-57, 17-26
vos remsite 5-54
vos rename 5-63, 17-23
vos restore 5-60, 5-62
vos syncserv 5-44
vos syncvldb 5-44
vos unlock 5-66
vos unlockvldb 5-66

vos zap 5-54
xstat_cm_test 7-27
xstat_fs_test 7-26
Common configuration files (server) 3-12
Compilation
date of, listing on binary file 11-11
Compiling
package prototype file 14-21
Complex mutual authentication
See also Mutual authentication
Computing environment
types compared 1-5
Configuration file lines in package 14-9
Configuration files (client) 12-3
Configuration files 1-18
in package 14-3
See also Common configuration files (server), Configuration files (client), Local configuration files (server)
Configuration scripts 13-12
effect 13-12
rc.afsd.large 13-12
rc.afsd.med 13-12
rc.afsd.small 13-12
using 13-13
Configuring
file server machines, about 3-12
AFSMonitor 7-21
Cache Manager to control performance 13-2
client machine, advice 2-27
file server machine, advice 2-23
file tree, advice 2-13
local disk of client with package 14-1
Connections statistic in Scout 7-4
Consistency guarantees
in data cache 1-10
in Ubik 2-52
Constants
in uss template file 16-14
Contacting processes
Authentication Server 3-5
Backup Server 3-8
BOS Server 3-4
File Server 3-3
NTPD 3-9
Protection Server 3-6
Salvager 3-3
Update Server 3-7
VL Server 3-6
Volume Server 3-3
Controlling authorization checking
for entire cell 8-3
Conventions
AFS pathnames 2-9
cell name 2-6
volume names 2-16, 5-9
Converting
existing UNIX accounts to AFS accounts 17-5
existing UNIX accounts to AFS accounts 16-10
Coordinator
election in Ubik 2-53
in Ubik 2-52
Core files
for server processes 3-16
Core leak
preventing with scheduled restarts 4-25
Correspondence
of volumes and directories 1-8
Corruption
symptoms of 5-45
types of 5-45
Counter
Protection Database's for AFS UID 18-34
CPS 18-8
Creating
Authentication Database entry with uss 16-39
Authentication Database entry without uss 17-8
Backup volume 5-20
cellular mount point 5-28
directory with uss 16-25
entry in /etc/passwd with uss 16-32
entry on ACL 19-13
entry on Negative rights part of ACL 19-14
file with uss 16-28, 16-30
hard link with uss 16-34
mount point in foreign cell 5-30
mount point when changing user name 17-24
mount point with uss 16-22
multiple Backup volumes at once 5-20
PAG with pagsh command 2-43
Protection Database machine entry with pts 18-18
Protection Database multiple group entries 18-23
Protection Database prefix-less group entry 18-23
Protection Database regular group entry 18-22
Protection Database user entry with pts 18-16
Protection Database user entry with uss 16-39
Protection Database user entry without uss 17-8
ReadOnly volume 5-14
ReadWrite mount point 5-28
ReadWrite volume 5-10
regular mount point 5-27
self-owned group 18-24
server encryption key 10-9
server process 4-14
standard files in new user account 2-34
symbolic link with uss 16-34
ticket (tokens) for server process 2-44
user account with uss 16-39
user account without uss 17-8
user accounts in bulk 16-48
volume with uss 16-19
Creation date
recorded in volume header 5-6
Creator
of Protection Database entry 18-3
of Protection Database entry, listing 18-10, 18-13
Criteria for replicating volumes 5-14
Cron-type server process
defined 4-5
defining in BosConfig file 4-16
defining time for 4-17
used to automate volume backup 5-21
Current protection subgroup 18-8
Curses graphics utility
and AFSMonitor 7-14
and Scout 7-2

D access right (on ACL) 19-5
Daily restart for new binaries
automatic, about 11-7

listing time for 4-25, 11-7
setting time for 4-27, 11-8
Data
availability interrupted by dumping 5-58
Data cache
changing location of disk cache 13-4
changing size temporarily 13-6
disk versus memory 13-4
flushing (forcing update) 13-24
listing current size 13-6
listing size specified in cacheinfo file 13-4
recommended size 13-5
resetting disk cache size to default value 13-6
using configuration scripts 13-12
V file in 12-5
Data collection
with xstat 7-25
Database files 3-15
Database server machine
adding to CellServDB file (client) 15-10
adding to CellServDB file (server) 9-6
adding to client's kernel list 15-8
client knowledge of 15-2
defined 1-20
identifying with bos status 4-11
listing from CellServDB file (client) 15-6
listing from CellServDB file (server) 9-5
reason to run three 2-24
removing from CellServDB file (client) 15-10
removing from CellServDB file (server) 9-8
removing from client's kernel list 15-8
Database server process
about starting and stopping 4-5
need to know about peers 9-2
restarting after changes to CellServDB file (server) 9-6, 9-8
Databases, distributed 2-51
Date
on binary file, listing 11-11
Date-specific restores 6-55
Debugging
trace for Cache Manager 13-21
Default
ACL 5-10
volume quota 5-10, 5-49
Defining
directory with uss 16-24
ReadOnly site in VLDB 5-18
server encryption key 10-9
server process in BosConfig file 4-14
Delete access right (on ACL) 19-5
Deleting
Authentication Database entry with uss 16-45
Protection Database user entry with uss 16-45
user account with uss 16-45
user accounts in bulk 16-48
user accounts with uss, issues to consider 16-45
See also Removing
Denying
access on ACL 19-14
privilege of all three types 20-11
Desynchronization of VLDB/volume headers
fixing 5-42
symptoms of 5-42
Determining
identity of binary distribution machine 4-11
identity of database server machines 4-11
identity of simple file server machines 4-12
identity of system control machine 4-11
role taken by file server machine 4-11
success of replication 5-14
Device names
in /usr/afs/backup/tapeconfig 6-14
Differences
between AFS and UNIX, summarized 2-2
Directories (client)
/usr/vice/cache 12-5
/usr/vice/etc 12-3
Directories (server)
/usr/afs/bin 3-10, 11-2
/usr/afs/db 3-15
/usr/afs/etc 3-12
/usr/afs/local 3-14
/usr/afs/logs 3-16
/vicepx 3-17
Directories
/afs 2-9, 2-13
/afs/cell_name 2-9, 2-13
/usr/afs/backup 6-15
conventional under /afs/cell_name 2-14
for grouping user home directories 2-33
lost+found 2-4
Directory
creating with uss 16-25
defining with uss 16-24
flushing from data cache on client machine 13-24
overwriting with uss 16-6
See also Root directory
Directory protection
See also Protection
Directory/file name
translating to volume ID number 5-38
translating to volume location 5-39
translating to volume name 5-37
Directory/volume correspondence 1-8
Disabling
authorization checking 8-4
Discarding
ticket/tokens 2-45
Disk
installing on file server machine 11-13
local (see entry: *Local disk*) 14-1
removing from file server machine 11-16
Disk cache
See also Data cache
Disk partition
grouping related volumes on 2-18
listing size of single 5-51
monitoring usage of 7-5
moving volumes to reduce overcrowding 5-40
salvaging all volumes on 5-47
Displaying
Cache Manager preferences for file server machines 13-26
Distributed databases 2-51
Distributed file system 1-5
Distribution
of databases 2-51
security issues 1-22
Distribution machine

See also Binary distribution machine
Dumb terminal
use in Scout 7-2
use with AFSMonitor 7-14
Dump 6-4
full 6-4
incremental 6-4
parent 6-4
Dump hierarchy 6-4, 6-22
adding dump levels 6-23
changing expiration dates 6-24
deleting dump levels 6-25
displaying 6-36
expiration dates 6-23
setting expiration dates 6-23
Dump ID numbers 6-6, 6-45
displaying 6-37, 6-40
Dump labels 6-10
Dump level 6-4
Dump levels 6-22
adding to Backup Database 6-23
changing expiration dates 6-24
deleting from Backup Database 6-25
displaying 6-36
expiration dates 6-6, 6-23
full 6-22
incremental 6-22
parent 6-22
setting expiration dates 6-23
Dump names 6-6
Dump set 6-4
Dump sets
creating 6-43
deleting from Backup Database 6-63
displaying 6-37
full dumps 6-43
incremental dumps 6-43
Dumping
Backup Database to tape 6-59
data 6-43
dump ID numbers 6-45
full dumps 6-43
incremental dumps 6-43
overview 6-44
tapes 6-45
volumes, fully vs. incrementally 5-58
volumes, reasons for 5-58
volumes, without using AFS Backup System 5-58

Election
of coordinator in Ubik 2-53
Emergency
server encryption keys mismatched 10-14
Enabling authorization checking 8-4
Encrypted network communication 2-48
Encryption key
See also Server encryption key
Entering
kas interactive mode 8-5
Entry in VLDB
information in 5-5
listing information from 5-31
listing, with volume header 5-36
locking 5-65
unlocking 5-65
EOF marker
determining size 6-12
EOF marker size
in /usr/afs/backup/tapeconfig 6-14
EOF markers 6-10
Erasing
all entries from ACL 19-18, 19-19
Examining
ACL 19-9
group's Protection Database entry 18-10
log files for server processes 4-28
machine's Protection Database entry 18-13
mount point 5-29
user's Protection Database entry 18-13
VLDB entry 5-31
VLDB entry with volume header 5-36
volume header 5-32
volume header with VLDB entry 5-36
See also Listing
Examples
a complicated uss template file 16-18
apropos operation code xxvi
help operation code xxv
library files for package 14-7
prototype files for package 14-5
Scout screen 7-9
simple uss template file 16-17
uss template file 16-17
Executing
command using uss template line 16-37
Expiration dates 6-23
absolute 6-23
changing 6-24
relative 6-23
setting 6-23
Extension
See also File extension

-f flag on vos release command 5-15
Failure
of file storage due to full partition 5-40
of uss account creation, avoiding 16-4
of uss account creation, recovering 16-5
of uss account deletion, avoiding 16-4
-fast flag on vos listvol command 5-33
Fetching statistic in Scout 7-4
File
creating standard ones in new user account 2-34
creating with uss 16-28, 16-30, 16-32
flushing from data cache on client machine 13-24
overwriting with uss 16-6
required on client machine local disk 2-27
File extension
.BAK 11-3, 11-9
.OLD 11-3, 11-9
File protection
See also Protection
File Server
as part of fs process 3-2, 4-4
basic description 1-12
building CPS 18-8
collecting data about 2-26
collecting data with xstat 7-25
determining user's access rights 18-8
examining log file 4-28

issuing callbacks 12-2
monitoring with Scout 7-7
when to contact 3-3
xstat data collections 7-25
xstat example commands 7-26
xstat libraries 7-25
xstat_fs_test example command 7-26
File server machine
advice for configuring 2-23
binary distribution role 1-19
Cache Manager preferences 2-29
configuration files in /usr/afs/etc 3-12
configuration files in /usr/afs/local 3-14
core files in /usr/afs/logs 3-16
database files in /usr/afs/db 3-15
database server role 1-20
defined 1-6
determining role of 4-11
displaying Cache Manager preferences 13-26
examining log files 4-28
first installed 2-23
installing command & process binaries 11-3
installing disk on 11-13
log files in /usr/afs/logs 3-16
monitoring outages of 7-7
naming partitions on 3-17
need for identical software on all CPU types 11-3
protecting directories on disk 2-24
rebooting 4-30
rebooting, about 2-26
removing disk from on 11-16
restoring partitions (Backup System) 6-55
roles summarized 1-17
salvaging all volumes on 5-47
setting Cache Manager preferences 13-26
setting home cell for 2-7
simple role 1-17
system control role 1-18
uninstalling command & process binaries 11-9
File Server Probe Interval
setting for a client machine machine 13-17
File storage
failed due to partition crowding 5-40
File system
defined 1-5
monitoring activity 13-21
salvager (see entry:*Salvager*) 3-2
File system salvager
See also Salvager
File tree
conventions for configuring 2-13
creating volumes to match top level directories 2-16
third level conventions 2-14
FileLog file 3-16
examining 4-28
Files
/afs/cell_name/service/etc/CellServDB.local 2-10
/afs/transarc.com/service/etc/CellServDB.export 2-10
/etc/fstab 11-14, 11-17
/etc/rc (on file server machine) 4-30
/usr/afs/backup/tapeconfig 6-14
/usr/afs/bin/backup 3-10
/usr/afs/bin/bos 3-10
/usr/afs/bin/bosserver 3-10
/usr/afs/bin/buserver 3-10
/usr/afs/bin/fileserver 3-11
/usr/afs/bin/kas 3-11
/usr/afs/bin/kaserver 3-11
/usr/afs/bin/ntpd 3-11
/usr/afs/bin/ntpdc 3-11
/usr/afs/bin/pts 3-11
/usr/afs/bin/ptserver 3-11
/usr/afs/bin/runntp 3-11
/usr/afs/bin/salvager 3-11
/usr/afs/bin/udebug 3-11
/usr/afs/bin/upclient 3-11
/usr/afs/bin/upserver 3-11
/usr/afs/bin/vlserver 3-11
/usr/afs/bin/volserver 3-11
/usr/afs/bin/vos 3-11
/usr/afs/db/bdb.DB0 3-15
/usr/afs/db/bdb.DBSYS1 3-15
/usr/afs/db/kaserver.DB0 3-15
/usr/afs/db/kaserver.DBSYS1 3-15
/usr/afs/db/prdb.DB0 3-15
/usr/afs/db/prdb.DBSYS1 3-15
/usr/afs/db/vldb.DB0 3-15
/usr/afs/db/vldb.DBSYS1 3-15
/usr/afs/etc/CellServDB (server) 1-18, 2-56, 3-12, 9-2
/usr/afs/etc/KeyFile 1-18, 3-13, 10-3
/usr/afs/etc/NoAuth 3-14
/usr/afs/etc/SALVAGE.fs 3-14
/usr/afs/etc/salvage.lock 3-14
/usr/afs/etc/ThisCell 1-18
/usr/afs/etc/ThisCell (server) 3-13
/usr/afs/etc/UserList 1-18, 3-13
/usr/afs/local/BosConfig 3-14, 4-3
/usr/afs/local/NoAuth 8-3
/usr/afs/logs/AuthLog 3-16
/usr/afs/logs/BackupLog 3-16
/usr/afs/logs/BosLog 3-16
/usr/afs/logs/FileLog 3-16
/usr/afs/logs/SalvageLog 3-16
/usr/afs/logs/VolserLog 3-16
/usr/vice/cache/AFSLog 12-5
/usr/vice/cache/CacheItems 12-5
/usr/vice/cache/V 12-5
/usr/vice/cache/VolumeItems 12-5
/usr/vice/etc/afsd 12-3
/usr/vice/etc/AFSLog 13-22
/usr/vice/etc/cacheinfo 12-3
/usr/vice/etc/CellServDB (client) 12-3
/usr/vice/etc/ThisCell (client) 12-4
/vicepx/FORCESALVAGE 3-17
CellServDB file (client) 15-2
examining log files 4-28
fms.log 6-12
package Makefile 14-14
Flag xix
See also Site flags, Status flag, Type flags
Flexible synchronization site
in Ubik 2-53
Flushing
data cache on client machine 13-24
Fms command 6-12
Fms.log file 6-12
Foreign cell 1-7
making local cell visible to 2-10
making visible in local cell 2-10
Format of CellServDB file (client) 15-3

Fs checkservers command 13-17
Fs cleanacl command 19-22
Fs commands 3-3
  granting privilege for 20-4
  preventing mutual authentication for 8-5
Fs copyacl command
  copying an ACL 19-19
Fs debug command 13-21
Fs examine command 5-38, 5-51
Fs flush command 13-24
Fs flushvolume command 13-25
Fs getcacheparms command 13-6
Fs getcellstatus command 13-15
Fs getserverprefs command 13-27
Fs listacl command 19-9
Fs listcells command 15-6
Fs listquota command 5-37, 5-39, 5-50
Fs lsmount command 5-29
Fs messages command 13-19, 13-20
Fs mkmount command 5-28
  for Backup volume 5-22
  for ReadWrite volume 5-11
  for renamed volume 5-64
  for restored volume 5-62
  when changing user name 17-24
  when creating user account 17-10
Fs monitor command 13-23
Fs newcell command 15-8
Fs process 3-2
  See also File Server, Salvager, Volume Server
Fs quota command 5-50
Fs rmmount command 5-29
  for Backup volume 5-57
  for ReadWrite volume 5-56
  when changing user name 17-24
  when removing user account 17-26
  when renaming volume 5-63
Fs setacl command
  adding entry to Negative rights list 19-15
  changing default set by user account creation 17-12
  complete syntax 19-11
  completely replacing ACL 19-18
  removing entry 19-14
  restoring previously denied rights 19-17
  when granting access 19-13
Fs setcachesize command 13-6
Fs setcell command 13-15
Fs setquota command 5-49
  to change default quota 17-12
Fs setserverprefs command 13-28
Fs setvol command 5-50
Fs whereis command 5-39
Fs-type server process
  defined 4-4
  defining in BosConfig file 4-15
Fsck command (AFS version) 2-4
Fsck command
  AFS compared to UNIX 2-4
Ftpd command
  AFS compared to UNIX 2-3, 2-58
Full dump
  outside Backup System 5-58
Full dumps 6-43
Full restores 6-55
Getting help xxv
Granting
  access on ACL 19-13
  access on ACL after having denied it 19-16
  privilege for backup commands 20-7
  privilege for bos commands 20-7
  privilege for fs commands 20-4
  privilege for kas commands 20-5
  privilege for pts commands 20-4
  privilege for vos commands 20-7
  privilege of all three types 20-10
Group
  adding members 18-25
  AFS UID of 2-36
  changing name of 18-31
  creating Protection Database entry 18-21
  creation-quota (see entry: *Group-creation quota*) 18-32
  defined 1-14
  deleting entry from Protection Database 18-27
  effect of ownership change on name 18-28
  group-owned 18-5
  listing creator 18-10
  listing groups owned 18-12
  listing members 18-11
  listing memberships in, for machine 18-14
  listing memberships in, for user 18-14
  listing number belonged to, for machine entry 18-13
  listing number belonged to, for user 18-13
  listing number of members 18-10
  listing owner 18-10
  listing privacy flags on ProtDB entry 18-36, 18-37
  listing those owned by user 18-15
  making into self-owned 18-24
  mapping name to AFS UID 18-10
  orphaned, listing 18-12
  prefix-less 18-3
  privacy flags on 2-37
  private use 18-5
  regular 18-3
  removing members 18-26
  restrictions on 2-37
  rules for naming 18-21
  self-owned 18-5
  setting privacy flags on ProtDB entry 18-38
  shared use 18-5
  system-defined 2-36, 18-7
  two types of 2-37
  use for prefix-less 18-6
  using on ACL 19-7
  using system-defined on ACL 19-7
  ways to use 18-4
Group use of group 18-5
Group-creation quota
  defined 18-4
  listing 18-32
  setting 18-32
Group-name field
  in regular group name 18-3
  in regular group name, rules for 18-21
Group-owned group 18-5
Grouping
  user home directories under /afs/cell_name/usr 2-33
GroupMax counter in Protection Database 18-34
Groups
  machine membership in 18-7

Groups command
  AFS compared to UNIX 2-3

Handling server encryption key emergency 10-14
Hard link
  AFS restrictions on 2-4
  creating with uss 16-34
  overwriting with uss 16-6
Help
  example command xxv
  online xxv
Help commands xxv
Highlighting in Scout 7-5

I access right (on ACL) 19-4
Identifying
  binary distribution machine 4-11
  database server machine 4-11
  roles taken by file server machine 4-11
  simple file server machine 4-12
  system control machine 4-11
Incremental dump
  outside Backup System 5-58
Incremental dumps 6-43
Inetd command
  AFS compared to UNIX 2-3, 2-58
Initializing
  scout 7-8
  server process 4-14
    See also Starting
Insert access right (on ACL) 19-4
Installing
  disk on file server machine 11-13
  server process & command suite binaries 11-3
  server process binaries into alternate directories 11-3
  server process binaries, about 3-10
  tape drives 6-13
Instance xviii
Intention flag in VLDB entry 5-8
Interactive mode (in Backup System)
  canceling operations 6-30
  entering 6-29
  job ID numbers 6-30
  leaving 6-30
  listing operations 6-30
Interactive mode
  for kas commands 8-5
  job ID numbers 6-29
Internet
  conventions for cell name 2-6
  Network Information Center 2-6

Job ID 6-29
Job ID numbers 6-30
  displaying 6-30
  using to cancel operations 6-30

K access right (on ACL) 19-5
Kas commands 3-5
  granting privilege for 20-5
  preventing mutual authentication for 8-5
Kas create command
  when changing user name 17-23
  when creating user account 17-10
Kas delete command
  when changing user name 17-23
  when removing user account 17-26
Kas examine command
  to check for ADMIN flag 20-5
  to inspect afs key 10-7
Kas interactive command 8-5
Kas setfields command 2-41, 2-42, 2-43, 17-16, 17-18, 20-6
Kas setpassword command 2-42, 2-43, 10-11, 17-20
  when handling key emergency 10-19
Kas unlock command 17-17
Kaserver process 3-4
    See also Authentication Server
Kaserver.DB0 file 3-15
Kaserver.DBSYS1 file 3-15
Kerberos
  use of user names 1-14
Kerberos Authentication 2-46
  support for in AFS 2-46
Kernel list of cells (client)
  changing 15-8
  derivation from CellServDB file 15-2
  examining 15-6
Key version number
  defined 10-3
KeyFile file
  adding server encryption key 10-9
  function of 1-18, 3-13, 10-3
  listing contents of 10-7
  removing server encryption key 10-12
Keyword
  using to look up command name xxv
Klog command 2-44
  when handling key emergency 10-17
Klog.krb command 2-46
Kpasswd command 2-42, 2-43
Kpwvalid program 2-43
Kvno
    See also Key version number

L access right (on ACL) 19-4
Labeling
  tapes 6-26
Learning
  AFS UID from group name 18-10
  AFS UID from machine IP address 18-13
  AFS UID from user name 18-13
  group name from AFS UID 18-10
  machine IP address from AFS UID 18-13
  user name from AFS UID 18-13
  volume ID given directory/file name 5-38
  volume ID given volume name 5-38
  volume ID number given volume location 5-38
  volume location given directory/file name 5-39
  volume location given volume name/ID number 5-39
  volume name given directory/file name 5-37
Length restriction on volume names 5-9
Levels
  of functionality in user account 17-7
Library files in package 14-5
  constructing 14-13
  examples 14-7
Libxstat_cm.a library 7-25
  data collections 7-25
  example command using 7-26

obtaining more information 7-26
routines 7-25
xstat_cm_test example command 7-27
Libxstat_fs.a library 7-25
data collections 7-25
example command using 7-26
obtaining more information 7-26
routines 7-25
xstat_fs_test example command 7-26
"Lightweight" process 2-51
Link
See also Hard link, Symbolic link
List-taking argument xviii
Listing
ACL entries 19-9
ADMIN flag in Authentication Database entry 20-5
CellServDB file (client), contents of 15-6
CellServDB file (server), contents of 9-5
client's kernel list of cells 15-6
creator of Protection Database entry 18-10, 18-13
current data cache size 13-6
data cache size specified in cacheinfo file 13-4
database server machines in CellServDB file (client) 15-6
database server machines in CellServDB file (server) 9-5
disk partition size 5-51
entries in BosConfig file 4-8
group's members 18-11
group-creation quota 18-32
groups machine belongs to 18-14
groups owned by group 18-12
groups owned by user 18-15
groups user belongs to 18-14
information from VLDB 5-31
information from volume header 5-32
KeyFile file, contents of 10-7
listing AFS UID counters 18-34
members of system:administrators group 20-4
owner of Protection Database entry 18-10, 18-13
privacy flags on Protection Database entry 18-36, 18-37
server encryption key from Authentication Database 10-7
server encryption keys in KeyFile file 10-7
server process status 4-7
time for daily binary check and restart 4-25, 11-7
time stamp on binary file 11-11
tokens held by issuer 2-44
users in UserList file 20-7
VLDB entry with volume header 5-36
volume header with VLDB entry 5-36
volume information 5-31
volume quota, percent used 5-50
volume quota, with volume & partition info. 5-51
volume quota, with volume size 5-50
volume size 5-50, 5-51
See also Examining
Ln command
AFS compared to UNIX 2-3
Local cell 1-7
granting foreign users access to 2-11
making foreign cells visible in 2-10
making visible to foreign cells 2-10
Local configuration files (server) 3-14
Local disk
configuring on client, using package 14-1
files required on client machine's 2-27
protecting on file server machine 2-24
Local password file
creating 17-3
creating entry with uss 16-32
creating with uss 16-8
interaction with AFS login 2-40
setting password in 17-3
setting password in, with uss 16-8
when not using AFS login 2-41
Location
setting for client's disk cache directory 13-4
standard for uss template file 16-16
Lock access right (on ACL) 19-5
Locked VLDB entry
listing information from 5-31
unlocking 5-65
Locking
VLDB entry 5-65
Log files
examining 4-28
fms.log 6-12
for replicated databases 3-15
for server processes 3-16
Login command (AFS version) 2-40
interaction with local password file 2-40
requirements when not using 2-41
Login.krb command 2-46
-long flag on vos listvol command 5-34
Lookup access right (on ACL) 19-4
Lost+found directory 2-4

Machine
assigning AFS UID 18-18
creating Protection Database entry 18-18
listing groups belonged to 18-14
listing number of groups belonged to 18-13
mapping IP address to AFS UID 18-13
Machines
membership in groups 18-7
Mainframe computing environment 1-5
Maintaining
CellServDB file (client) 15-3
CellServDB file (server) 9-3
synchrony of VLDB with volume headers 5-7
Majority
defined for Ubik 2-52
Makefile for package 14-14
modifying 14-19
Man command xxvii
Manual pages (UNIX style) xxvii
Maximum volume quota 5-49
See also Volume quota
Members
adding to group 18-25
of group, listing 18-11
of group, listing number of 18-10
removing from group 18-26
Membership
of system-defined groups 18-7
privacy flag on Protection Database entry 18-36
Memory cache
See also Data cache

Memory state of BOS Server 4-5
Message line in Scout 7-5
Mode bits
See also UNIX
Modifying
clients to run package 14-22
package Makefile 14-19
Monitoring
Cache Manager performance 13-21
Cache Manager processes with AFSMonitor 7-1
disk usage with Scout 7-5
File Server processes with AFSMonitor 7-1
File Server processes with Scout 7-1
outages with Scout 7-7
server processes 4-1
Mount point
cellular 5-26
changing when renaming user 17-24
choosing name for user volume's 2-33
creating cellular 5-28
creating multiple per volume 5-9
creating ReadWrite 5-28
creating regular 5-27
creating with uss 16-22
defined 1-8, 5-8
distinguishing different types 5-25
examining contents of 5-29
ReadWrite 5-25
Regular 5-25
removing 5-29, 5-53
removing when removing user account 17-26
Mounting
Backup volume 5-21
disk on file server machine 11-13
foreign volume in local cell 5-26
ReadOnly volume 5-16
ReadWrite volume 5-11
volume (general instructions) 5-24
volume, about 5-8
volumes in the local cell 5-27
Moving
volume 5-40
Mutual authentication
complex, summarized 1-23
failure due to mismatched keys 10-14
logic of complex 1-24
preventing 8-5
purpose of 1-22
server encryption key in 1-22, 1-23, 10-2
session key in 1-23
shared secret in 1-22
short summary 1-22
simple vs. complex 1-22
simple, summarized 1-23
ticket in 1-23
ticket-granter in 1-23
token in 1-23
See also Authorization checking

Name
of Protection Database entry, changing 18-30
See also Cell name, User name, Volume name
Name space
See also AFS global name space
Needs salvage status flag in volume header 5-6
Negative rights on ACL
defined 19-6
Network
reducing traffic on, through caching 1-10
Network
as computing environment 1-5
defined 1-5
encrypted communication in AFS 2-48
partition, defined 2-53
Network Information Center (for Internet) 2-6
Network Time Protocol Daemon
See also NTPD
NEW site flag in VLDB
as indicator of failed replication 5-15
NFS/AFS Translator
about 2-59
NoAuth file 3-14
creating in emergencies 10-16
None grouping of access rights
using with fs setacl command 19-5
Normal rights on ACL
defined 19-6
NotRun status flag in BosConfig file 4-3
changing to Run 4-20
NTPD
basic description 1-16
invoked by runntp process 3-8
when to contact 3-9
Number variables
in uss template file 16-16

Off-line status flag in volume header 5-6
OLD site flag in VLDB
as indicator of failed replication 5-15
OLD version of binary file
See also .OLD (at beginning of Index)
OldFiles directory
as mount point for Backup volume 5-21
On-line status flag in volume header 5-6
Online help xxv
Operation code xviii
Outages
BOS Server's role in reducing 1-12
due to automatic server restart 4-25
due to server process restart 4-23
due to Ubik election 2-53
monitoring with Scout 7-7
Output of bos status command
how to interpret 4-8
Overcrowding of disk partition
effect on users 5-40
moving volumes to reduce 5-40
Overwriting
directories/files/links with uss 16-6
Owned privacy flag on Protection Database entry 18-36
Owner
of Protection Database entry defined 18-3
of Protection Database entry, changing 18-28
of Protection Database entry, listing 18-10, 18-13
of Protection Database entry, rules for assigning 18-22
Owner-name field
in regular group name 18-3
in regular group name, rules for 18-21

Package

compiling prototype files 14-3, 14-21
configuration file lines 14-9
configuration files 14-3
constructing prototype and library files 14-13
directory structure 14-4
example library files 14-7
example prototype files 14-5
library files 14-5
Makefile 14-14
modifying clients to run 14-22
modifying the Makefile 14-19
preparing prototype files 14-3, 14-7
prototype file 14-2
to update client's CellServDB file 15-10
PAG 2-38
creating with pagsh command 2-43
Pagsh command 2-43
Pagsh.krb command 2-46
Parameters
See also Command parameters
Participating in AFS global name space 2-9
Partition
housing AFS volumes 3-17
restoring (Backup System) 6-55
restoring to a new location (Backup System) 6-56
restoring to the same location (Backup System) 6-56
See also Disk partition, Network partition
Password
AFS compared to UNIX 2-3
changing in AFS 2-42
checking quality of 2-43
consequences of multiple failed authentication attempts 2-41
expiration of 2-42
improving security of 17-14
in local password file 2-40
lifetime of 2-42
restricting reuse 2-43
setting in Authentication Database 17-20
setting in local password file 17-3
setting in local password file with uss 16-8
Personal computing environment 1-5
Personal workstation
as typical AFS machine 1-6
Plus sign on argument xviii
Port offset numbers 6-11
displaying 6-39
in /usr/afs/backup/tapeconfig 6-14
Possible variations
on replication 5-16
Prdb.DB0 file 3-15
Prdb.DBSYS1 file 3-15
Preferences 2-29, 13-26
displaying 13-27
setting 13-28
Prefix-less group
suggestions for use 18-6
Preventing
core leaks, with scheduled restarts 4-25
mutual authentication 8-5
Previewing
uss account creation 16-4
uss account deletion 16-4
Privacy flags on Protection Database entry 2-37
listing 18-37
setting 18-38
Private use of group 18-5
Privilege
granting all three types 20-10
granting for backup commands 20-7
granting for bos commands 20-7
granting for fs commands 20-4
granting for kas commands 20-5
granting for pts commands 20-4
granting for vos commands 20-7
revoking all three types 20-11
three types in AFS 20-2
Privilege required
for AFSMonitor 7-14
for Scout 7-2
uss commands 16-7
Privileged commands 8-2
Process
lightweight 2-51
See also AFS server processes, Server process
Process authentication group 2-38
See also PAG
Programs, setuid
See also Setuid programs
Protection
AFS compared to UNIX 2-2
AFS vs UNIX 19-2
implications of directory-level 19-3
in AFS 1-13, 19-2
in Unix 1-13
See also Authorization, Security
Protection Database 1-14, 18-2
changing group name 18-31
changing owner of entry 18-28
changing user name 17-22
creating machine entry with pts 18-18
creating multiple group entries 18-23
creating prefix-less group entry 18-23
creating regular group entry 18-22
creating user entry 17-8
creating user entry with pts 18-16
creating user entry with uss 16-39
creator of entry defined 18-3
deleting entry 17-27, 18-27
deleting user entry with uss 16-45
examining group entry 18-10, 18-13
group entry explained 18-2
group-creation quota defined 18-4
information in entry 18-3
listing counters for AFS UIDs 18-34
listing entry creator 18-10, 18-13
listing entry owner 18-10, 18-13
listing group-creation quota 18-32
listing privacy flags 18-10, 18-13, 18-36
listing privacy flags on 18-37
machine entry explained 18-2
mapping of name to AFS UID 18-2
membership count defined 18-3
owner of entry defined 18-3
setting counters for AFS UIDs 18-35
setting group-creation quota 18-32
setting privacy flags on 18-38
user entry explained 18-2
Protection group
See also Group

Protection Server
  about starting and stopping 4-5
  as ptserver process 3-5
  basic description 1-13
  building CPS 18-8
  on database server machine 1-21
  restarting after changes to CellServDB file (server) 9-6, 9-8
  when to contact 3-6
Prototype files in package
  about 14-2
  constructing 14-13
  examples 14-5
  preparing 14-3
Pts adduser command 18-25
  to add members to system:administrators 20-4
  when making group self-owned 18-24
Pts chown command 18-29
  to make group self-owned 18-24
Pts commands 3-6
  granting privilege for 20-4
  preventing mutual authentication for 8-6
Pts createuser command
  for multiple machines 18-19
  for multiple users 18-17
  for single user 18-16, 18-19
  when creating user account 17-9
Pts delete command 18-27
  when removing user account 17-27
Pts examine command
  for group 18-10
  for listing group-creation quota 18-32
  for machine 18-13
  for user 18-13
  listing privacy flags 18-37
Pts listmax command 18-34
Pts listowned command
  for group 18-12
  for user 18-15
Pts membership command
  for group 18-11
  for user 18-14
  to list members of system:administrators 20-4
Pts removeuser command 18-26
  to remove member from system:administrators 20-4
Pts rename command 17-22, 18-31
Pts setfields command
  setting group-creation quota 18-32
  setting privacy flags 18-38
Pts setmax command 18-35
Ptserver process 3-5
    See also Protection Server

Quota
    See also Group-creation quota, Volume quota

R access right (on ACL) 19-5
Rc.afsd.large script 13-12
  using 13-13
Rc.afsd.med script 13-12
  using 13-13
Rc.afsd.small script 13-12
  using 13-13
Rcp command
  AFS compared to UNIX 2-3, 2-58
Read access right (on ACL) 19-5
Read grouping of access rights
  using with fs setacl command 19-5
ReadOnly ID number in volume header 5-6
ReadOnly "path" 5-24
ReadOnly volume 1-9
  changing name of 5-63
  choosing site for 5-17
  creating, about 5-14
  creating, how to 5-17
  defined 5-3
  defining site for in VLDB 5-18
  dumping 5-58
  ID number in VLDB 5-5
  mounting 5-16
  moving 5-40
  need for all-or-nothing release 5-14
  releasing 5-18
  removing, effect of 5-53
  removing, how to 5-56
  selecting site for 2-20
ReadWrite ID number in volume header 5-6
ReadWrite mount point 5-25
  distinguishing from regular 5-25
ReadWrite "path" 5-24, 5-25
ReadWrite volume 1-9
  as source for clones 5-3
  changing name of 5-63
  cloning for Backup version 5-20
  cloning for replication 5-14
  creating 5-10
  dumping 5-58
  ID number in VLDB 5-5
  mounting 5-11
  moving 5-40
  removing, effect of 5-53
  removing, how to 5-55
  types suitable for replication 5-14
Rebooting
  file server machine, instructions 4-30
  file server machine, limiting 2-26
Recycling
  useCounts of tapes (Backup System) 6-8
Regular mount point 5-25
  distinguishing from ReadWrite 5-25
Release
  flags in VLDB 5-5
Release stage in replication 5-14
ReleaseClone 5-15
ReleaseClone number in volume header 5-6
Releasing
  ReadOnly volume 5-18
  ReadOnly volume, forcing new cloning 5-15
  ReadOnly volume, need for all-or-nothing 5-14
Remote services
  modifications for AFS 2-58
Remove privacy flag on Protection Database entry 18-36
Removing
  ADMIN flag from Authentication Database entry 20-6
  core files from /usr/afs/logs 11-12
  database server machine from CellServDB file (client) 15-10
  database server machine from CellServDB file (server) 9-8
  database server machine from client's kernel list 15-8

disk from file server machine 11-16
entry from ACL 19-14
members from groups 18-26
members from system:administrators group 20-4
mount point 5-29, 5-53
mount point when changing user name 17-24
mount point when removing user account 17-26
obsolete .BAK and .OLD version of binaries 11-12
obsolete AFS UIDs from ACL 19-21
Protection Database entry 17-27, 18-27
server encryption key from KeyFile file 10-12
server process from BosConfig file 4-18
user account components 17-25
user account, about 2-35
users in UserList file 20-8
volume 5-53
volume when removing user account 17-26
See also Deleting
Renaming
user account components 17-22
volume 5-63
volume when changing user name 17-23
Replacing
all entries on ACL 19-18, 19-19
Replicated database files 3-15
Replication
appropriate situation for 2-19
defined 1-9, 5-4
detailed discussion 5-14
determining success of 5-14
forcing creation of new clone 5-15
need for all-or-nothing release 5-14
release stage 5-14
role of ReleaseClone in 5-15
site definition stage 5-14
types of volumes suitable for 5-14
variations possible in 5-16
Requirements
basic configuration of client machine 12-1
for Scout 7-2
See also Privilege
Resetting
disk cache size to default value 13-6
Resizing
Scout display 7-7
Restart times for BOS Server
about 2-25, 4-25
listing 4-25, 11-7
setting 4-26, 4-27, 11-8
Restarting
server process to use new binaries immediately 11-5
server process when changing authorization checking 8-4
server process, selected 4-24
server process, except BOS Server 4-24
server process, including BOS Server 4-23
Restoring
access on ACL after having denied it 19-16
Backup Database from tape 6-59
data 6-55
data that no longer exists 6-55
full restores 6-55
incremental restores 6-55
overwriting existing data 6-55
preserving existing data 6-55
synchrony of VLDB and volume headers 5-42
volumes, not using AFS Backup System 5-58
Restrictions
on hard links in AFS 2-4
on volume names 2-16, 5-9
Reverse video
use in Scout 7-5
Reverting
to old version of server process & command binaries 11-9
Revoking
privilege of all three types 20-11
Rlogind command
AFS compared to UNIX 2-3, 2-58
Roles for file server machine
determining 4-11
summarized 1-17
Root directory 1-9, 5-8
Root user
limiting logins as 2-50
Root.afs volume 2-16
Root.cell volume 2-16
Rsh command
AFS compared to UNIX 2-3, 2-58
Rules
for uss bulk input file 16-48
for uss template file 16-17
group names 18-21
Run status flag in BosConfig file 4-3
changing to NotRun 4-18, 4-19
Runntp process 3-8
See also NTPD

Salvage.fs file 3-14
Salvage.lock file 3-14
SalvageLog file 3-16
examining 4-28
Salvager
as part of fs process 3-2, 4-4
basic description 1-16
examining log file 4-28
how to run 5-46
running before VLDB/volume header resynchronization 5-42
when to contact 3-3
Salvaging
all volumes on file server machine 5-47
all volumes on partition 5-47
single volume 5-48
Saving
previous version of binaries 11-3
Scheduling
creation of Backup volumes 5-21
Scout 7-1
attention levels, setting 7-5
banner line 7-3
basename 7-3
command syntax 7-8
disk usage 7-5
display, resizing 7-7
examples 7-9
features summarized 7-2
highlighting in 7-5
message/probe line 7-5
outages, monitoring 7-7

requirements 7-2
resizing display 7-7
reverse video in 7-5
screen examples 7-9
screen layout 7-3
setting terminal type 7-2
starting 7-8
statistics available 7-4
stopping 7-9
Screen layout in Scout 7-3
Scripts
Cache Manager configuration 13-12
rc.afsd.large 13-12
rc.afsd.med 13-12
rc.afsd.small 13-12
Secondary site
in Ubik 2-52
Security
AFS features 2-47
encrypted network communication 2-48
suggestions for improving 2-49
See also Mutual authentication
Self-owned group 18-5
Server encryption key 1-23
adding to KeyFile file 10-9
changing frequently 10-5
checksum as stand-in for 10-6
defined 1-22, 10-2
emergency need to replace 10-14
in Authentication Database 10-4
in KeyFile file 10-3
listing from Authentication Database 10-7
listing from KeyFile file 10-7
password-like nature of 10-3
removing from KeyFile file 10-12
role in mutual authentication 10-2
sites for 10-3
Server portion
See also Update Server
Server process
binaries (see entry: *Server process binaries* 11-2
creating cron type 4-16
creating fs type 4-15
creating simple type 4-14
creating ticket (tokens) for 2-44
creating, about 4-14
cron type, defined 4-5
daily check and restart for new binaries 11-7
defined 1-6
defining cron type in BosConfig file 4-16
defining fs type in BosConfig file 4-15
defining simple type in BosConfig file 4-14
different names for 3-2
examining log files 4-28
fs type, defined 4-4
list of AFS's 1-11
listing entry in BosConfig 4-8
listing status 4-7
need to know about database server processes 9-2
removing from BosConfig file, about 4-14
removing from BosConfig file, how to 4-18
restarting specific 4-24
restarting for changed binaries 11-3, 11-9
restarting immediately 4-23
restarting to use new binaries immediately 11-5
restarting, except BOS Server 4-24
restarting, including BOS Server 4-23
simple type, defined 4-4
starting up, about 4-14, 4-19
stopping permanently, about 4-14, 4-19
stopping permanently, how to 4-18, 4-19
See also AFS server processes
Server process binaries
about installing 3-10
checking time stamp 11-11
in /usr/afs/bin 3-10
installing 11-3
removing obsolete 11-12
reverting to old version 11-9
uninstalling 11-9
Server ticket
See also Ticket
Server, defined 1-6
Server/client model 1-6
Session key 1-23
Setting
ACL for directory with uss 16-27
ACL on home directory with uss 16-24
AFS UID counters 18-35
BOS Server's weekly restart time 4-26
Cache Manager preferences for file server machines 13-26
cell name 2-7
client-to-file-server probe interval 13-17
disk cache location in cacheinfo file 13-4
entries on ACL, about 19-11
group-creation quota 18-32
home cell for client machine 13-18
password in Authentication Database 17-20
privacy flags on Protection Database entry 18-38
terminal type for Scout 7-2
ThisCell file (client), value in 13-18
time for daily binary check and restart 4-27, 11-8
volume quota on multiple volumes 5-50
volume quota on single volume 5-49
volume quota with uss 16-21
Setuid programs
controlling running on client machine 13-15
restrictions on 2-5
Shared secret 1-22
Shared use of group 18-5
Shorthand groupings
using with fs setacl command 19-5
Simple file server machine 1-17
identifying with bos status 4-12
Simple mutual authentication
See also Mutual authentication
Simple-type server process
defined 4-4
defining in BosConfig file 4-14
Site
count in VLDB 5-5
for volume, defined 5-3, 5-5
Site definition stage in replication 5-14
Size
See also Data cache, size
Size quota
See also Volume quota
Slowed performance
preventing in AFS 1-10

"Source" volume 1-9
Stages in volume replication 5-14
Starting
  database server process, about 4-5
  scout 7-8
  server process 4-14, 4-19
    See also Initializing
Statistics available in Scout 7-4
Status
  listing for server process 4-7
  privacy flag on Protection Database entry 18-36
Status flag in BosConfig file
  changing NotRun to Run 4-20
  changing Run to NotRun 4-18, 4-19
  NotRun, meaning of 4-3
  Run, meaning of 4-3
Status flag in volume header 5-6
Stopping
  database server process, about 4-5
  scout 7-9
  server process and immed. restarting 4-23
  server process, permanently 4-18, 4-19
Storing statistic in Scout 7-4
Suitability of volumes for replication 5-14
Switch
  defined xviii
  omitting xx
Symbolic link
  at second (cell) level, to save typing 2-13
  creating with uss 16-34
  overwriting with uss 16-6
Symptoms
  of corruption 5-45
  of VLDB/volume header desynchronization 5-42
Synchronization site
  in Ubik 2-52
Synchrony of VLDB and volume headers
  importance of 5-42
  maintained by VL and Volume Servers 5-7
  restoring 5-42
  symptoms of lack of 5-42
@sys variable in pathnames 2-29
System administrator privilege
    See also Privilege
System control machine 1-18
  as distributor of UserList file 20-7
  as source for KeyFile file 10-4
  as source for server's CellServDB file 9-3
  identifying with bos status 4-11
System groups
  defined 18-7
  on ACLs 18-8
  using on ACLs 19-7
System outages
  due to automatic server restart 4-25
  due to server process restart 4-23
  due to Ubik election 2-53
System outages, reducing 1-12
System type
  Transarc's standard names 2-22
System:administrators group
  about 2-36
  adding members to 20-4
  listing members of 20-4
  membership defined 18-8
  privileges of 18-8, 20-2
  removing members from 20-4
System:anyuser group
  about 2-36
  membership defined 18-8
System:authuser group
  about 2-36
  membership defined 18-8

Tape
  determining EOF marker size for backup 6-12
  determining tape size for backup 6-12
Tape archiving 6-8
Tape Coordinator machines 6-11, 6-13
  assigning file ownership 6-15
  using IBM RS/6000 machines 6-16
Tape Coordinators 6-17, 6-32
  adding to Backup Database 6-16
  checking the status of 6-33
  displaying port offset numbers 6-39
  entry in /usr/afs/backup/tapeconfig 6-14
  port offset numbers 6-11
  starting 6-32
  stopping 6-33
Tape drives
  installing 6-13
Tape labels 6-10, 6-25
  useCounts of tapes 6-8
Tape names 6-6
Tape recycling schedules 6-7
Tape size
  in /usr/afs/backup/tapeconfig 6-14
Task ID 6-29
Terminal type
  setting for AFSMonitor 7-14
  setting for Scout 7-2
ThisCell file (client) 12-4
  how used by programs 2-8
  setting value in 13-18
ThisCell file (server) 1-18, 3-13
Ticket 1-23
  creating for server process 2-44
Ticket-granter 1-23
Tickets
  discarding 2-45
  listing for user 2-44
Time stamp
  on binary file, listing 11-11
Tokens
  command 2-44
  creating for server process 2-44
  data in 1-23
  discarding 2-45
  listing for user 2-44
  one-per-cell rule 2-44
  setting default lifetimes for users 2-45
Tokens.krb command 2-46
Translating
  AFS UID to group name 18-10
  AFS UID to machine IP address 18-13
  AFS UID to user name 18-13
  directory/file name to volume ID number 5-38
  directory/file name to volume location 5-39
  directory/file name to volume name 5-37
  group name to AFS UID 18-10

machine IP address to AFS UID 18-13
user name to AFS UID 18-13
volume location to volume ID number 5-38
volume name to volume ID number 5-38
volume name/ID number to volume location 5-39
Transparent access 1-7, 5-4
Turning off authorization checking 8-4
Turning on authorization checking 8-4
Type
flags in VLDB 5-5

Ubik 2-51
automatic updates 2-52
consistency guarantees 2-52
coordinator 2-52
election of coordinator 2-53
failure due to mismatched server encryption keys 10-14
features summarized 2-51
majority defined for 2-52
need for all database servers running 2-56
need for correct *CellServDB* 2-56
need for synchronized clocks 2-54
replicated database files 3-15
requirements summarized 2-54
secondary site 2-52
server and client portions of 2-51
synchronization site 2-52
use of CellServDB file 9-2
UID
See also AFS UID, UNIX UID
Umount command 11-17
Undefined access rights 19-6
Uninstalling
server process & command suite binaries 11-9
server process binaries in alternate directories 11-9
UNIX
differences from AFS, summarized 2-2
manual pages for AFS command suites xxvii
mode bits, interpretation in AFS 19-23
UNIX UID
functional difference from AFS UID 1-14, 18-3
matching AFS UID 2-32, 16-8
Unlocking
VLDB entry 5-65
Unlog command 2-45
when handling key emergency 10-16
Unmounting
disk from file server machine 11-16
volume 5-29, 5-53
Upclient process 3-7
See also Update Server
Update date
recorded in volume header 5-7
Update Server
about starting and stopping 4-6
as distributor of binaries 3-10, 11-2
as distributor of CellServDB file (server) 9-3
as distributor of configuration files 3-12
as distributor of KeyFile file 10-4
as upserver and upclient processes 3-7
basic description 1-15
client portion 1-15
client portion for binaries 1-19
client portion for configuration files 1-19
server portion 1-15
server portion on binary distribution machine 1-19
server portion on system control machine 1-19
when to contact 3-7
Updating
CellServDB file (client) with package 15-10
CellServDB file (client) without package 15-11
Upserver process 3-7
See also Update Server
UseCounts of tapes (Backup System) 6-8
User
account (see entry: *User account*) 16-2, 17-2
adding to group 18-25
assigning AFS UID 18-16
assigning name in pts 18-16
creating account for, with uss 16-39
creating account for, without uss 17-8
creating Protection Database entry 18-16
deleting account for, with uss 16-45
deleting entry from Protection Database 18-27
listing groups belonged to 18-14
listing groups owned 18-15
listing number of groups belonged to 18-13
listing privacy flags on ProtDB entry 18-36, 18-37
mapping name to AFS UID 18-13
name (see entry: *User name*) 16-2, 17-2
removing from group 18-26
setting privacy flags on ProtDB entry 18-38
User account
advice on creating 2-30
components of 16-2, 17-2
converting existing UNIX to AFS 17-5
converting existing UNIX to AFS 16-10
creating many at once 16-48
creating standard files in 2-34
creating with uss 16-39
creating without uss 17-8
deleting many at once 16-48
deleting with uss 16-45
different types of accounts 16-12, 17-7
issues to consider when deleting 16-45
matching AFS and UNIX UIDs 17-3
matching AFS and UNIX UIDs 16-8
methods for grouping 16-22
previewing uss creation of 16-4
previewing uss deletion of 16-4
removing from system 17-25
removing, about 2-35
suggestions for grouping home directories 2-33
two methods for creating and deleting 16-3
User name
as part of volume name 2-32
assigning in uss 16-39
assigning without uss 17-8
changing 17-22
choosing, advice on 2-31
use by Kerberos 1-14
UserList file 1-18, 3-13
adding users to 20-7
listing users in 20-7
privileges of membership in 20-2
removing users from 20-8
UserMax counter in Protection Database 18-34
Uss
advantages over manual account creation 16-3
assigning AFS UID 16-8

avoiding account creation failure 16-4
avoiding account deletion failure 16-4
converting existing UNIX accounts 16-10
creating directory 16-25
creating entry in /etc/passwd 16-32
creating file by echoing 16-30
creating file from prototype 16-28
creating hard link 16-34
creating symbolic link 16-34
creating volume 16-19
defining directory for even distribution 16-24
different types of accounts 16-12
executing command from template 16-37
mounting volume 16-22
overwriting behavior 16-6
previewing effect of command 16-4
privilege required 16-7
recovering from account creation failure 16-5
reissuing command, effect of 16-5
setting ACL for directory 16-27
setting ACL on home directory 16-24
setting volume quota 16-21
specifying password/authentication security 16-36
the types of accounts that can be created 16-3
Uss add command 16-41
Uss bulk command 16-52
Uss bulk input file
rules for constructing 16-48
Uss delete command 16-46
Uss template file
A line in 16-36
advantages of 16-3
constants in 16-14
creating directory 16-25
creating file in 16-28, 16-30
creating hard link in 16-34
creating mount point 16-22
creating symbolic link in 16-34
creating volume 16-19
D line 16-25
defining directory for even distribution in 16-24
E line in 16-30
examples 16-17
executing command from 16-37
F line in 16-28
G line in 16-24
L line in 16-34
number variables in 16-16
password/authentication security 16-36
rules for constructing 16-17
S line in 16-34
setting ACL for directory 16-27
setting ACL on home directory 16-24
setting volume quota 16-21
standard locations for 16-16
summary of lines 16-12
V line in 16-19
variables in 16-14
X line in 16-37
zero length 16-19

V file in data cache 12-5
Variables
@sys in pathnames 2-29
in uss template file 16-14
Variations possible
in replication 5-16
VL Server
about starting and stopping 4-5
as vlserver process 3-6
basic description 1-14
importance to transparent access 1-14
on database server machine 1-21
restarting after changes to CellServDB file (server) 9-6, 9-8
role in VLDB/volume header synchronization 5-8, 5-42
when to contact 3-6
VLDB 1-14
defining ReadOnly site in 5-18
examining entry, incl. locked one 5-31
examining entry, with volume header 5-36
information in 5-5
intention flag set in by VL Server 5-8
locking entry 5-65
release flags in 5-5
site count in 5-5
synchronizing with volume headers 5-7, 5-42
type flags in 5-5
unlocking entry 5-65
Vldb.DB0 file 3-15
Vldb.DBSYS1 file 3-15
Vlserver process 3-6
See also VL Server
Vnode index 5-13
VolserLog file 3-16
examining 4-28
Volume
as unit of backup 1-8, 5-4
as unit of replication 1-8, 5-4
automating creation of Backup version 5-21
backing up (Backup System) 6-43
Backup (see entry: *Backup volume*) 5-20
benefits of for system efficiency 5-4
correspondence with directory 1-8
creating Backup version of many at once 5-20
creating with uss 16-19
creation date in volume header 5-6
defined 1-8, 5-3
displaying dump history (Backup System) 6-40
dumping without AFS Backup System 5-58
flushing from data cache on client machine 13-24
grouping related on same partition 2-18
header (see entry: *Volume header*) 5-6
ID number (see entry: *Volume ID number*) 5-5
in load balancing 1-8, 5-4
information about in VLDB 5-5
listing information about 5-31
location (see entry: *Volume location*) 5-39
mounting 1-9
mounting more than once 5-9
mounting, about 5-8
moving 5-40
name (see entry: *Volume name*) 5-9
number of accesses in day 5-7
overwriting with restore (Backup System) 6-55
preserving with restore (Backup System) 6-55
quota (see entry: *Volume quota*) 5-49
ReadOnly (see entry: *ReadOnly volume*) 5-14
ReadOnly, defined 1-9
ReadWrite (see entry: *ReadWrite volume*) 5-10

ReadWrite, defined 1-9
removing 5-53
removing when removing user account 17-26
removing, alternate commands for 5-54
renaming 5-63
replicating 5-14
replication of 1-9
restoring (Backup System) 6-55
restoring data that no longer exists (Backup System) 6-55
restoring without Backup System 5-58
root directory of 1-9
root.afs 2-16
root.cell 2-16
salvaging 5-45
separate for each top level directory 2-16
size, listing 5-50
synchronizing VLDB and volume header for 5-42
unit of resource management 1-8
update date in volume header 5-7
volume ID number (see entry: *Volume ID number*) 5-5
when replication is appropriate 2-19
where to place replicated 2-20
Volume entries 6-3, 6-18
adding to Backup Database 6-21
deleting from Backup Database 6-22
listing 6-36
suggestions 6-19
Volume header 5-6
examining 5-32
examining, with VLDB entry 5-36
in /vicepx directories 3-17
synchronizing with VLDB 5-42
synchronizing with with VLDB 5-7
Volume ID number
defined 5-5
learning from directory/file name 5-38
learning from volume location 5-38
learning from volume name 5-38
translating to volume location 5-39
Volume location
learning from directory/file name 5-39
learning from volume name/ID number 5-39
translating to volume ID number 5-38
Volume Location Database
See also VLDB
Volume Location Server
See also VL Server
Volume name
changing 5-63
changing when renaming user 17-23
conventions for 2-16, 5-9
learning from directory/file name 5-37
restrictions on 2-16, 5-9
suggested scheme for 2-17
translating to volume ID number 5-38
translating to volume location 5-39
two required names 2-16
Volume quota
as recorded in volume header 5-6
changing default assigned when creating user account 17-12
default set by vos create 5-10
listing percent used 5-50
listing with volume & partition info. 5-51
listing with volume size 5-50
setting on multiple volumes 5-50
setting on single volume 5-49
setting with uss 16-21
Volume Server
as part of fs process 3-2, 4-4
basic description 1-14
examining log file 4-28
role in VLDB/volume header synchronization 5-8, 5-42
when to contact 3-3
Volume sets 6-3, 6-18
adding to Backup Database 6-20
deleting from Backup Database 6-21
listing 6-36
VolumeItems file 12-5
Vos addsite command 5-18
Vos backup command 5-22
Vos backupsys command 5-23
Vos commands 3-3, 3-6
granting privilege for 20-7
preventing mutual authentication for 8-6
Vos create command 5-11
when creating user account 17-10
Vos delentry command 5-55
Vos dump command 5-59
Vos examine command 5-37
to learn volume ID 5-38
Vos listpart command 11-14
Vos listvldb command 5-31
to learn volume location 5-39
Vos listvol command 5-33
output with -extended flag 5-34
output with -fast flag 5-33
output with -long flag 5-34
to learn volume ID 5-38
Vos lock command 5-66
Vos move command 5-41
when removing file server machine disk 11-17
Vos partinfo command
when creating ReadOnly volume 5-17
when creating ReadWrite volume 5-10
when restoring volume 5-61
Vos release command 5-18
forcing new cloning with -f flag 5-15
Vos remove command
for Backup volume 5-57
for ReadOnly volume 5-56
for ReadWrite volume 5-55
when removing user account 17-26
Vos remsite command 5-54
Vos rename command 5-63
when changing user name 17-23
Vos restore command
to create new volume 5-62
to overwrite volume 5-60
Vos syncserv command
effects of 5-43
syntax for 5-44
Vos syncvldb command
effects of 5-43
syntax for 5-44
Vos unlock command 5-66
Vos unlockvldb command 5-66
Vos zap command 5-54

W access right (on ACL) 19-5
-wait flag on bos shutdown command 4-21
Weekly restart of BOS Server
  automatic, about 2-25, 4-25
  listing time for 4-25
  setting time for 4-26
Window
  resizing in Scout 7-7
Workstations statistic in Scout 7-4
Write access right (on ACL) 19-5
Write grouping of access rights
  using with fs setacl command 19-5

Xstat 7-25
  a requirement for running AFSMonitor 7-14
  data collections 7-25
  example commands 7-26
  libxstat_cm.a library 7-25
  libxstat_fs.a library 7-25
  obtaining more information 7-26
  xstat_cm_test example command 7-27
  xstat_fs_test example command 7-26
Xstat_cm_test command 7-27
Xstat_fs_test command 7-26

Table of Contents

About This Guide . . . xiii

Audience and Purpose . . . xiv
Organization of the Document . . . xiv
How to Use This Document . . . xv
Related Documents . . . xvi
Document Conventions . . . xvii
Command Syntax . . . xviii
Rules for Typing AFS Commands . . . xx
Rules for Using Abbreviations and Aliases . . . xxii
Getting Online Help on AFS Commands . . . xxv
Ordering, Support, and Training Information . . . xxviii

Concepts and Issues

1. An Overview of AFS for the System Administrator . . . 1-1

1.1. A Broad Overview of AFS . . . 1-2
1.2. More Detailed Discussions of Some Basic Concepts . . . 1-5
1.2.1. Networks . . . 1-5
1.2.2. Distributed File Systems . . . 1-5
1.2.3. Servers and Clients . . . 1-6
1.2.4. Cells . . . 1-7
1.2.5. The Uniform Name Space and Transparent Access . . . 1-7
1.2.6. Volumes . . . 1-8
1.2.7. Mount Points . . . 1-8
1.2.8. Replication . . . 1-9
1.2.9. Caching and Callbacks . . . 1-9
1.3. AFS Server Processes and the Cache Manager . . . 1-11
1.3.1. The File Server . . . 1-12
1.3.2. The Basic OverSeer Server . . . 1-12
1.3.3. The Authentication Server . . . 1-13
1.3.4. The Protection Server . . . 1-13
1.3.5. The Volume Server . . . 1-14
1.3.6. The Volume Location (VL) Server . . . 1-14
1.3.7. The Update Server . . . 1-15
1.3.8. The Backup Server . . . 1-15
1.3.9. The Salvager . . . 1-16
1.3.10. The Network Time Protocol Daemon . . . 1-16
1.3.11. The Cache Manager . . . 1-16
1.4. The Four "Roles" for File Server Machines . . . 1-17
1.4.1. Simple File Server Machines . . . 1-17
1.4.2. The System Control Machine . . . 1-18
1.4.3. Binary Distribution Machines . . . 1-19
1.4.4. Database Server Machines . . . 1-20
1.5. AFS Security: A More Detailed Look at Mutual Authentication . . . 1-22
1.5.1. Simple and Complex Mutual Authentication . . . 1-22

2. Issues in Cell Set-Up and Administration . . . 2-1

2.1. Differences between AFS and UNIX: A Summary . . . 2-2
2.1.1. Differences in File and Directory Protection . . . 2-2
2.1.2. Differences in Authentication . . . 2-3
2.1.3. Differences in the Semantics of Standard UNIX Commands . . . 2-3
2.1.4. The AFS version of the fsck Command . . . 2-4
2.1.5. Creating Hard Links . . . 2-4
2.1.6. AFS Implements "Save on Close" . . . 2-4
2.1.7. Setuid Programs . . . 2-5
2.2. Choosing a Cell Name . . . 2-6
2.2.1. How to Choose a Cell Name . . . 2-6
2.2.2. How to Set the Cell Name . . . 2-7
2.2.3. Why Choosing the Appropriate Cell Name is Important . . . 2-8
2.3. Participating in the AFS Global Name Space . . . 2-9
2.3.1. What the Global Name Space Looks Like . . . 2-9
2.3.2. Making Your Cell Visible to Others . . . 2-10
2.3.3. Making Other Cells Visible in Your Cell . . . 2-10
2.3.4. Granting and Denying Foreign Users Access to Your Cell . . . 2-11
2.4. Setting Up Your File Tree . . . 2-13
2.4.1. The Top ***/afs*** Level . . . 2-13
2.4.2. The Second "Cell" Level . . . 2-13
2.4.3. The Third Level . . . 2-14
2.5. Setting Up Your Volumes . . . 2-16
2.5.1. Creating Volumes that Correspond to Your File Tree . . . 2-16
2.5.2. Restrictions on Volume Names . . . 2-16
2.5.3. Two Required Volume Names . . . 2-16
2.5.4. Naming Volumes to Make Administration Easier . . . 2-17
2.5.5. Using Prefixes on Related Volumes . . . 2-17
2.5.6. Grouping Related Volumes on a Partition . . . 2-18
2.5.7. When to Replicate Volumes . . . 2-19
2.5.8. Smaller Volumes are More Efficient for Load Balancing . . . 2-20
2.5.9. Space Quotas on Volumes . . . 2-21
2.5.10. The Default Quota and ACL on a New Volume . . . 2-21
2.6. Supported System Types . . . 2-22
2.7. Setting Up File Server Machines . . . 2-23
2.7.1. The Supported System Types . . . 2-23
2.7.2. The Four Roles for a File Server Machine . . . 2-23
2.7.3. Your Cell’s First File Server Machine . . . 2-23
2.7.4. Installing Additional File Server Machines . . . 2-24
2.7.5. Running Three Database Server Machines . . . 2-24
2.7.6. Protecting Files on A File Server Machine’s Local Disk . . . 2-24
2.7.7. Keeping All Binaries Available . . . 2-25
2.7.8. Configuring Partitions to Store AFS Data . . . 2-25
2.7.9. Automatic Process Restarts . . . 2-25
2.7.10. Limiting Reboots . . . 2-26
2.7.11. Monitoring File Server and Cache Manager Processes . . . 2-26
2.8. Setting Up Client Machines . . . 2-27
2.8.1. The Supported System Types . . . 2-27
2.8.2. Configuring the Local Disk . . . 2-27
2.8.3. Using the package Program to Configure Local Disks . . . 2-28
2.8.4. Client Access to Foreign Cells Can Vary . . . 2-28

2.8.5. Using the ***@sys*** Variable in Pathnames 2-29
2.8.6. Setting a Cache Manager’s Preferences for File Server Machines 2-29
2.9. Setting Up AFS User Accounts 2-30
2.9.1. The Two Methods for Creating AFS User Accounts 2-30
2.9.2. The Components of an AFS User Account 2-30
2.9.3. Levels of Account Functionality 2-30
2.9.4. Choosing User Names and Naming Other Account Components 2-31
2.9.5. Converting Unix Accounts into AFS Accounts 2-33
2.9.6. Grouping Home Directories 2-33
2.9.7. Making a Backup Version of User Volumes Available 2-34
2.9.8. Creating Standard Files in New AFS Accounts 2-34
2.9.9. Removing User Accounts 2-35
2.10. Using AFS Protection Groups 2-36
2.10.1. Group AFS UIDs 2-36
2.10.2. The Three System Groups 2-36
2.10.3. The Two Types of User-Defined Groups 2-37
2.10.4. Rules Concerning Groups 2-37
2.11. Login and Authentication in AFS 2-38
2.11.1. Using PAGs to Identify AFS Tokens 2-38
2.11.2. Using the AFS login Program 2-40
2.11.3. Not Using the AFS login Program 2-41
2.11.4. Limiting the Number of Failed Authentication Attempts 2-41
2.11.5. Changing Passwords 2-42
2.11.6. Using pagsh to Obtain a New PAG 2-43
2.11.7. The One-Token-Per-Cell-Per-Machine-Per-PAG Rule 2-44
2.11.8. Using klog to Obtain a New Token 2-44
2.11.9. Using tokens to Examine Tokens 2-44
2.11.10. Using unlog to Discard Tokens 2-45
2.11.11. Setting Default Token Lifetimes for Users 2-45
2.11.12. Support for Kerberos Authentication 2-46
2.12. Security and Authorization in AFS 2-47
2.12.1. Some of AFS’s Important Security Features 2-47
2.12.2. Three Types of Privilege 2-48
2.12.3. Authorization Checking versus Authentication 2-49
2.12.4. General Suggestions for Improving Security in Your Cell 2-49
2.13. AFS’s Distributed Database Technology 2-51
2.13.1. Ubik’s Structure and Features 2-51
2.13.2. What Ubik Needs from the System Administrator 2-54
2.14. Backing Up the System 2-57
2.14.1. Backup Volumes 2-57
2.14.2. The AFS Backup System 2-57
2.15. Using Standard Remote Services in the AFS Environment 2-58
2.16. Accessing AFS through NFS 2-59

Managing File Server Machines

3. File Server Machine Processes and Files 3-1

3.1. Brief Descriptions of the AFS Server Processes 3-2
3.1.1. The ***fs*** Collection of Processes: the File Server, Volume Server and Salvager 3-2
3.1.2. The ***bosserver*** Process: the Basic OverSeer Server 3-4

3.1.3. The ***kaserver*** Process: the Authentication Server . . . 3-4
3.1.4. The ***ptserver*** Process: the Protection Server . . . 3-5
3.1.5. The ***vlserver*** Process: the Volume Location Server . . . 3-6
3.1.6. The ***upserver*** and ***upclient*** Processes: the Update Server . . . 3-7
3.1.7. The ***buserver*** Process: the Backup Server . . . 3-8
3.1.8. The ***runntp*** Process: the Network Time Protocol Daemon . . . 3-8
3.2. Binaries in */usr/afs/bin* . . . 3-10
3.3. Common Configuration Files in */usr/afs/etc* . . . 3-12
3.4. Local Configuration Files in */usr/afs/local* . . . 3-14
3.5. Replicated Database Files in */usr/afs/db* . . . 3-15
3.6. Log Files in */usr/afs/logs* . . . 3-16
3.7. Volume Headers on the */vicepx* Partitions . . . 3-17

4. Monitoring and Controlling Server Processes . . . 4-1

4.1. Summary of Commands . . . 4-1
4.2. Controlling and Checking Process Status . . . 4-3
4.2.1. The Information in ***BosConfig*** . . . 4-3
4.2.2. The Types of Processes . . . 4-3
4.2.3. How the BOS Server Uses the Information in ***BosConfig*** . . . 4-5
4.2.4. About Starting and Stopping the Database Server Processes . . . 4-5
4.2.5. About Starting and Stopping the Update Server . . . 4-6
4.3. Listing Process Status and Information from *BosConfig* . . . 4-7
4.3.1. Possible Process Statuses as Reported by **bos status** . . . 4-8
4.3.2. Information from ***BosConfig*** as reported by **bos status** . . . 4-9
4.4. Listing Which Machines Are Taking Special Roles . . . 4-11
4.4.1. What Output Looks Like For a Simple File Server Machine . . . 4-12
4.4.2. What Output Looks Like for the System Control Machine . . . 4-12
4.4.3. What Output Looks Like For a Binary Distribution Machine . . . 4-13
4.5. Creating and Removing Processes . . . 4-14
4.6. Stopping and Starting Processes "Permanently" . . . 4-19
4.7. Stopping and Starting Processes Temporarily . . . 4-21
4.8. Stopping and Immediately Restarting Processes . . . 4-23
4.9. Setting the BOS Server's Restart Times . . . 4-25
4.10. Examining a File Server Machine's Log Files . . . 4-28
4.11. Rebooting a File Server Machine . . . 4-30

5. Managing Volumes . . . 5-1

5.1. Summary of Commands . . . 5-1
5.2. About Volumes . . . 5-3
5.2.1. The Three Types of Volumes . . . 5-3
5.2.2. How Volumes Improve System Efficiency . . . 5-4
5.2.3. The Role of the Volume Location Server and VLDB . . . 5-4
5.2.4. The Information in Volume Headers . . . 5-6
5.2.5. Keeping the VLDB and Volume Headers Synchronized . . . 5-7
5.2.6. About Mounting Volumes . . . 5-8
5.2.7. About Volume Names . . . 5-9
5.3. Creating ReadWrite Volumes . . . 5-10
5.3.1. Default Settings for a New Volume . . . 5-12
5.4. Clones and Cloning . . . 5-13
5.5. Creating and Releasing ReadOnly Volumes (Replication) . . . 5-14
5.5.1. The Types of Volumes to Replicate . . . 5-14

5.5.2. The Two Stages of AFS Replication 5-14
5.5.3. Determining if Replication Is Successful 5-14
5.5.4. Using the **-f** Flag to Force Creation of a New Clone 5-15
5.5.5. Variations on Replication 5-16
5.5.6. Using ReadOnly Volumes 5-16
5.6. Creating Backup Volumes 5-20
5.6.1. Backing Up Multiple Volumes at Once 5-20
5.6.2. Automating Creation of Backup Volumes 5-21
5.6.3. Making the Contents of Backup Volumes Available to Users 5-21
5.7. Mounting Volumes (Creating, Removing, and Listing Mount Points) 5-24
5.7.1. The Rules of Mount Point Traversal 5-24
5.7.2. The Three Types of Mount Points 5-25
5.7.3. Creating, Removing, and Listing Mount Points 5-27
5.7.4. Creating Mount Points in a Foreign Cell 5-30
5.8. Listing Information About Volumes 5-31
5.8.1. Examining the VLDB Entry for One or More Volumes 5-31
5.8.2. Examining the Volume Header for One or More Volumes 5-32
5.8.3. Examining Both the VLDB Entry and Volume Header for a Volume 5-36
5.8.4. Translating between Volume Names and IDs, Directory Names, and Locations 5-37
5.9. Moving Volumes 5-40
5.9.1. Moving the Three Types of Volumes 5-40
5.10. Synchronizing the VLDB and Volume Headers 5-42
5.10.1. Keeping the VLDB Synchronized with Volume Headers 5-42
5.10.2. Some Indications that the VLDB is Out of Sync 5-42
5.10.3. How to Synchronize the VLDB with Volume Headers 5-42
5.10.4. The **vos syncvldb** Command 5-43
5.10.5. The **vos syncserv** Command 5-43
5.11. Salvaging Volumes 5-45
5.11.1. The Kinds of Volume Corruption and Their Symptoms 5-45
5.11.2. How the Salvager Handles Corruption 5-45
5.11.3. How to Run the Salvager 5-46
5.12. Setting and Listing Volume Quota and Current Size 5-49
5.13. Removing Volumes and their Mount Points 5-53
5.13.1. Removing a ReadWrite Volume 5-53
5.13.2. Removing a ReadOnly Volume 5-53
5.13.3. Removing a Backup Volume 5-54
5.13.4. Other Removal Commands 5-54
5.14. Dumping and Restoring Volumes 5-58
5.14.1. About Dumping Volumes 5-58
5.14.2. About Restoring Volumes 5-58
5.15. Renaming Volumes 5-63
5.16. Unlocking and Locking VLDB Entries 5-65
5.16.1. Locking VLDB Entries 5-65
5.16.2. Unlocking VLDB Entries 5-65

6. Backing Up the System 6-1

6.1. Summary of Commands 6-1
6.2. Introduction to the Backup System 6-3
6.2.1. Volume Sets and Volume Entries 6-3
6.2.2. Dumps and Dump Sets 6-4
6.2.3. Dump Hierarchies and Dump Levels 6-4

6.2.4. Dump Names and Tape Names 6-6
6.2.5. Expiration Dates 6-6
6.2.6. Tape Recycling Schedules 6-7
6.2.7. Tape Archiving 6-8
6.2.8. Restoring Backed Up Data 6-9
6.2.9. Tape Labels, Dump Labels, and EOF Markers 6-10
6.2.10. Tape Coordinator Machines 6-11
6.2.11. The Backup Database 6-11
6.3. Configuring the Backup System 6-12
6.3.1. Determining Information About Tape Capacity and EOF Marker Size 6-12
6.3.2. Configuring Tape Coordinator Machines 6-13
6.3.3. Adding and Removing Tape Coordinators 6-17
6.3.4. Defining Volume Sets and Volume Entries 6-18
6.3.5. Creating Dump Hierarchies 6-22
6.3.6. Labeling Tapes 6-25
6.4. Interfaces to the Backup System 6-28
6.4.1. A Note on Performing backup Commands in a Foreign Cell 6-28
6.4.2. The Command Shell Window and Interactive Mode 6-28
6.4.3. The Tape Coordinator Window 6-32
6.5. Listing Backup Information 6-35
6.6. Backing Up Data 6-43
6.6.1. Some Warnings about Making Backups 6-43
6.6.2. The Backup Process: an Overview 6-44
6.6.3. Using Tapes with a Backup Operation 6-45
6.6.4. Placing Multiple Dumps on a Dump Set 6-47
6.7. Procedures for Automating the Backup System 6-50
6.7.1. Disabling Initial Prompts for Tape Insertion 6-50
6.7.2. Automatic Scheduling of Dumps 6-51
6.8. Restoring and Recovering Data 6-55
6.8.1. Specifying the Type and Destination of a Restore Operation 6-55
6.9. The Backup Database 6-59
6.9.1. About Backing Up the Backup Database 6-59
6.9.2. About Restoring the Backup Database 6-60
6.9.3. About Managing the Backup Database 6-63

7. Monitoring and Auditing Tools 7-1

7.1. Summary of Commands 7-1
7.2. Scout 7-2
7.2.1. About Scout 7-2
7.2.2. Starting and Stopping scout 7-8
7.2.3. Examples of scout Commands and Displays 7-9
7.2.4. Example 1 7-9
7.3. AFSMonitor 7-14
7.3.1. About AFSMonitor 7-14
7.4. The xstat Data Collection Facility 7-25
7.4.1. The ***xstat*** Libraries 7-25
7.4.2. Example ***xstat*** Commands 7-26
7.5. Auditing AFS Events on AIX File Servers 7-29
7.5.1. Enabling and Disabling AFS Auditing 7-29
7.5.2. Configuring AFS Auditing on AIX File Servers 7-30

8. Managing Authentication and Authorization Requirements 8-1

8.1. Summary of Commands 8-1
8.2. Authentication versus Authorization 8-2
8.3. Controlling Authorization Checking for a File Server Machine 8-3
8.3.1. When to Turn Off Authorization Checking 8-3
8.3.2. How the Server Processes Know When to Check for Authorization 8-3
8.3.3. Controlling Authorization Checking for the Entire Cell 8-3
8.3.4. The Need to Restart Processes 8-3
8.4. Preventing Mutual Authentication for an Individual Command 8-5
8.4.1. When to Prevent Mutual Authentication 8-5

9. Maintaining *CellServDB* on File Server Machines 9-1

9.1. Summary of Commands 9-1
9.2. About *CellServDB* 9-2
9.2.1. Negative Consequences of an Incorrect ***CellServDB*** 9-2
9.2.2. When to Update ***CellServDB*** 9-3
9.2.3. How to Maintain ***CellServDB*** 9-3
9.2.4. Clients Consult ***/usr/vice/etc/CellServDB*** 9-4
9.2.5. Informing Other Cells of Changes to Your ***CellServDB*** 9-4
9.3. Listing the Contents of *CellServDB* 9-5
9.4. Adding Database Server Machines to *CellServDB* 9-6
9.5. Removing Database Server Machines from *CellServDB* 9-8

10. Managing Server Encryption Keys 10-1

10.1. Summary of Commands 10-1
10.2. About Server Encryption Keys 10-2
10.2.1. The Role of Server Encryption Keys in Mutual Authentication: A Review 10-2
10.2.2. Maintaining AFS Server Encryption Keys 10-3
10.3. Listing Server Encryption Keys 10-6
10.4. Adding Server Encryption Keys 10-9
10.5. Removing Server Encryption Keys 10-12
10.6. Handling Server Encryption Key Emergencies 10-14
10.6.1. Preventing Mutual Authentication 10-14
10.6.2. Disabling Authorization Checking by Hand 10-15
10.6.3. Restarting Processes 10-15
10.6.4. Working Quickly on Each Machine 10-15
10.6.5. Working at the Console 10-15
10.6.6. Changing Individual ***KeyFile***s 10-16
10.6.7. Two Component Procedures 10-16

11. Installing New Server Process Software and Disks 11-1

11.1. Summary of Commands 11-1
11.2. About Installing New Server Process Binaries 11-2
11.3. Installing New Binaries 11-3
11.3.1. How and When a Process Starts Using New Binaries 11-3
11.3.2. Automatic Saving of the Current Version 11-3
11.3.3. Installation into Alternate Directories 11-3
11.4. Determining the Automatic Restart Time for New Binaries 11-7
11.4.1. Using New Binaries Immediately 11-7
11.5. Reverting to an Old Version of Binaries 11-9

11.5.1. How Reverting to an Old Version Works 11-9
11.5.2. Using the Reverted Binaries Immediately 11-9
11.5.3. Reverting Within an Alternate Directory 11-9
11.6. Checking Binary Version Dates 11-11
11.7. Removing Obsolete .BAK and .OLD Copies 11-12
11.8. Adding and Mounting a New Disk and/or Partition 11-13
11.8.1. About Adding a New AFS Partition 11-13
11.9. Removing and Unmounting a Disk and/or Partition 11-16

Managing Client Machines

12. Client Machine Processes and Files 12-1

12.1. About the Cache Manager 12-2
12.2. Configuration files in *\/usr/vice/etc* 12-3
12.3. Cache-related files in */usr/vice/cache* 12-5

13. Configuring the Cache Manager 13-1

13.1. Summary of Commands 13-1
13.2. Overview: Controlling Cache Manager Performance 13-2
13.3. Determining Cache Type, Size, and Location 13-4
13.3.1. Using Disk or Memory Caching 13-4
13.3.2. How the Cache Manager Determines Default Cache Size and Location 13-4
13.3.3. Setting and Listing Cache Size and Location 13-4
13.3.4. Recommendations for Cache Size 13-5
13.3.5. How the Cache Manager Chooses Data to Discard 13-7
13.4. Controlling Other Cache Parameters with afsd 13-8
13.4.1. Some Cache Configuration Parameters 13-8
13.4.2. Controlling Disk Cache Configuration 13-9
13.4.3. Controlling Memory Cache Configuration 13-10
13.4.4. Using Cache Manager Configuration Scripts 13-12
13.5. Determining if a Client Allows Programs to "Run Setuid" 13-15
13.6. Setting a Client’s File Server Probe Interval 13-17
13.7. Setting a Client’s Cell Membership 13-18
13.8. Logging and Disabling Cache Manager Messages 13-19
13.9. Monitoring Cache Manager Performance 13-21
13.10. Forcing the Update of Cached Data 13-24
13.11. Setting Cache Manager Preferences 13-26

14. Configuring Client Machines with package 14-1

14.1. Summary of Commands and Prototype/Configuration File Lines 14-1
14.2. Using the package Program 14-2
14.2.1. Using Package on File Server Machines 14-2
14.3. Package Overview 14-2
14.3.1. Preparing Prototype Files 14-3
14.3.2. Compiling Prototype Files 14-3
14.3.3. Preparing Clients 14-3
14.4. The package Directory Structure 14-4
14.4.1. The ***src*** directory 14-4
14.4.2. The ***lib*** directory 14-4
14.4.3. The ***etc*** directory 14-4

14.5. Example Prototype and Library Files 14-5
14.5.1. Example Prototype File 14-5
14.5.2. Example Library File 14-7
14.6. Package Configuration File Line Syntax 14-9
14.6.1. Local Files versus Symbolic Links 14-9
14.6.2. Defining a Directory 14-10
14.6.3. Defining a File 14-10
14.6.4. Defining a Symbolic Link 14-11
14.6.5. Defining a Block Special Device 14-11
14.6.6. Defining a Character Special Device 14-12
14.6.7. Defining a Socket 14-12
14.7. Constructing Prototype and Library Files 14-13
14.8. The Package *Makefile* 14-14
14.8.1. Overview 14-14
14.8.2. The CONFIG Section 14-14
14.8.3. The BASE_LIBS Section 14-15
14.8.4. The MACHINE_LIBS Section 14-16
14.8.5. The LIBS Section 14-17
14.8.6. The .SUFFIXES Section 14-17
14.8.7. The Makefile Instructions Section 14-17
14.9. Modifying the Makefile 14-19
14.9.1. Adding a New Prototype File 14-19
14.9.2. Adding a New System Type 14-19
14.9.3. Adding New Library Files 14-20
14.10. Compiling Prototype Files 14-21
14.11. Modifying Client Machines 14-22
14.12. Running package 14-24

15. Maintaining *CellServDB* and Cell Knowledge on Client Machines 15-1

15.1. Summary of Commands 15-1
15.2. About *CellServDB* 15-2
15.2.1. How the Cache Manager and Command Interpreters Use ***CellServDB*** 15-2
15.2.2. Maintaining ***CellServDB*** 15-3
15.2.3. The Format of ***CellServDB*** 15-3
15.2.4. Negative Consequences of Incorrect Cell Information 15-4
15.2.5. File Server Machines Consult ***/usr/afs/etc/CellServDB*** 15-5
15.3. Listing the Contents of *CellServDB* and the Kernel List 15-6
15.4. Changing the Kernel List of Cells Only 15-8
15.5. Changing *CellServDB* 15-10
15.6. Establishing a Central Update Source for *CellServDB* 15-13

Managing Users, Groups, and Protections

16. Creating and Deleting User Accounts With uss 16-1

16.1. Summary of Commands 16-1
16.2. About Creating and Deleting AFS User Accounts Using uss 16-2
16.2.1. Components of an AFS User Account 16-2
16.2.2. The Two Methods for Creating and Deleting Accounts 16-3
16.2.3. The Advantages of Using a Template File for Account Creation 16-3
16.2.4. Previewing Account Creation and Deletion 16-4

16.2.5. Avoiding Failures During Account Creation and Deletion 16-4
16.2.6. Reissuing uss Commands (to Recover from Creation/Deletion Failures) 16-5
16.2.7. The ***-admin*** Argument: An Important Note for Ultrix Users 16-6
16.2.8. Privilege Requirements for the uss Commands 16-7
16.3. About Creating Local Password File Entries Using uss 16-8
16.3.1. Assigning AFS and UNIX UIDs 16-8
16.3.2. Specifying Passwords in the Local Password File 16-8
16.4. About Using uss When Converting Existing UNIX Accounts 16-10
16.4.1. Making UNIX and AFS UIDs Match 16-10
16.4.2. Setting the Password Field Appropriately 16-11
16.4.3. Moving UNIX files into AFS 16-11
16.5. Constructing a Template for Account Creation 16-12
16.5.1. The Instruction Lines: A Functional Summary 16-12
16.5.2. The Three Types of User Accounts 16-12
16.5.3. Setting Template Values with Variables and Constants 16-13
16.5.4. Where to Place Template Files 16-16
16.5.5. Some General Rules for Constructing a Template 16-17
16.5.6. Some Example Templates 16-17
16.5.7. Creating a Volume with the "V" Line 16-19
16.5.8. Evenly Distributing User Home Directories with the "G" Line 16-24
16.5.9. Creating Directories with the "D" Line 16-25
16.5.10. Creating Files from a Prototype with the "F" Line 16-28
16.5.11. Creating One-Line Files with the "E" Line 16-30
16.5.12. Creating Links with the "L" and "S" Lines 16-34
16.5.13. Specifying the Level of Password/Authentication Security with the "A" Line 16-36
16.5.14. Executing Commands with the "X" Line 16-37
16.6. Creating Individual Accounts with uss add 16-39
16.6.1. Template Variables and Command Line Arguments: A Review 16-39
16.7. Deleting Individual Accounts with uss delete 16-45
16.7.1. Issues to Consider Before Deleting an Account 16-45
16.8. Creating and Deleting Multiple Accounts with uss bulk 16-48
16.8.1. Constructing a Bulk File for Account Creation and Deletion 16-48
16.8.2. Example Uses of the Bulk Input File 16-50

17. Administering User Accounts 17-1

17.1. Summary of Commands 17-1
17.2. The Components of a Complete AFS User Account 17-2
17.3. About Creating */etc/passwd* Entries 17-3
17.3.1. Assigning AFS and UNIX UIDs 17-3
17.3.2. Specifying Passwords in the Local Password File 17-3
17.4. About Converting Pre-Existing UNIX Accounts 17-5
17.4.1. Making UNIX and AFS UIDs Match 17-5
17.4.2. Setting the Password Field Appropriately 17-6
17.4.3. Moving UNIX files into AFS 17-6
17.5. Creating AFS User Accounts 17-7
17.5.1. The Three Types of User Accounts 17-7
17.5.2. Changing Default Quota and ACL After Volume Creation 17-12
17.6. Improving Password/Authentication Security 17-14
17.7. Changing Passwords 17-20
17.8. Listing and Setting Users' Volume Quota 17-21
17.9. Changing User Names 17-22

17.10. Removing User Accounts **17-25**

18. Administering the Protection Database **18-1**

18.1. Summary of Commands **18-1**
18.2. About the Protection Database **18-2**
18.2.1. Protection Database Entries 18-3
18.2.2. Using Groups Effectively 18-4
18.2.3. Using Prefix-Less Groups 18-6
18.2.4. Placing Machines in Groups 18-7
18.2.5. The Standard System Groups 18-7
18.2.6. A Note on Reusing AFS UIDs 18-9
18.3. Listing Information about Groups **18-10**
18.4. Listing Group-Related Information about Users and Machine Entries **18-13**
18.5. Creating User Entries **18-16**
18.6. Creating Machine Entries **18-18**
18.7. Creating Regular and Prefix-Less Groups **18-21**
18.7.1. Rules for Group Names and Ownership 18-21
18.8. Adding Members to Groups and Removing Them from Groups **18-25**
18.9. Deleting Entries from the Protection Database **18-27**
18.10. Changing the Owner of an Entry **18-28**
18.10.1. The Name-Changing Implications of Changing Ownership 18-28
18.11. Changing the Name of an Entry **18-30**
18.12. Changing Group-Creation Quota **18-32**
18.13. Listing and Setting the AFS UID Counters **18-34**
18.14. Listing and Setting Protections on Database Entries **18-36**
18.14.1. Default Rights 18-36
18.14.2. Constraints on Privacy Flags 18-37

19. Managing Access Control Lists **19-1**

19.1. Summary of Commands **19-1**
19.2. About File Protection in AFS **19-2**
19.2.1. Differences Between UNIX and AFS File Protection 19-2
19.2.2. The Implications of Directory-Level Protection 19-3
19.2.3. The Access Rights Defined 19-4
19.2.4. Normal versus Negative Rights 19-6
19.2.5. Groups on ACLs Can Extend Access to Many People 19-7
19.3. Examining ACLs **19-9**
19.4. About Changing ACLs **19-11**
19.5. Granting Access **19-13**
19.6. Denying Access **19-14**
19.7. Restoring Access After Denying It **19-16**
19.8. Completely Replacing an ACL **19-18**
19.9. Copying ACLs Between Directories **19-19**
19.10. Removing Obsolete AFS UIDs from ACLs **19-21**
19.11. How the UNIX Mode Bits Work in AFS **19-23**

20. Managing Privileged Users **20-1**

20.1. Summary of Commands **20-1**
20.2. The Different Types of Privilege **20-2**
20.2.1. Membership in ***system:administrators*** 20-2
20.2.2. The ADMIN Flag in the Authentication Database 20-2

20.2.3. Inclusion in ***/usr/afs/etc/UserList*** 20-2
20.2.4. The Reason for Separate Privileges 20-3
20.2.5. A Warning about ***anonymous*** 20-3
20.3. Granting Privilege for pts and fs: the *system:administrators* Group 20-4
20.4. Granting Privilege for kas: the ADMIN Flag 20-5
20.5. Granting Privilege for bos, vos, and backup: the *UserList* 20-7
20.6. Granting All System Administrator Privileges 20-10

Appendixes

Appendix A. AFSMonitor Statistics A-1

A.1. The Cache Manager Statistics A-2
A.1.1. Performance Statistics Section (PerfStats_section) A-3
A.1.2. Server Up/Down Statistics Section (Server_UpDown_section) A-4
A.1.3. RPC Operation Measurements Section (RPCop_section) A-8
A.1.4. Authentication and Replicated File Access Section (Auth_Access_section) A-22
A.2. The File Server Statistics A-24
A.2.1. Performance Statistics Section (PerfStats_section) A-24
A.2.2. RPC Operations Section (RPCop_section) A-27

Appendix B. AIX Audit Events B-1

B.1. Introduction B-2
B.2. Audit-Specific Events B-3
B.3. Volume Server Events B-4
B.4. Backup Server Events B-7
B.5. Protection Server Events B-10
B.6. Authentication Events B-12
B.7. File Server and Cache Manager Interface Events B-14
B.8. BOS Server Events B-16
B.9. Volume Location Server Events B-19

Index Index-1