Table of Contents

Replace this page with the tab separator for

Table of Contents

About This Guide

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

Audience and Purpose

This guide describes concepts and procedures necessary for using an AFS® file system. This document is intended for AFS users who are familiar with UNIX® but not necessarily AFS.

Basic AFS concepts and guidelines for using the AFS file system are described in the beginning of this document. Concepts important to a specific task (or group of related tasks) are presented in context, prior to the procedures. The *AFS User's Guide* also describes some of the differences between the UNIX file system and AFS.

The remainder of the *User's Guide* provides explanations, examples and detailed procedures for basic AFS functions, including logging in, changing a password, listing information, protecting files, creating groups, and troubleshooting.

Procedures describe the command, switches, and instances necessary to do a specific task. Procedures are task-specific and may not describe each command option; Appendix A describes each command in detail.

Organization

This document is divided into nine tabbed sections; some sections have several chapters. Below is a short description of the contents of these tabbed sections:

- **Concepts**

  This section introduces users to the basic concepts and functions of AFS. The *AFS User's Guide* is designed for experienced UNIX users with no previous knowledge of AFS. It is important that users be familiar with the terms described in this section before using AFS. The *Concepts* section also describes AFS command structure and AFS online help.

- **Using AFS**

  This section describes the concepts and task-oriented procedures for using AFS: logging in, authenticating, unlogging, logging out, accessing files and directories in AFS, and changing your passwords.

- **Listing Information**

  This section describes how to list information concerning AFS: listing your volume quota, the location of directories, the status of file server machines, and the foreign cells you can access.

- **Protecting Directories**

  This section describes the concepts and procedures necessary to protect your directories using AFS access control lists: how to list directory protection, set directory protection, and create and manage groups.

- **Troubleshooting**

  This section describes a basic diagnostic sequence and step-by-step diagnostic and corrective steps for specific problems.

- **Appendix A: Command Reference**

  This section is a reference for AFS command syntax and descriptions and command short forms and aliases. It is intended for advanced AFS users.

- **Appendix B: NFS/AFS Translator**

  This section describes how to use the NFS/AFS Translator to access the AFS file space from NFS™ (Networking File System).

- **Glossary**

  This section defines terms used in the *AFS User's Guide*.

- **Index**

  This section provides an overall index listing for the *AFS User's Guide* and separate index entries for major topics such as commands, examples, procedures, and troubleshooting.

How To Use This Guide

Before you begin using AFS, read the *Concepts* section. This will give you an overview of the AFS file system and define the terms you will encounter while working with AFS.

Next, follow the procedures outlined in the *Using AFS* section. This will get you started using AFS as an authenticated user. This section describes how to access files in the AFS file space and how to quit AFS.

The remainder of the document describes specific procedures for using AFS. Each procedure describes the command syntax, the required switches and instances, and then presents several examples.

If you require more information, Appendix A, gives detailed descriptions and syntax for AFS commands.

Related Documents

The AFS Documentation Kit also includes the following documents:

- ***AFS Command Reference Manual***

  This reference manual details the syntax of each AFS command and is intended for the experienced AFS administrator, programmer, or user. For each AFS command, the *AFS Command Reference Manual* list the command syntax, aliases and abbreviations, description, arguments, warnings, output, examples, and related topics. Procedural and conceptual information is not included. Commands are organized alphabetically.

- ***AFS System Administrator's Guide***

  This guide describes concepts and procedures necessary for managing an AFS system. The procedures described in this document should be performed by AFS system administrators.

- ***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.

Document Conventions

The following conventions are used in this document:

1. Command names, switches, and flags appear in **bold type**.
2. Variable information (user-specific information, such as a machine name) and command instances appear in *italic type*.
3. Filenames appear in *italic type*.
4. New terms appear in ***bold italic type***.
5. Sample command lines appear in **bold type**.
6. Example screen output appears in `typewriter type`.
7. The following symbols appear in command syntax and online help. **Do not type these symbols when entering a command**.
   - Square brackets [ ] surround optional items.
   - Angle brackets < > surround instances (user-supplied information).
   - A plus sign + follows a switch that accepts multiple instances.
   - A percent sign `%` represents the command shell prompt. Your prompt may be different.

Concepts

Replace this page with the tab separator for

Concepts

1. Basic AFS Concepts

This chapter introduces basic concepts for the AFS file system and defines terms. It assumes that you are already familiar with standard UNIX® commands, file protection, file system hierarchy, and pathname conventions. It includes the following sections:

1.1. Working in AFS

AFS makes it easy for people to work together on the same files, no matter where the files are located. AFS users do not have to know the physical location of files, even though a file can be stored on any number of file server machines. AFS users simply request a given file by its uniform UNIX pathname and AFS finds the correct file automatically, just as systems with a single file storage machine do.

However, while AFS makes file sharing easy, it does not compromise the security of the shared files. It is just as easy to prohibit other users from accessing files as it is to give them access.

1.1.1. Client/Server Computing

The AFS file system utilizes a ***client/server computing*** model. In client/server computing, there are two types of machines: ***server machines*** and ***client machines***. Server machines store data and perform services for client machines. Client machines perform computations for users often by accessing data and services on server machines. Some machines act as both clients and servers. In most cases, you will work on a client machine, accessing files stored on a ***file server machine***. Figure 1-1 illustrates the basic client/server interactions.

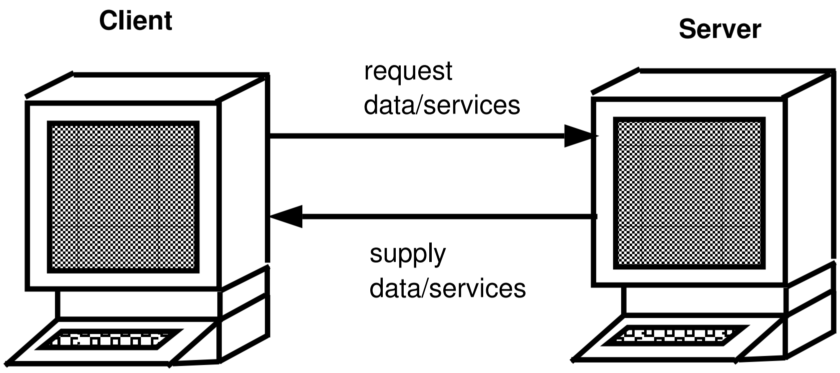

Figure 1-1. AFS Uses a Client/Server Computing Model

1.1.2. A Distributed File System

AFS is a ***distributed file system*** which joins the file systems of individual file server machines so that files stored (distributed) on each file server machine in a ***network*** is as accessible to a user as the files stored on his or her client machine's local disk. Figure 1-2 graphically represents how a distributed file system like AFS is set up on a network.

A distributed file system has two main advantages over a conventional centralized file system:

- Increased System Availability -- Copies of popular software programs can be distributed on many file server machines so the loss of a single machine (and sometimes even multiple machines) does not necessarily make those programs unavailable. Instead, user requests for those programs are routed to available machines on the network. Under a centralized file system, the loss of the central file storage machine effectively shuts down the entire system.
- Increased System Efficiency -- In a distributed file system, the work load is distributed over many smaller (and less expensive) file server machines that tend to be more fully utilized than the larger (and more expensive) file storage machine of a centralized file system.

AFS hides the existence of its underlying distributed nature. Working on AFS "looks and feels" like working on the file system of a single local machine – except you can access many more files. And because AFS relies on the power of users' client machines for computation, the number of users on the AFS file system do not slow AFS performance appreciably, making it a very efficient computing environment.

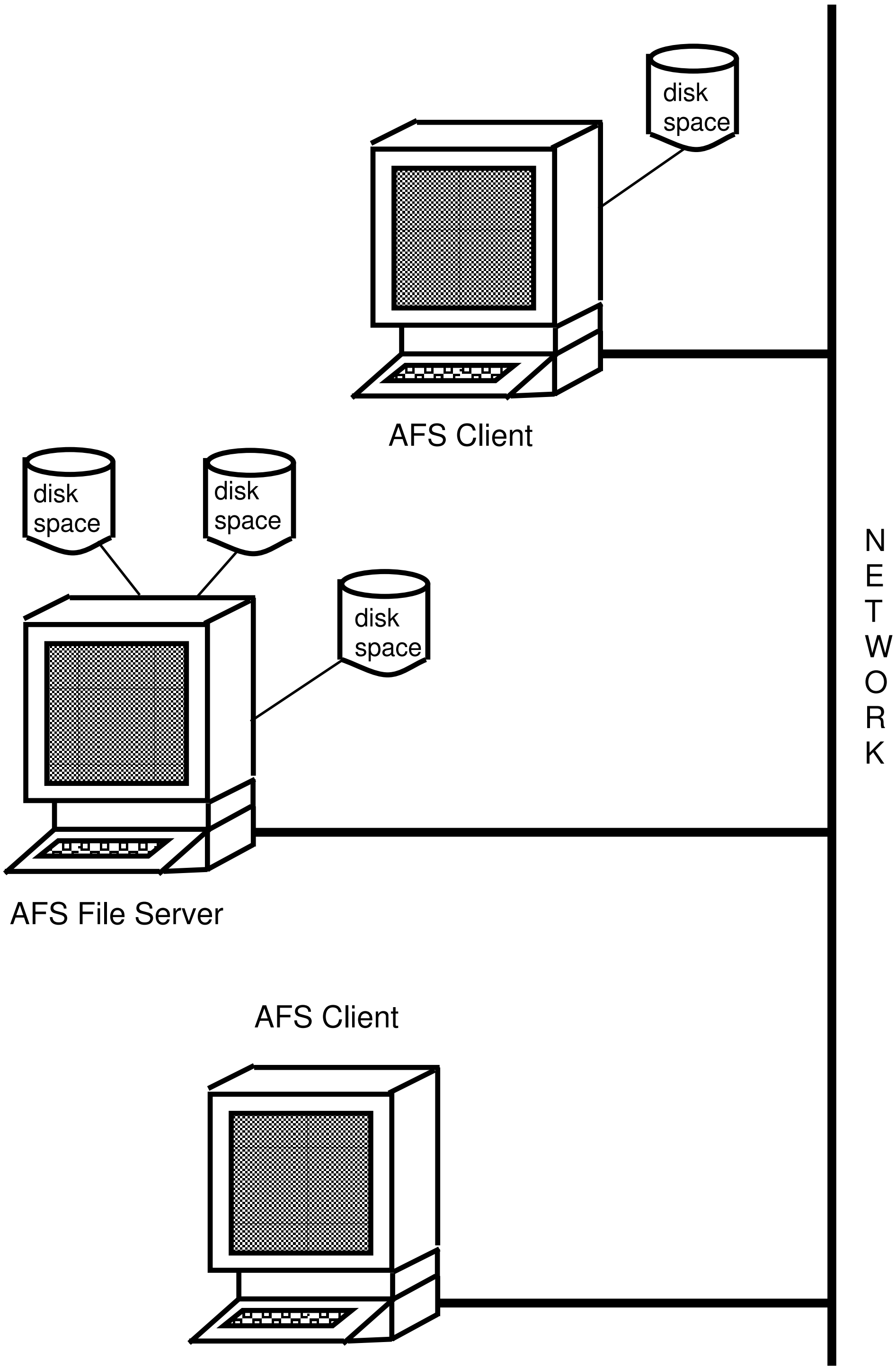

Figure 1-2. AFS Operates on a Network of Client and Server Machines

1.2. The Components of AFS

1.2.1. AFS File Space and Local File Space

Just like a UNIX file system, AFS uses a hierarchical file structure – a tree. By convention, */afs* is the root and all of the subdirectories and files under */afs* make up the ***AFS file space***. The AFS file space is an extension of a local machine's file space. The other directories in the file structure, which might include */usr*, */etc*, */bin,* will either be located on your local disk or contain links to files stored in AFS.

Files relevant only to the local machine are stored in the local machine file space, freeing the local machine's disk space for other uses. All other files can be stored in AFS, allowing shared access to and use of those files.

Note: You can only use AFS commands on files in the AFS file space or via links made to the AFS file space.

1.2.2. Cells and Sites

The ***cell*** is the administrative domain in the AFS file space. Each cell is autonomously administered. The cell's administrators determine how client machines are configured and how much file server machine storage space is available to each user.

A cell's domain can consist of a company, a university department, or any defined group of users. From a hardware perspective, a cell is a grouping of client machines and server machines that are members of the same "home" or local cell. Figure 1-3 illustrates a cell from a hardware perspective An AFS ***site*** is a grouping of one or more related cells. For example, the cells that make up Carnegie Mellon form a single site.

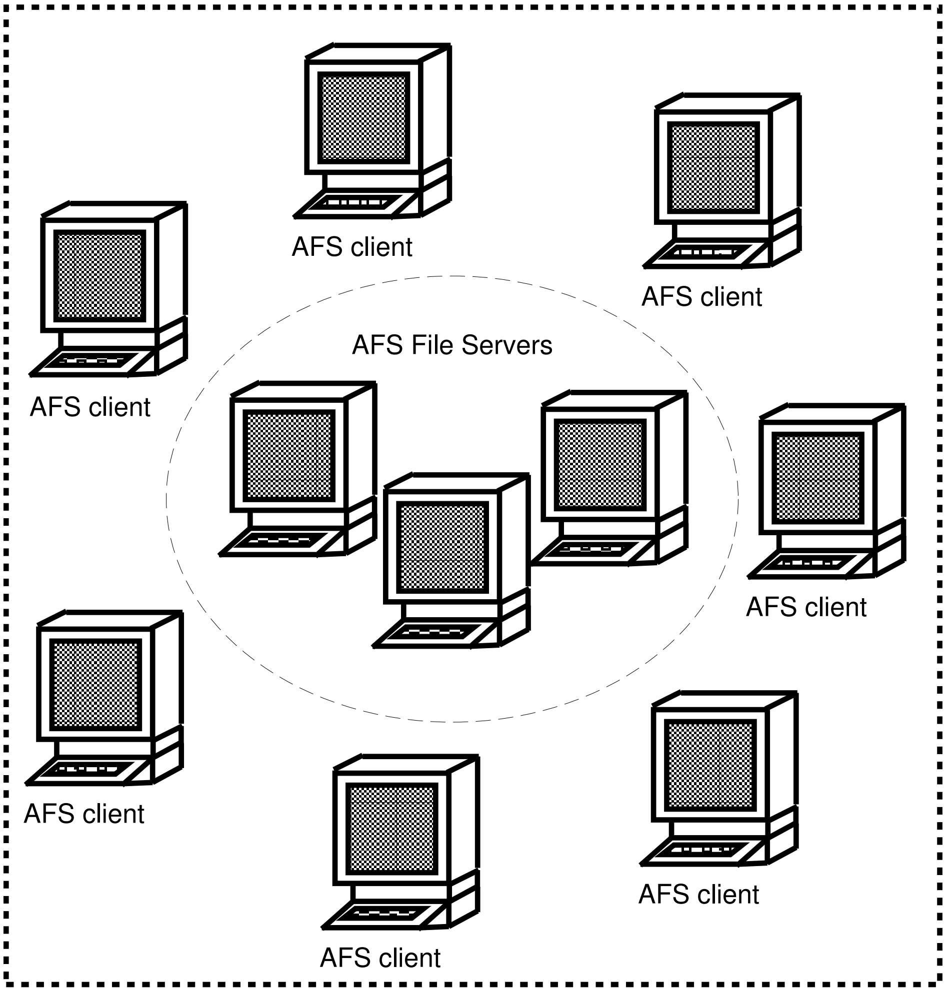

Figure 1-3. A Cell from a Hardware Perspective

By convention, the subdirectories of */afs* are the cellular file trees which contain subdirectories and files relevant to a single cell. For example, directories and files relevant to the Transarc Corporation cell are stored in the subdirectory */afs/transarc.com*.

While it organizes and maintains its own file space, each cell and each site can also connect with the file space of the other cells and sites running AFS on the same network. The result is a huge file space that allows file sharing within and across cells and sites. Figure 1-4 illustrates this interaction.

The cell containing the client machine you are using is called your ***local cell***. All other cells in the AFS file space are termed ***foreign cells***.

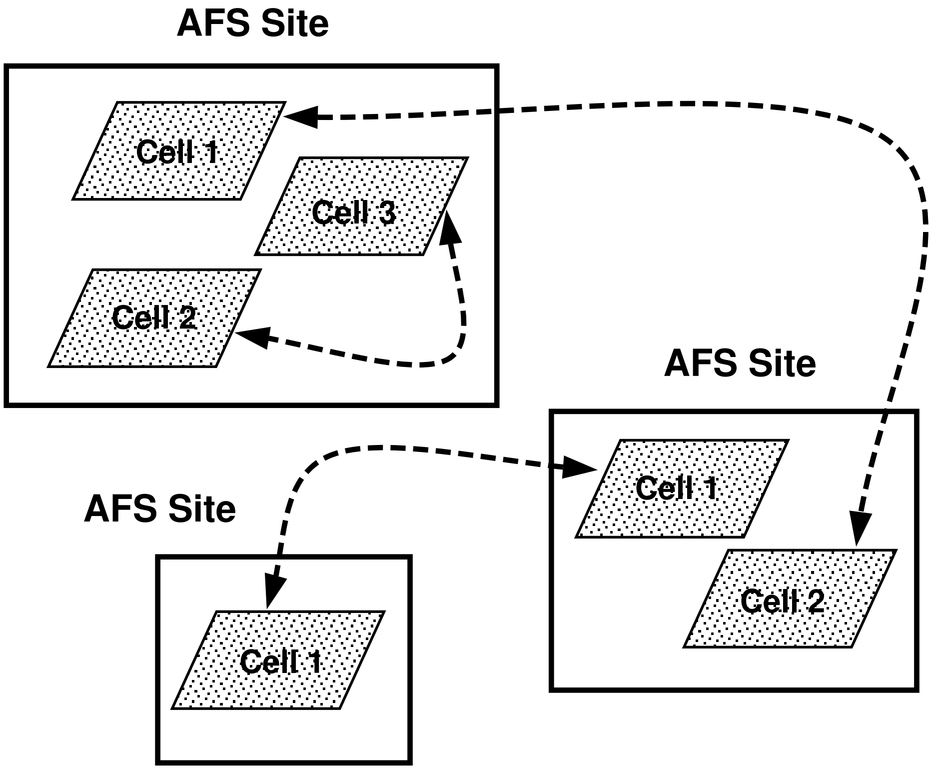

Figure 1-4. AFS Allows File Sharing within and across Cells and Sites

1.2.3. Volumes and Mount Points

The disks in a computer are divided into sections called ***partitions***. AFS further divides partitions into subsections called ***volumes***. A volume houses a subtree of related files and directories. The volume provides a convenient "container" for storing, backing up, and moving related files and directories.

Access to a volume is through the volume's ***mount point*** which points to the location (the file server machine and partition) of the volume. A mount point looks like a regular UNIX directory and when you move (that is, **cd**) to a new directory you may cross a mount point without even knowing it. When you cross a mount point you move from one volume to another and possibly from one file server and partition to another. Of course, this will be transparent to you because AFS interprets mount points and retrieves the files and directories that you request from the appropriate location, relieving you of the need to know their volume, partition, and file server locations.

You do not need to know which file server machine and partition houses your volume because AFS finds your files for you automatically. If you are interested, you can find a volume's location; section 4.2 tells you how.

Most AFS cells use one volume for each user's home directory; this allows all files and subdirectories in a user's home directory to be stored together on one partition of one file server machine. User volumes are typically named *user.username*. For example, a user named *smith* in the cell *comp.com* has a volume named *user.smith* and a mount point in the */afs/comp.com/usr* directory named *smith* pointing to the volume *user.smith*. Figure 1-5 provides an example of these relationships.

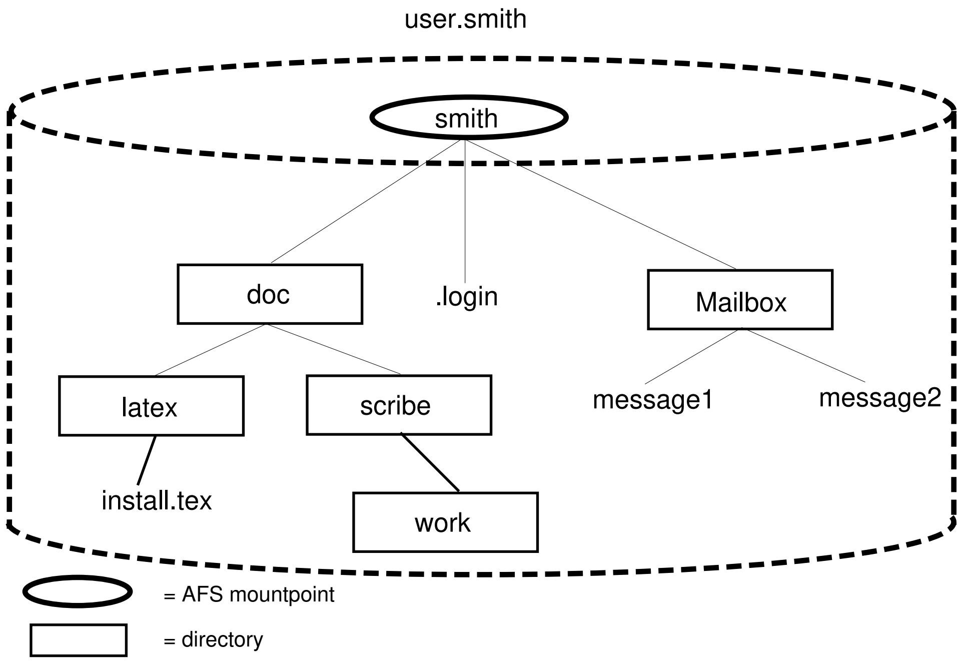

Figure 1-5. A Mount Point is the Entrance to a Volume

Because individual user home directories are nearly always contained in the same volume, when you **cd** from one home directory to another you often cross volume, partition, and file server boundaries. This is also true for other related groupings of directories.

Figure 1-6 illustrates this point. If you start at the root (*/afs*) of the AFS file tree in the example Transarc cell, you retrieve information from partition */vicepa* on Transarc File Server 1 (*/afs* is a mount point to the volume `root.afs` located on partition */vicepa* of Transarc File Server 1). If you **cd** into the Transarc cell file space in */afs/transarc.com*, you access the volume `root.cell` mounted at */afs/transarc.com* and you retrieve information from partition */vicepa* on Transarc File Server 2. Notice that *smith*'s home directory is located in a volume on Transarc File Server 2, while *pat*'s home directory is located in a volume on Transarc File Server 1. If *smith* needed files in the cs.cmu.edu cell, he would **cd** to */afs/cs.cmu.edu* and the data retrieved would come from */vicepa* on cs.cmu.edu File Server 1 (in the volume `root.cell` mounted at */afs/cs.cmu.edu*).

Because AFS volumes are stored on different file servers, when one file server crashes only the volumes on that server are inaccessible. Volumes stored on other servers are still accessible. However, if the mount point to a volume is stored in a volume on a crashed server, the former volume is also inaccessible. For that reason, volumes containing frequently used directories (for example, */afs* and */afs/<cellname>*) are often copied and distributed to many file servers.

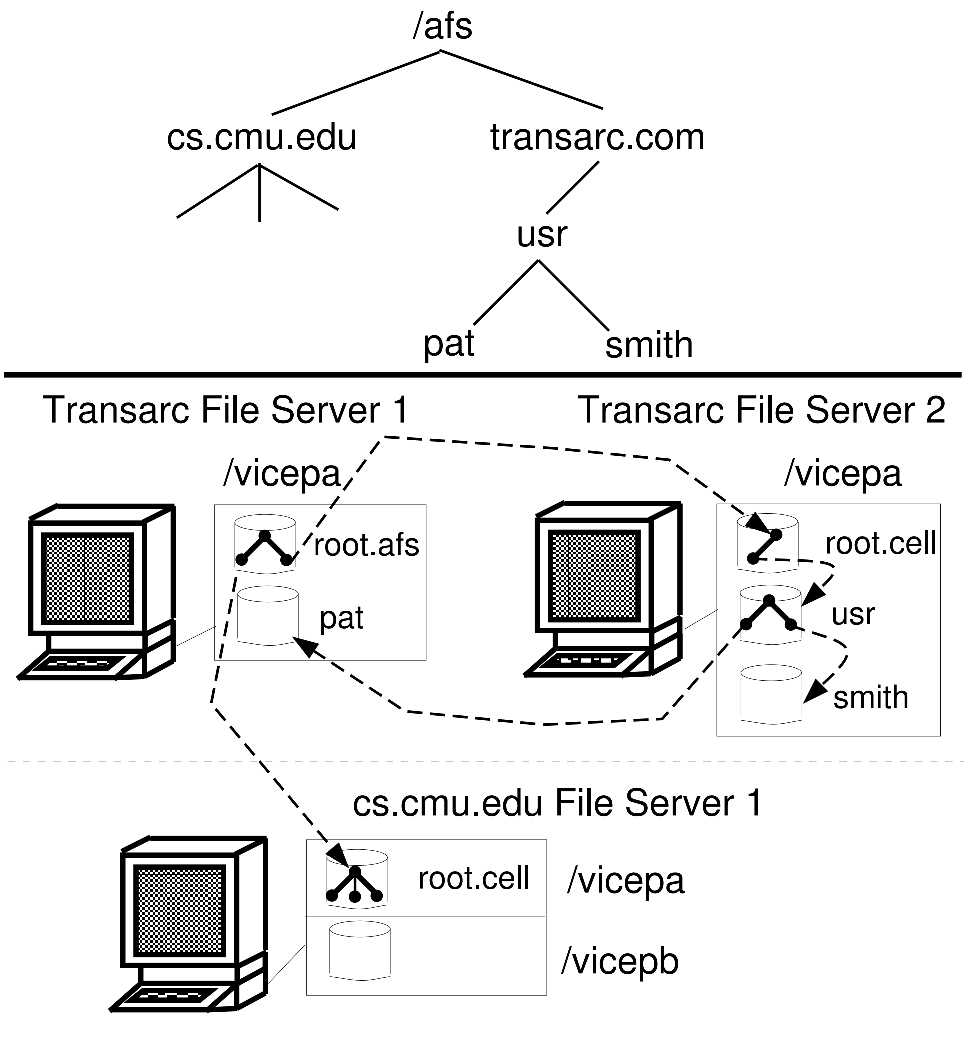

Figure 1-6. A Pathname Can Contain Mount Points to Many Volumes

1.2.4. Volume Quotas

Each volume has a size limit, or ***quota***, assigned by the system administrator. A volume's quota, measured in 1 kilobyte blocks, represents the maximum amount of disk space the volume can contain. If you attempt to exceed a volume quota, you will receive an error message. Section 4.1 explains how to check the quota on a volume.

Volumes have completely independent quotas. For example, say your current directory is */afs/comp.com/usr/smith/public*, and the user.smith volume has 1000 free blocks. You **cd** to */afs/comp.com/usr/pat* and try to copy a 500 block file from your directory to *pat's* and discover that you do not have enough space. You check the volume quota for the *user.pat* volume, and find that the volume only has 50 free blocks. This occurs because *smith's* volume has a different quota than *pat's*.

1.3. Using Files in AFS

1.3.1. The Client’s Cache Manager

As an AFS user, you work on an AFS client machine. A process called a ***Cache Manager*** on that machine is your agent in accessing information stored in the AFS file space. When you access a file, the Cache Manager on your client machine requests the file from the appropriate file server machine and stores (or "caches") a copy of the requested file on your client machine’s local disk. Your client machine uses the local copy of the cached file so it does not need to continue sending network requests to the file server for data from the stored file.

Because the Cache Manager is storing a copy of the file on your machine, any changes you make to your copy of the file are not made to the central version until the file closes. When the Cache Manager sends the changed file back to the appropriate file server, your changed version replaces the file stored on the server. For instance, every time you issue your text editor’s standard "save" command your cached copy is written to the file server’s disk.

Should the file server storing the file you are working on crash, you may continue to work locally on that file, but you cannot save it until the server is back up.

1.3.2. Updating Copies of Cached Files

When the central version of a file changes on the file server, AFS advises all other Cache Managers with copies of that file that their version is no longer valid. AFS has a special mechanism to ensure that these Cache Manager notifications are performed efficiently. When the file server sends the Cache Manager a copy of a modifiable file, it also sends a ***callback***. A callback is a "promise" from the file server that it will contact the Cache Manager if the centrally stored copy of the file is changed while it is being used. If that happens, the file server "breaks" the callback. If you run a program requesting data from that changed file, the Cache Manager notices the broken callback and gets an updated copy of the file from the file server. Callbacks ensure that you are working with the most recent copy of a file.

Note: The callback mechanism does not guarantee that you will immediately see the changes someone else makes to a file you are using. Your Cache Manager does not notice the broken callback until your application program asks it for more data from the file.

1.3.3. Multiple Users Modifying Files

As with a standard UNIX file system, if multiple users modify the same file, the changes saved last are the changes you see, regardless of who made the changes. When collaborating with someone on the same files, it is important to coordinate your work so you do not overwrite each other’s changes. AFS allows you to prevent other users from accidentally overwriting your files by limiting access to your directories using access control lists (ACLs). (See chapter 5 for further information about ACLs.)

1.4. AFS Security

Because AFS is easily accessed by many users, several methods are used to ensure system security, including:

- Passwords and mutual authentication to verify that only authorized users access AFS.
- Access control lists to allow individual users to restrict or allow access to their own directories.

1.4.1. Passwords and Mutual Authentication

AFS uses two related methods to ensure that only authorized users access AFS: passwords and mutual authentication. Both methods require that a user prove his or her identity to the system.

When you first identify yourself to AFS, you must authenticate yourself to prove that you are who you say you are. To do this, you must type in the password associated with your user name.

When you correctly type your AFS password, your Cache Manager receives a ***token***, indicating that you are a valid AFS user. A token is a package of information that is scrambled by an AFS authentication program using your AFS password as a key. Your Cache Manager can unscramble the token because it knows your password and AFS’s method of scrambling. If your Cache Manager can unscramble the token and use its information, you are ***authenticated*** as an authorized AFS user.

The token is proof to the other AFS file servers that you are authenticated and can access the AFS file space. This serves as the basis for the second means through which AFS creates security, called ***mutual authentication***. Under mutual authentication, both parties communicating across the network prove their identities to one another. AFS requires mutual authentication whenever a server and client (most often, a Cache Manager) communicate with each other.

The mutual authentication protocol that AFS uses is designed to make it very difficult for people to fraudulently authenticate; here’s how it works. Before it can communicate with file servers, your Cache Manager must have a valid token; it gets this token when you authenticate with AFS. When your Cache Manager contacts a file server, it also sends your token, coded to be recognized only by an AFS file server. If the server recognizes your token it can communicate with your Cache Manager. In turn, the Cache Manager accepts the file server as genuine if the file server can unscramble and use the information in the token. Mutual authentication is complete when both your Cache Manager and the file server have proven their identities to one another.

1.4.2. Access Control Lists

AFS uses ***access control lists (ACLs)*** to determine who can access the information in the AFS file space. An ACL exists for each directory in the file system, specifying what actions different users can perform on that directory and its files. An ACL can contain up to 20 user and/or group entries; each entry lists the user or group and the access rights of each user or group.

The owner of a directory and system administrators can always define the composition of an ACL. Users automatically own their home directories and subdirectories. Other non-owner users can define a directory's ACL only if specifically granted that right on the ACL. See chapter 5 for more information on access control lists.

A group can be composed of one or more users and client machines. If a user is a member of a group, he or she has all of the rights granted to that group (as if he or she were listed directly on the ACL). If a user is logged into a client machine that is a member of a group, he or she has all of the rights granted to that group. See chapter 6 to learn how to define groups.

Users who are not authenticated in your local cell are automatically assigned to a group of unauthenticated users called *system:anyuser*. See section 5.3.3 for a discussion of how to grant access rights to users in the *system:anyuser* group.

Note: You can use the UNIX mode bits to further control access on specific files within an AFS directory; however, the effect of these mode bits is different under AFS than standard UNIX. See section 1.5.3 for more information.

1.5. Differences Between UNIX and AFS

AFS is designed to be similar to the UNIX file system. For instance, many of the basic UNIX file manipulation commands (**cp** for copy, **rm** for remove, etc.) are the same in AFS as they are as in standard UNIX. All your application programs will also work as they did before. However, there are differences between a standard UNIX file system and AFS. These differences are discussed in sections 1.5.1 through 1.5.5.

1.5.1. File Sharing

AFS allows users to share remote files as easily as local files. To access a file on a remote machine in AFS, you simply specify the file's pathname. In standard UNIX, you must either log in to the remote machine or explicitly transfer the file from the remote machine to the local machine.

AFS users can see and share all the files under the */afs* subtree, given the appropriate privileges. An AFS user who has the necessary privileges can access a file in any AFS cell, simply by specifying the file's pathname. File sharing in AFS is not restricted by geographical distances or operating system differences.

1.5.2. Login and Authentication

To become an authenticated AFS user, you need to provide an authentication password to AFS. There are two methods of doing this:

- In cells using the AFS login program, logging in is a one-step process; your initial login provides authentication.
- In cells not using AFS login, you must
    1. Log in to your local machine.
    2. Issue the **pagsh** command to create a place to hold your token.
    3. Issue **klog** to authenticate with AFS and get your token.

AFS authentication passwords are stored in the AFS Authentication Database, rather than in the local password file (*/etc/passwd* or equivalent). If your cell uses AFS login, you may change your password with a single command. In cells not using AFS login, you will need to issue two commands to change the passwords in AFS's Authentication Database and in the local password file.

See section 3.4 for more information on logins and authentication.

1.5.3. File and Directory Protection

AFS does not rely on the mode bit protections of a standard UNIX system (though it does interact with these mode bits). Instead, AFS uses an ACL on each directory to control access. The differences between the two methods are summarized below:

- UNIX mode bits use three types of access rights: execute, read, and write. You can define up to seven types of access rights on an ACL: read, lookup, insert, delete, write, lock, and administer. See section 5.2 for a discussion of these rights. Section 5.7 explains how AFS uses UNIX mode bits.
- UNIX mode bits allow you to grant three levels of access rights: the rights of the user (*owner*), the rights of the other members of the user's group (*group*), and the rights of all users (*other*). On an ACL, you may place up to 20 entries (user/group and corresponding access rights). Unlike standard UNIX, a user may be a member of an unlimited number of groups and groups can be defined by both users and system administrators (in addition to the three system-defined groups). Chapter 6 describes the use of groups in AFS.
- UNIX mode bits are set individually on each file and directory. An ACL applies to all of the files in a directory. While at first glance the AFS method might seem less precise, in actuality (given a proper directory structure) there are no major disadvantages to directory level protections and they require much less work to establish and maintain.

1.5.4. Machine Outages

The kinds of failures you experience when a standard UNIX file system goes down is different than when one or more individual AFS file servers are unavailable. When a standard UNIX file system crashes, the system simply "locks up" or "freezes" and you may loose changes to any files in which you were working.

When an AFS file server machine crashes, you will not be able to access the files on that machine, but if a copy of a file you are attempting to access is available on another file server machine you may not even notice the server outage. This is because AFS gives your cell's system administrators the ability to store copies of popular programs on multiple file servers. Choosing between these copies is done by the Cache Manager; when one copy becomes unavailable, the Cache Manager simply chooses another.

If there are no other copies of a file stored on a crashed server, you might still be able to use that file if an up-to-date copy is held by your client machine's Cache Manager. However, you cannot save changes to files stored on a crashed file server until that server is running again.

1.5.5. Remote Commands

UNIX remote commands (**ftp, rcp, rsh, rlogin**) remain available in AFS. The remote commands run programs on a remote machine without explicitly telnetting to it. If the remote machine has a Cache Manager, your token is used there also and you are authenticated as long as the remote command executes. If the remote machine does not run a Cache Manager, you receive the message:
"`Warning: unable to authenticate.`" You are logged into the remote machine, but you are not authenticated to AFS. You can access the local files on the remote machine and the AFS directories as *system:anyuser*, but you cannot access protected AFS directories.

1.6. Using AFS with NFS

Many sites currently use the Networking File System (NFS). Their machines can access the AFS file space through the NFS/AFS Translator™. Appendix B explains the use of the NFS/AFS Translator.

2. AFS Command Syntax and On-line Help

The AFS commands available to regular users are used to authenticate users, list AFS information, protect directories, create and manage groups, and create and manage ACLs. There are three general types of commands available to all AFS users: file server commands, protection server commands, and miscellaneous commands. This chapter discusses the syntax of these AFS commands, the rules that must be followed when issuing them, and ways of accessing help relevant to them.

The chapter includes the following sections:

2.1. AFS Command Syntax

Most AFS commands use the following syntax:

<**command suite**> <*operation code*> **-switch** <*instance*>+ [**-flag**]

The ***command suite*** indicates the general type of command and the server process that is contacted to initiate the command. Regular AFS users have access to two main command suites and a miscellaneous set of commands:

- The **fs** command suite is used to issue file server commands that interact with the ***File Server*** process.
- The **pts** command suite is used to issue protection commands that interact with the ***Protection Server*** process.
- The miscellaneous commands are not associated with any command suite; these commands do not have a command suite, rather they have an operation code only.

The ***operation code*** indicates the action that the command will perform. Miscellaneous commands have operation codes only.

A command may have multiple ***arguments*** and/or ***flags***:

- Arguments are used to supply additional information for use by the command and consist of a paired ***switch*** and ***instance***. A switch defines the type of argument and is always preceded by a hyphen; arguments may take multiple instances if a plus sign (+) appears after the instance. An instance represents some variable piece of information that will be used by the command. Arguments may be optional or required.
- Flags are used to direct a command to perform in a specific way (for example, to generate a specific type of output). Flags are always preceded by a hyphen and are always optional.

2.1.1. AFS Command Symbol Conventions

The following symbols appear in the documentation when describing command syntax. Do not type these symbols when typing a command.

- Square brackets [ ] surround optional items.
- Angle brackets < > surround instances (user-supplied information).
- A plus sign + follows an argument that accepts multiple instances.
- A percent sign % represents the command shell prompt.
- Words in *italic* indicate variable input.
- Words in **bold** must be typed exactly as they appear.

2.1.2. Command Syntax Example

In the following AFS command

`%` **fs setacl -dir $HOME -acl pat all terry none -negative**

- **fs** is the command suite that indicates that this is a file server command.
- **setacl** is the ***operation code*** that directs the File Server process to set an access control list.
- **-dir $HOME** and **-acl pat all terry none** are ***arguments***.
  - **-dir** and **-acl** are switches; **-dir** indicates the name of the directory whose ACL should be set, and **-acl** indicates how the ACL should be be set.
  - **$HOME** and **pat all terry none** are ***instances*** of the arguments. **$HOME** defines a specific directory for the directory argument. The **-acl** argument has two instances specifying two ACL entries: **pat all** and **terry none**.
- **-negative** is a flag; it directs the command to put the access list entries on the negative rather than the normal list.

2.2. Rules for Using AFS Commands

This section contains rules you should follow when using AFS commands.

2.2.1. Spaces and Lines

Separate each command element (command suite, operation code, switches, instances, and flags) with a space. Multiple instances of an argument are also separated by a space.

Type all AFS commands on one line, followed by a carriage return. Some commands in this document appear on more than one line, but that is for legibility reasons only.

2.2.2. Abbreviations and Aliases for Operation Codes

You can type operation codes in one of three ways:

- You can type the operation code in full. The operation codes for miscellaneous commands must always be typed in full.
- You can abbreviate the operation code as long as the abbreviation is unambiguous (that is, it cannot be confused with any other operation code in the particular command suite).
- You can use the alias for the operation code, if one exists.

For example, the **fs listacl** command can be issued as follows:

- **fs listacl** (full command)
- **fs lista** (abbreviation)
- **fs la** (alias)

Appendix A provides information on the full and abbreviated command syntax as well as any aliases for all of the commands discussed in this guide.

2.2.3. Omitting Argument Switches

You may omit an argument's switch if

- The command has only one argument.
- None of the arguments that precede that argument are missing.
- The arguments are arranged in the proper order (according to the syntax definition for that command).
- None of the arguments that precede that argument are used to supply multiple instances.

For example,

% fs setacl /afs/tr/usr/terry/private pat rl

is an acceptable short form for

% fs setacl -dir /afs/tr/usr/terry/private -acl pat rl

However, the following is not an acceptable short form because the arguments are not in the prescribed order:

% fs setacl -acl pat rl /afs/tr/usr/terry/private

2.2.4. Shortening Switches and Flags

If you are required to use a switch or if you decide to use a flag, you can often shorten the name of that switch or flag. When you shorten a flag or switch name, make sure that the shortened version cannot be mistaken for any other flag or switch within that command.

For example, if you were issuing the **fs setacl** command, you could abbreviate all of the switches and flags of the command down to the initial letter because all of the switches and flags begin with a different letter. However, if you were issuing the **knfs** command, the **-host** argument and **-help** flag both begin with an "h," so the shortest unambiguous abbreviation for **-host** is **-ho** and the shortest for **-help** is **-he**. See appendix A for information on the syntax abbreviations of the commands in this Guide.

2.2.5. Shortening Directory References

Of the AFS command arguments that require directory or pathnames instances, most will accept one or more of the following short forms:

- Single dot (**.**) – The single dot is used to specify the current working directory.
- Double dot (**..**) – The double dot is used to specify the parent directory of the current working directory.
- Home environment variable (**$HOME**) – This environment variable specifies the user's home directory.

For example, if the user *terry* wanted to give READ and LOOKUP rights on his home directory to his manager *pat*, *terry* might type

% fs setacl -dir $HOME -acl pat rl

or if *terry* was in his home directory he might type

% fs setacl -dir . -acl pat rl

Both of the previous examples are acceptable short forms for

% fs setacl -dir /afs/tr/usr/pat -acl pat rl

2.3. Getting Help in AFS

AFS on-line help consists of syntax and alias messages, command descriptions, and command suite manual pages.

2.3.1. Displaying Command Syntax and Aliases

If you know a command suite and an operation code, but you cannot remember the syntax or alias(es), use the **help** operation code to list them. For example, typing

% **fs help listacl**

displays

```
fs la: list access control list (alias for listacl)
Usage: fs la [-path <dir/file path>+] [-help ]
```

You can also display the syntax of a command by using the **-help** flag, available with most AFS commands. For example, typing

% **fs setacl -help**

displays

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

2.3.2. Displaying Operation Code Descriptions

If you want to list a short description about the operation codes in a command suite, use the **help** operation code alone with the command suite. Typing **fs help** displays a short description of every operation code for the **fs** command suite.

If you cannot remember the name of a particular operation code but you think it may be associated with a certain command suite, use the **apropos** operation code with that command suite and supply a string relevant to the task you want to perform. If the string contains spaces, surround it with quotes.

For example, if you want to set the contents of an ACL but you cannot remember which **fs** command performed this function, you might type

% **fs apropos set**

and the following would be displayed:

```
debug: set debugging info
monitor: set cache monitor host address
setacl: set access control list
setcachesize: set cache size
setcell: set cell status
setquota: set volume quota
setserverprefs: set file server ranks
setvol: set volume status
sysname: get/set sysname (i.e. @sys) value
```

If you type an **apropos** operation code and a help string with no matches, like

`%` **fs apropos wrong**

the following message would appear:

```
Sorry, no commands found
```

Note: The help string must be enclosed in double quotes (" ") if it contains spaces.

2.3.3. Accessing Manual Pages

Manual pages are available for AFS command suites. Manual pages list the operation codes for the command suite, describe which AFS server or client process the command suite provides an interface to, and detail the syntax of each operation code.

At the command prompt, type

`%` **man** <*command suite*>

For example, typing

`%` **man fs | more**

displays

```
30 April 1990                                                  FS(1)

NAME
     fs - suite of commands for contacting the AFS File
     Server and configuring the Cache Manager.

SYNOPSIS
     fs <option>

     Each of the the following operation codes is a valid
     <option> and has its own set of associated arguments.
     See the "OPTIONS" section below for syntax
     definitions.

     apropos - search by help text
     cachesize - set cache size
     checkservers - check functioning servers
     checkvolumes - check volume/name mappings
     cleanacl - clean up access control list
     copyacl - copy access control list
     debug - set debugging info
     diskfree - show server disk space usage
     examine - display volume status
     exportafs - enable/disable exporters to AFS
     flush - flush file from cache
     flushvolume - flush all data in volume
     getcacheparms - get cache usage info
     getcellstatus - get cell status
     help - get help on commands
     listacl - list access control list
     listcells - list configured cells . . .
```

Using AFS

Replace this page with the tab separator for

Using AFS

3. Using AFS – the Fundamentals

This section contains instructions for performing four basic AFS tasks: entering AFS, exiting AFS, accessing directories and files in AFS, and changing your password.

It includes the following sections:

3.1. Entering AFS

Entering AFS is a two-step process: logging in to the system and becoming authenticated with AFS. When you log in, you establish your system identity; when you authenticate, you prove your identity to AFS and establish yourself as an authenticated AFS user.

When you authenticate with AFS, you are given a token which your Cache Manager uses on your behalf to prove your status as an authenticated user. Users who are not authenticated (that is, do not have any tokens) have very limited access to AFS directories and files.

Sections 3.1.1 and 3.1.2 provide general information on logging in and authentication; for specific instructions, see page 3-5.

3.1.1. Logging in

Cells using the AFS version of **login** combine logging in and authenticating into a one-step procedure. Cells using their own **login** program use a two-step login procedure. Your system administrator can tell you which type of login your cell uses or you can check for tokens after logging in. He or she can also give you information on any differences between your login procedure and the two standard login methods described here.

3.1.1.1. AFS Login

If your cell uses the AFS version of the **login** program, you are automatically authenticated in your local cell when you correctly type your user name at the **login:** prompt and your password at the **Password:** prompt. You can verify that you are authenticated by issuing the **tokens** command. (See section 3.1.2.5 for information on listing tokens.)

3.1.1.2. Two-Step Login

If your cell does not use AFS **login**, you will need to perform a two-step procedure:

1. **Log in:** Log in using the standard UNIX login.
2. **Authentication:** Issue the **pagsh** command to create a holder for your token (called a PAG for process authentication group) and prevent root users on your machine from being able to use your tokens. Issue the **klog** command to authenticate.

Note: If your cell uses a two-step login procedure, you may use different passwords for logging in and authentication because you are asked to provide a password for each step. Talk with your system administrator.

3.1.2. Authentication

To work effectively in the AFS file space, you must ***authenticate*** with AFS. When you do, your Cache Manager is given a ***token*** as proof of your authenticated status. Your Cache Manager uses your token when requesting services from AFS servers; in turn, the servers accept the token as evidence of your authentication. If you do not have a token, AFS considers you to be an anonymous user (a member of the group *system:anyuser*) and your access to AFS file space is severely limited.

3.1.2.1. Getting Tokens

The **klog** command is used to get tokens (i.e., authenticate). If your cell uses the AFS login program, **klog** is issued automatically as part of that login. If your cell uses the two-step login, you must issue the **klog** command after you log in on your local machine. If you want to get new tokens, you can always use the **klog** command. See section 3.1.1 for information on logging in.

A token is only useful in the cell from which it was issued. Therefore, although you may have a token for your local cell, that token does not give you any special status in foreign cells. Unless you are authenticated in a foreign cell, that cell considers you to be an anonymous user (i.e., a member of *system:anyuser*).

3.1.2.2. Getting Tokens for Foreign Cells

You can have tokens in use simultaneously for your local cell and for any foreign cells in which you have an account. To obtain tokens in foreign cells, use the **klog** command with the **-cell** argument.

If you have an account in a foreign cell, it is simple to access the files in the cell; just authenticate in each cell (using **klog**) and then access the files in those cells by using the standard pathnames. This is much simpler than logging in and out of machines in foreign cells. (See section 3.3 for more information on accessing directories and files.)

3.1.2.3. Limits on Token Acquisition

You can get a maximum of one token per cell per client machine per token holder (PAG). For example, if you telnet to two different machines, you get a token for each machine; however, neither machine can use the token of the other.

If you are using the two-step login procedure and you fail to issue the **pagsh** command before executing **klog**, your token will be associated with your UNIX UID (user identification). This association can cause two problems:

- The user "root" can execute the UNIX **su** command to assume your UNIX UID and then use your tokens.

- You may be unable to execute certain commands that run in the "set user ID mode" like **lp** or **lpr**.

If you already have a token for a particular cell, issuing **klog** will overwrite the old token with a new one. This may be useful if the token is nearing the end of its effective lifetime and you want to run an extended job in the background. See section 3.1.2.6 for information on token lifetimes.

3.1.2.4. Getting Tokens as Another User

It is possible to authenticate as another user if you know the user's password and the user has an AFS account (and, of course, you have that user's permission to use his or her account). If you authenticate as another user, you remain logged in as yourself, but you are authenticated as the other user. Any current token you may have in that user's cell is destroyed because the Cache Manager on your machine can only hold one token per authentication session in a given cell.

3.1.2.5. Listing Tokens

The **tokens** command allows you to list your tokens. The output of the **tokens** command lists the AFS UID (user identification) of the token owner, the cell that the token is valid in, and the expiration date of the token.

After authentication, you may want to list your tokens. And if you think one or more tokens may have expired, issue the **tokens** command to check.

3.1.2.6. Token Lifetimes

Tokens have a limited lifetime. You can determine the expiration date of a token by issuing the **tokens** command. If you are ever unable to perform a task that you normally can, you should check to see if the appropriate token is still valid. If a token expires while you are running a background job, that job may fail if it needs a token for authentication purposes.

The default lifetime of your token is set by your system administrators. If you want a token with a lifetime that is less than the default lifetime, you may specify that lifetime with the **-lifetime** argument on the **klog** command. You may <u>not</u> specify a lifetime longer than the default. The page on **klog** in Appendix A explains how to use the **-lifetime** argument. It also discusses the settings, defined by your system administrator, that limit token lifetime. If you are interested, ask your system administrator about the settings in your cell.

To Log in – login

Provide your user name at the initial **login:** prompt and your login password at the **Password:** prompt as shown in the following example. (Your password will not appear on the screen.)

```
login: <user name>
Password: <password>
```

If the "`AFS (R) 3.3 login`" banner appears after you type your password, your cell uses the AFS **login**. You are now an authenticated AFS user. If the AFS banner does not appear, then the **login** program did not authenticate you and you should use the **pagsh** and **klog** commands.

To Authenticate with AFS – pagsh and klog

If your cell is not using AFS **login**, you will need to authenticate after login. Use the **pagsh** and **klog** commands to authenticate. (Your password will not appear on the screen.)

At the command shell prompt, type

```
% pagsh
% klog
password: <your AFS password>
```

When your command shell prompt returns, you are an authenticated AFS user. You can use the **tokens** command to make sure that you are authenticated.

Note: Appendix A describes the full syntax of the **klog** command and its uses. The **pagsh** command has no arguments or flags; however, Appendix A provides additional information on this command.

To Check Tokens – tokens

Use the **tokens** command to list your tokens. If you are not authenticated, no tokens are listed. If you are authenticated, **tokens** lists the following information for each token: the AFS UID associated with the token, the expiration date of the token, and the name of the cell in which the token was issued.

At the command shell prompt, type:

% tokens

If you have no tokens, the following output appears:

```
Tokens held by the Cache Manager:

    --End of list--
```

If you have one or more tokens, your output should look like this:

```
Tokens held by the Cache Manager:

User’s (AFS ID 1022) tokens for afs@comp.com [Expires Jan 3 14:35]
User’s (AFS ID 9554) tokens for afs@stateu.edu [Expires Jan 4 1:02]
   --End of list--
```

In the above example, the tokens listed for AFS UID 1022 expire on January 3 at 2:35 p.m for cell *comp.com*. For AFS UID 9554 the tokens expire on January 4 at 1:02 a.m. for cell *stateu.edu*.

Example: Authenticating as Yourself in a Local Cell

A user, *terry*, cannot save a file. He uses the **tokens** command and finds that his tokens have expired. He decides to re-authenticate in his local cell, so he types:

```
% klog
Password: <terry’s password>
```

When the command shell prompt reappears, *terry* is authenticated; to make sure he is authenticated, he types

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

User’s (AFS ID 1022) tokens for afs@comp.com [Expires Jan 3 14:35]
   --End of list--
```

Any tokens *terry* may have had for previous authentications in his local cell are destroyed because the Cache Manager on *terry*’s machine allows only one token per cell per token holder (PAG).

Example: Authenticating as a Another User

In this example, *terry* wants to authenticate in his local cell as another user, *indira*. At the command shell prompt, *terry* types

```
% klog indira
Password: <indira’s password>
% tokens
Tokens held by the Cache Manager:

User’s (AFS ID 1053) tokens for afs@comp.com [Expires Jan 3 14:55]
   --End of list--
```

The token for *terry*’s previous authentication in his local cell is destroyed because the Cache Manager on *terry*’s machine allows only one token per authentication session per cell.

Example: Authenticating in a Foreign Cell

Then *terry* decides to authenticate in another cell where he has an account, *stateu.edu*. In *stateu.edu*, terry's user name is *ts09*, so he types

```
% klog -principal ts09 -cell stateu.edu
Password: <ts09_password>
% tokens
Tokens held by the Cache Manager:

User's (AFS ID 9554) tokens for afs@stateu.edu [Expires Jan 3 15:37]
User's (AFS ID 1022) tokens for afs@comp.com [Expires Jan 3 14:35]
   --End of list--
```

Now *terry* has a token in his local cell and in the cell *stateu.edu*.

3.1.3. Limits on Failed Authentication Attempts

Your system administrator may set a limit to the number of times that you can incorrectly enter your password during consecutive authentication attempts (whether using AFS **login** or **klog**). Should you exceed this number, you may be locked out of your account for a period of time. The purpose of this limit is to keep unauthorized users from trying to guess your password in an attempt to break into your account.

To find out if your user account has such a limit, issue the **kas examine** command or talk to your system administrator. If your account does have such a limit, you should type your password very carefully to avoid exceeding the limit.

If you receive the following message during authentication you will know that the authentication failure limit has been exceeded for your account.

```
Unable to authenticate to AFS because ID is locked - see your
system admin
```

You should contact your system administrator and explain the circumstances of the failure. If you believe someone may be attempting to break into your account, you should explain this to your system administrator. Your system administrator can unlock your account or inform you of when it should be available again.

To list your authentication limit and lockout time – kas examine

Issue the **kas examine** command to determine if there is a limit on the number of unsuccessful authentication attempts for your user account and any associated lockout time:

```
% kas examine <user name>
password for user name: <user name's AFS password>
User data for user name
 key (0) cksum is 3414844392,  last cpw: Thu Dec 23 16:05:44 1993
 password will expire:  Fri Jul 22 20:44:36 1994
 5 consecutive unsuccessful authentications are permitted.
 The lock time for this user is 25.5 minutes.
 User is not locked.
 entry never expires. Max ticket lifetime 100.00 hours.
 last mod on Thu Jul 1 08:22:29 1993 by admin
```

Line three of the output of the command indicates the maximum number of unsuccessful authentication attempts that are allowed before you will be locked out of your account. Line four indicates the amount of time that you will be locked out of your account should the number of failed authentication attempts be exceeded (either by you or someone else).

3.2. Exiting AFS

Just as accessing AFS is a two-step process (logging in and authenticating), exiting AFS is a two-step process: unauthenticating and logging out. You need to ***unauthenticate*** because logging out does not necessarily destroy your existing tokens; whereas unauthenticating specifically destroys tokens. The **unlog** command is used to unauthenticate, while the command used to log out is system dependent.

While you should unauthenticate before logging out, you can use the **unlog** command any time you feel the need to unauthenticate. For example, you might want to unauthenticate before leaving your workstation unattended so that other users cannot walk up to your workstation and access AFS files under your identity. When you come back to your workstation to resume work, you can re-authenticate using **klog**.

You should not **unlog** if you have processes running in the background, even if you log out. Most processes require the use of your token to prove their authenticity to AFS servers.

If you have tokens from different cells, you can selectively unauthenticate from some cells by using the **-cell** argument with the **unlog** command. You may want to do this if you plan no additional work in the corresponding cell.

To Unauthenticate with all cells – unlog

When you want to destroy all of your current tokens, use the **unlog** command by typing

% **unlog**

To ensure that your tokens were destroyed, use the **tokens** command. You should see a display similar to the example below:

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

   --End of list--
```

To Unauthenticate from some cells – unlog with the -cell flag

At the command shell prompt, type **unlog** and the name of each cell for which you want to destroy your token. It is best to provide the full name of the cell (such as *cell.school.edu* or *business.com*). At the command shell prompt type:

% **unlog -cell** <*cell name*>+

where *cell name* is used to list one or more specific cells to unauthenticate in; without this argument, all tokens are destroyed.

Example: Unauthenticating from a Specific Cell

In the following example, a user has tokens in both the *accounting* and *marketing* cells at her company. She wants to destroy the accounting token (for cell *accnt.biz.com*) but keep the marketing token (for cell *mrkt.biz.com*).

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

User's (AFS ID 35) tokens for afs@accnt.biz.com [Expires Jan 3 16:25]
User's (AFS ID 674) tokens for afs@mrkt.biz.com [Expires Jan 5 9:57]
   --End of list--

% unlog accnt.biz.com

% tokens
Tokens held by the Cache Manager:

User's (AFS ID 35) tokens for afs@accnt.biz.com [Expires Jan 3 16:25]
   --End of list--
```

To Log Out – regular logout command

After you have unauthenticated, logging out is just a matter of issuing your regular logout command. At the command shell prompt, type your log out command:

% **logout**

or

% **exit**

or

% **<Ctrl-D>**

3.3. Accessing Directories and Files in AFS

Once you have logged in and are authenticated, you can access files in AFS just as you do in the UNIX file system. The only difference is that you have the potential of accessing many more files because you are not limited to those on your local disk. Just as in UNIX, you can only access those files for which you have permission; though, in AFS, permission is set using access control lists (ACLs). (See chapter 5 for more information on ACLs.)

3.3.1. AFS Pathnames

AFS pathnames look very similar to standard UNIX file system names. The main difference is that every AFS pathname begins with */afs*. The */afs* is the hook that connects every cell's file space into one uniform file space.

The second element in AFS pathnames is generally a cell's name, to indicate the cell that contains the file. For example, the Transarc Corporation cell is called *transarc.com* and the pathname of every file in its file space begins with */afs/transarc.com*. Some cell names have matching shortened forms (such as *tr* and *transarc* for *transarc.com* and *andrew* for *andrew.cmu.edu*). The shortened forms may be used in place of the complete cell names in most cases. Talk with your system administrator to see if there is an abbreviated name in use by your cell.

The rest of the pathname depends on how your cell's administrators organized the local file space. If you want to learn how a cell has organized its file space, **cd** to the */afs/<cellname>* directory for that cell and use the **ls** command to see the directories at the third level.

To access directories and files in your local cell you must

- Specify the pathname.
- Have permission to access the desired files based on the directory's access control list.

Example: Viewing Another User's Directory

The user *terry* wants to look for a file belonging to another user, *pat*. At the command prompt, *terry* types the **ls** command and the pathname for *pat*'s *public* directory in the *comp.com* cell.

```
% ls /afs/comp.com/usr/pat/public
doc/                    directions/
guide/                  jokes/
library/
```

3.3.2. Accessing Foreign Cells

You can share files not only with people on your own local cell, but with anyone using AFS on your network, regardless of geographical location. There are two criteria for sharing files in foreign cells:

- The foreign cell must be on your Cache Manager's foreign cell list (set by your cell's system administrator).
- You must have access to the file through the directory's access control list (set by the owner of the directory).

If you meet these two requirements, you can access directories and files in foreign cells by specifying the pathname.

3.3.2.1. Your Cache Manager's Foreign Cell List

Your system administrators choose the cells your workstation can reach by including their names on your Cache Manager's foreign cell list. If a foreign cell is not listed, the Cache Manager will not be able to reach that cell. (See chapter 4.4 to learn how to determine which foreign cells you can access.)

3.3.2.2. Foreign Cell Users on Access Control Lists

To gain access to a file in a foreign cell, you must have permission on the access control list of the appropriate directory. This permission can be set up in one of two ways:

- The owner of the directory can give access to the *system:anyuser* group, and as a member of that group you gain access. The disadvantage to this approach is that every other user can access the directory in the same way.
- The system administrator in the foreign cell can set up an account for you, and the owner of the directory can put your local user name on the access control list of the directory. This is a much safer method of providing access to foreign users.

Whether or not you have an account in a foreign cell, you are automatically a member of *system:anyuser*. If you have an account in a foreign cell, you must authenticate (**klog**) in that cell before you can take advantage of the account's privileges.

See chapter 5 for information on directory and file protection. See section 3.1.2.2 for information on authenticating in a foreign cell.

3.4. Changing Your Password

In cells that use the AFS version of **login**, the AFS password is used for both logging in and authenticating. In this case, you can use a single command, **kpasswd**, to change your AFS password.

Cells that use a two-step login procedure require a UNIX password and an AFS password. (The two passwords may be the same or different, at your discretion.) In this case, use **kpasswd** to change you AFS password and **/bin/passwd** to change your UNIX password.

If your system administrator has defined a valid lifetime for your password, you should change your password prior to its expiration. You can determine the expiration date of your password by issuing the **kas examine** command. (The system administrator may also configure your login mechanism so that the number of days until password expiration is displayed after a successful authentication; this mechanism may also prompt you to change your password when the password expiration date is close.)

Of course, you can always change your password prior to the expiration of the password's lifetime. However, your system administrator may have established a minimum time between consecutive password changes; if this is the case, you will be prevented from changing your password (after a previous change) until the minimum time has elapsed. If you attempt to change a password before this minimum time has elapsed, you will receive the following message:

```
kpasswd:  password was not changed because you changed it too
recently; see your system administrator
```

When you change your password, you should keep the following in mind:

- Your system administrator may have established certain rules for creating passwords in your cell. For example, passwords in your cell may need to be eight characters or longer. Talk with your system administrator to determine if any password creation rules exist in your cell. If you fail to follow the creation rules, you may be forced to create a new password.
- When you change your password, you may be prevented from reusing any of your last twenty passwords. Talk with your system administrator to determine whether or not this restriction applies. If you attempt to reuse any of your last twenty passwords, you will receive the following message:

  ```
  kpasswd:  Password was not changed because it
  seems like a reused password
  ```

AFS does not impose any additional restrictions on passwords. Follow the conventions for your site.

To list password expiration date – kas examine

Issue the **kas examine** command to list the expiration date of your password:

```
% kas examine <user name>
password for user name: <user name's AFS password>
User data for user name
 key (0) cksum is 3414844392,  last cpw: Thu Dec 23 16:05:44 1993
 password will expire:  Fri Jul 22 20:44:36 1994
 5 consecutive unsuccessful authentications are permitted.
 The lock time for this user is 25.5 minutes.
 User is not locked.
 entry never expires. Max ticket lifetime 100.00 hours.
 last mod on Thu Jul 1 08:22:29 1993 by admin
```

Line two of the output lists the expiration date, if any, of your password.

To Change Your AFS Password – kpasswd

At the command shell prompt, type **kpasswd**. You will be prompted to enter your old password and new password and to confirm your new password. (Your passwords will not appear on the screen.)

```
% kpasswd
Old password: <old_password>
New password (RETURN to abort): <new_password>
Retype new password: <new_password>
```

To Change Your UNIX Password – /bin/passwd

At the command shell prompt, type **/bin/passwd**. You will be prompted to enter your old password and new password and to confirm your new password. (Your passwords will not appear on the screen.)

```
% /bin/passwd
Changing password for user name.
Old password: <old_password>
New password: <new_password>
Retype new passwd: <new_password>
```

Listing Information

Replace this page with the tab separator for

Listing Information

4. Listing Information about AFS

This chapter contains instructions for obtaining information that will help you use AFS more effectively. It includes instructions on

- Checking volume quota.
- Finding the file server location of a file or directory.
- Listing the status of file server machines.
- Determining access to foreign cells.

It includes the following sections:

4.1. Checking a Directory's Volume Quota

The files in your home directory are generally stored together in a single volume. (See section 1.2.3 for more information on volumes.) To divide your cell's available disk space as fairly as possible, your system administrators impose a size limit, or ***quota***, on each volume. You cannot store more data in a volume than its quota allows. If you are close to exceeding a volume's quota, you may be unable to save the changes you have made to a file stored in that volume. Similarly, you may sometimes be unable to save a file because the disk partition that contains a volume is full, even though your volume is not close to exceeding its quota.

You should check the quota of your home volume periodically to make sure you have adequate space. Also, if you encounter problems saving a file, check the quota of the volume in which the file is stored. You can check volume quota using any of the following **fs** commands:

- The **fs quota** command lists the percentage of the volume quota used.
- Both the **fs listquota** and **fs examine** commands list the volume name, its maximum size, and its current size. In addition
  - The **fs listquota** command lists the percentage used of both the volume and the partition.
  - The **fs examine** command lists the partition's maximum size, its current size, and any messages associated with the volume.

To Check Percentage of Quota Used – fs quota

The **fs quota** command lists the percentage of the quota currently used on the volume that contains a specified directory or file. At the command shell prompt, type

% fs quota [-path <*dir/file path*>$^{+}$**]**

where *dir/file path* specifies the UNIX pathname of each directory whose volume quota information is to be displayed. You can specify multiple directories or files. If no *dir/file path* is specified, the default is the current directory.

For example, if the user *terry* wants to know the percentage of quota used for the volumes that contain two directories – his home directory and the home directory of *pat*, another user – he would type

% fs quota -path /afs/cell/usr/terry /afs/cell/usr/pat
```
34% of quota used.
85% of quota used.
```

To List Quota and General Information on a Volume – fs listquota

Use the **fs listquota** command to display

- The name of the volume (usually *user.username* for a user volume).
- The size of the volume quota, expressed as a number of kilobyte blocks. Each block is 1024 bytes; therefore, *1024* indicates a one megabyte quota.
- The number of kilobyte blocks currently used.
- The percentage of the quota used.
- The percentage of available space used on the disk partition housing the volume. This is not directly related to the amount of the volume quota used.

At the command shell prompt, type

% **fs listquota [-path** <*dir/file path*>**+]**

where *dir/file path* specifies the UNIX pathname of each directory or file whose volume quota and other information is to be displayed. You can specify multiple directories or files. If no *dir/file path* is specified, the default is the current directory.

For example, if *terry* wants to display quota information about the volume that houses his home directory, he would type

% **fs listquota ~terry**
```
Volume Name     Quota    Used    % Used   Partition
user.terry      10000    3400       34%         86%
```

Note: If *terry* is already in a directory contained in his home volume, he could simply type **fs listquota** to get the same information.

To List General Information on a Volume – fs examine

The **fs examine** command lists information about the volume that contains a specified directory or file, including:

- The ID number of the volume (abbreviated in the output as "vid").
- The volume name.
- The volume's quota and current size, in kilobyte blocks.
- The number of kilobyte blocks available on the disk partition housing the volume and the total size of that partition.
- The current "off-line" message associated with the volume, if any, as set by a system administrator.
- The current "message of the day" associated with the volume, if any, as set by a system administrator.

At the command shell prompt, type

% fs examine [-path <*dir/file path*>**+]**

where *dir/file path* specifies the UNIX pathname of each directory or file whose volume information is to be displayed. You can specify multiple directories or files. If no *dir/file path* is specified, the default is the current directory.

For example, if *terry* wants to display information about the volume that houses the current directory, he would type

```
% fs examine
Volume status for vid = 536871122 named user.terry
Current disk quota is 10000
Current blocks used are 5745
The partition has 1593 blocks available out of 99162
```

4.2. Locating Files and Directories

Normally, you need not be concerned about which file server machine stores a file or directory. Given the pathname to a file, the Cache Manager on your client machine accesses the server machine for you automatically (see section 1).

However, it is sometimes useful to know the location of a file or directory. For example, if a file server that houses a file you are working on goes down, you must wait until the server comes back up to save that file. And because your system administrators can move volumes that house specific files to different file server machines, you may want to occasionally determine the location of the files and directories that you commonly use. To learn the current location of a file or directory, use the **fs whereis** command.

To List the Location of a File or Directory – fs whereis

The **fs whereis** command lists the file server machine on which a file or directory is stored. If the output of the command contains more than one machine it means that the volume containing that file or directory is copied and exists at each listed site. In general, popular files that do not change are often copied so that if one of the servers containing those files goes down the files are still available on other file servers. Volumes that contain binary files, such as those for text editors, are often replicated; user volumes are rarely replicated.

At the command shell prompt, type

% **fs whereis [-path** <*dir/file path*>$^+$**]**

where *dir/file path* specifies the UNIX pathname of each file or directory whose location is desired. If this argument is omitted, the location of the current directory is returned.

For example, if the user *terry*, from the cell *comp.com*, wants to see the location of the files in his home directory, he would type

```
% fs whereis /afs/comp.com/usr/terry
File /afs/comp.com/usr/terry is on host fs2.comp.com
```

If terry wanted to check the locations of his and *pat*'s home directories, he would type

```
% fs whereis ~terry ~pat
File /afs/comp.com/usr/terry is on host fs2.comp.com
File /afs/comp.com/usr/pat is on host fs3.comp.com
```

4.3. Checking the Status of File Server Machines

Sometimes one or more file server machines in your cell will become unavailable due to hardware problems, software problems, or routine maintenance. In these cases, you will be unable to access files stored on those machines or save any changes you have made to files that are stored on those machines until the machines come back into service. (Remember, your Cache Manager may have copies of your files stored locally that you can still work on.)

The **fs checkservers** commands allows you to check if the file server machines in your cell (and other cells) are operative. Depending on how the command is issued, you can control which file server machines are checked; to check the file server machines in

- Your cell only, type **fs checkservers** without any arguments.
- Every cell that your Cache Manager has been in contact with, add the **-all** flag to **fs checkservers**.
- A particular foreign cell only, type the complete cell name following **fs checkservers**.

Note: The **fs checkservers** command may take a while to complete because it waits a few minutes for each machine that does not respond immediately. If you want your command shell prompt to return immediately, run **fs checkservers** in the background using the "&" symbol (for example, **fs checkservers -all &**).

To Check File Server Machine Status – fs checkservers

The **fs checkservers** command allows you to determine the status of the file server machines. At the command shell prompt, type

% **fs checkservers [-cell** <*cell to check*>**] [-all]**

If all contacted servers are running, you will see the following output:

```
All servers are running.
```

Otherwise, the output lists which machines are unavailable:

```
These servers are still down:
     fs1.comp.com
     fs3.comp.com
```

For example, if *terry* wants to check the file server machines in his cell only, he would type

```
% fs checkservers &
All servers are running.
```

If *terry* wants to check the file server machines in every cell which his Cache Manager contacted, he would type

```
% fs checkservers -all &
These servers are still down:
      fs1.comp.com
      server7.school.edu
```

Note: By combining **fs checkservers** and **fs whereis** (see section 4.2) you can find out if a volume you are using is on an unavailable file server machine. If a text file is stored in a volume kept on a server that is down, you will probably be unable to access it.

4.4. Determining Access to Foreign Cells

The Cache Manager on your machine maintains a list of foreign cells and the servers in those cells. You can only authenticate in a foreign cell if it is in your Cache Manager's list (and, of course, if you have an account in that cell). Use the **fs listcells** command to list the foreign cells your Cache Manager recognizes.

To List Foreign Cells – fs listcells

The **fs listcells** command has no arguments or flags. However, because it may take a while to complete you may want to run it in the background using the "&" symbol.

At the command shell prompt, type:

% fs listcells &

```
Cell athena.mit.edu on hosts
     MAEANDER.MIT.EDU
     ORF.MIT.EDU.
Cell testers.athena.mit.edu on hosts
     DOT.MIT.EDU.
Cell alefnull.mit.edu on hosts
     ALEFNULL.MIT.EDU.
Cell aix.ifs.umich.edu on hosts
     35.1.37.34.
Cell nih.gov on hosts
     vice1.alw.nih.gov
     vice3.alw.nih.gov.
Cell mtxinu.com on hosts
     kineo.mtxinu.COM.
```

Protecting Directories

Replace this page with the tab separator for

Protecting Directories

5. Protecting Your Directories and Files

This chapter explains how to protect the files and directories in AFS by defining access rights. It includes the following sections:

5.1. Access Control Lists

Every directory has its own ***access control list*** (ACL) that is used to define which users can access the directory and its files and in what manner. Each ACL can have up to 20 user and/or group entries (an entry lists the user or group and the corresponding access rights for that user/group).

You can grant any combination of rights to a user/group or specifically deny access to that user/group on an ACL. A user entry applies to the corresponding user only; a group entry applies to all of the members of a group. Section 5.2 provides information on the type of rights available on an ACL. Sections 5.4 and 5.5 explain how to examine and change ACLs, respectively.

A group can consist of both users and machines. If a user is a member of a group, he or she has all of the rights of that group. If a user is logged into a machine that is a member of a group, he or she has all of the rights of that group. Both regular users and system administrators can create groups for use on ACLs. To learn more about group creation, see chapter 6.

AFS automatically defines three system groups in each cell; these groups can, among other things, be used to give access to large numbers of users at once. System-defined groups can be added to an ACL just like any other group. To learn more about the system-defined groups, see section 5.3.

Although AFS relies on ACLs for basic file and directory protection, it also uses the UNIX mode bits to a limited extent. See section 5.7 to learn more about UNIX mode bits and their meaning in AFS.

5.1.1. Directory Level Access Control

Defining access at the directory level has several consequences:

- Files inherit the access rights associated with their parent directory. If you move a file to a new directory, the file gets the access rights of its new directory. (So if you want a file to be as protected as in the original directory, make sure that the new directory has an equally restrictive ACL.) If you change the ACL on a directory, the protection on all of its files also changes.
- When you create a subdirectory, it inherits the current access rights of its parent directory. You can then set the subdirectory's ACL to be different from its parent's. However, if you make the ACL on a parent directory more restrictive than on a subdirectory, you will make it very difficult for someone to reach the subdirectory even though they may still have rights to it. Specifically, a person must have the LOOKUP right (defined in section 5.2) on the parent directory to reach its subdirectories.

As a general rule, it makes sense to have fairly liberal access rights on your home directory and then create subdirectories with more restrictive rights if you want to protect certain files more carefully.

5.2. AFS Access Rights

There are seven standard access rights available on ACLs. To give a user or group a specific set of rights, you must create an entry for that user or group with the corresponding rights. The rights apply to the directory on which they are placed. Each right has an accepted single-character abbreviation, which appears after the complete name in the following definitions.

Note: Users need the LOOKUP right before they can access files or directories in any other way.

Below is a list of the seven rights and an explanation of the privileges granted by each right:

- The LOOKUP (**l**) right allows the possessor to
  - Issue the UNIX **ls** command to list the names of the files and subdirectories in the directory. It does not allow the possessor to list the contents of a subdirectory in the directory unless he or she also has LOOKUP on that subdirectory's ACL.
  - Issue the UNIX **ls -ld** command to long list the directory element itself.
  - Examine the ACL for the directory.
  - Access the directory's subdirectories, which are protected by their own ACLs.
- The INSERT (**i**) right allows the possessor to
  - Add new files to the directory, either by creating new files or by copying existing files.
  - Create new subdirectories.
- The DELETE (**d**) right allows the possessor to
  - Remove files from the directory or move them into other directories (where the user also has the INSERT right).
  - Remove subdirectories (from directories where the user also has the INSERT right).
- The ADMINISTER (**a**) right allows the possessor to change the ACL for the directory. Users always have this right on their home directory and its subdirectories.
- The READ (**r**) right allows the possessor to read the contents of the files in the directory (that is, read the data in the file).
- The WRITE (**w**) right allows the possessor to modify the contents of the files in the directory and to change their UNIX mode bits with the UNIX **chmod** command.

- The LOCK (**k**) right allows the possessor to run programs that need to place advisory locks on files in the directory ("flock" files). Most programs do not need to place advisory locks on files; however, if you get a message stating that an application cannot lock a file, it may be because you do not have LOCK permission on the directory.

5.2.1. Shorthand Forms for Common Combinations of Rights

You are free to combine the rights in any way in an ACL entry, but certain combinations make more sense than others. Four of the more common combinations have corresponding shorthand forms. When defining ACLs, you can type either the individual letters corresponding to the rights or the shorthand forms, which are

- **write** – all rights except ADMINISTER (**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.

5.2.2. Normal and Negative Rights

AFS allows you to grant access by setting ***normal*** rights and to deny access by setting ***negative*** rights. You can either grant or deny any user or group any combination of the seven access rights defined previously.

When you set normal rights, you are giving permission for the user or group to perform the specified actions. See section 5.5.1 to learn how to grant access by adding users to the `Normal rights` list.

When you set negative rights, you are explicitly denying the user or group permission to perform the specified actions by adding them to the `Negative rights` list. For example, if you want to give READ rights to all but one person in a group, you would put the group on the `Normal rights` list and the person you wanted to deny on the `Negative rights` list. See section 5.5.2 to learn how to deny access and section 5.5.3 to learn how to restore access you previously denied.

5.2.3. Other Variable Rights

Eight additional access rights, represented by the uppercase letters **A** through **H**, are available for use on ACLs. These letters have no standard meanings and are usually ignored by the AFS server processes. Your AFS system administrator may assign meanings to them and place them on ACLs to control access in new ways. If you encounter these rights on an ACL, ask your AFS system administrator about their meaning. You can use the **fs listacl** and **fs setacl** commands to inspect, set, and remove the eight variable rights in the same way as the seven standard rights.

5.3. Using System Groups on ACLs

AFS defines three ***system groups*** that you can put on your ACLs. System groups enable you to grant access to a wide range of people simultaneously. You do not have control over the membership of these groups, however, so consider carefully what kind of rights you wish to give them. (You do control the membership of the groups you own; see chapter 6 to learn how to create your own groups.)

The three system-defined groups are

- ***system:anyuser*** – This group includes anyone who can gain access to your cell, including people who have logged into a workstation locally but have not authenticated to AFS, people who have used telnet from outside the cell, or people who have contacted the local file server from a workstation in another AFS cell.
- ***system:authuser*** – This group includes anyone who is currently authenticated in your cell. This means any user who has an AFS account in your cell and is authenticated. See chapter 3.1.2 to learn more about authentication.
- ***system:administrators*** – This group includes only the few people in your cell designated as system administrators and thus authorized to operate and administer AFS. There is no reason to specifically grant or deny access to this group since its members automatically have ADMINISTER rights on all directories.

5.3.1. Allowing Access to Subdirectories

A user must have the LOOKUP right on a directory to access its subdirectories in any way. Even if a user has extensive rights on a subdirectory, he will not be able to access it if he does not have the LOOKUP right to the parent directory.

You can grant the LOOKUP right in one of three ways: grant it to a system group (*system:anyuser* or *system:authuser*), grant it to each authorized user individually, or grant it to a group of authorized users you define yourself (see chapter 6).

Granting rights to *system:anyuser* is the easiest option and is generally safe because the LOOKUP right only allows someone to list the contents of the directory, not to read the files in it. If you want to be slightly more restrictive, substitute *system:authuser* for *system:anyuser* and only locally authenticated users will be able to list the contents of the subdirectory.

5.3.2. Allowing Access to Services

The *system:anyuser* group may need certain rights on some of your directories so that process daemons can provide services such as printing and mail delivery. For example, a printing daemon may need READ rights on certain directories (in addition to LOOKUP rights) to print the contents of files; or a mail daemon may need INSERT rights to be able to give you new pieces of mail. Your system administrator has probably already granted the necessary rights on the appropriate directories when they were created and should be able to provide you with more information.

Your home directory's ACL may be set to grant READ and LOOKUP rights to *system:anyuser* by default. This means that any user who is logged in can read (**r**) the files in your directory and display status (**l**) about those files. Users cannot list and display the statuses of files in your subdirectories unless they have **rl** rights on these subdirectories as well as **l** rights on your home directory. You can view the ACL on your home directory with the **fs listacl** command described in section 5.4.

5.3.3. Allowing Access to Users from Foreign Cells

The only way to grant access to users from foreign cells who do not have an account in your cell is to put *system:anyuser* on an ACL. Remember that *system:anyuser* includes <u>everyone</u> who can reach your cell, not just authenticated users from foreign cells.

5.4. Listing an ACL

The **fs listacl** command is used to view a directory's ACL. You should view a directory's ACL before making changes to it. The ACL of a user's home directory usually grants READ and LOOKUP rights to *system:anyuser* and all rights (**rlidwka**) to the directory's owner. Section 5.3 explains system-defined groups (such as *system:anyuser*) and why they should appear on certain ACLs.

To List a Directory's ACL – fs listacl

To list an ACL, type

`%` **fs listacl [-path** *<dir/file path>*⁺**]**

where *dir/file path* specifies each directory or file whose ACL is to be listed. If you specify a filename, the parent directory of the file is used. The default is the current directory.

The output of the **fs listacl** command lists user names or group names and their associated access rights (represented by the single letters defined in section 5.2). `Normal rights` are shown separately from `Negative rights` (if negative rights exist).

Example: Listing the ACL for a Home Directory

If the user *terry* (local cell is *comp.com*) wants to see the ACL for his home directory, he would type:

```
% fs listacl $HOME
Access list for /afs/comp.com/usr/terry is
Normal rights:
   system:anyuser rl
   terry rlidwka
```

This indicates that *terry* has all seven rights on his home directory, and all users who belong to the *system:anyuser* group have READ and LOOKUP rights. This allows all users to list *terry's* home directory, read files in *terry*'s home directory, and list *terry*'s subdirectories.

Note: *terry* could also have typed **fs listacl ~terry**; or, if he was already in his home directory, he could type **fs la**.

Example: Listing the ACL for the Current Directory

Suppose *terry* is in his subdirectory called *strategy* and wants to check its ACL. He dos not need to specify a directory because the current directory is the default for the **fs listacl** command. At the command shell prompt, he would type

```
% fs listacl
Access list for . is
Normal rights:
   system:authuser rl
   pat rlw
   terry rlidwka
Negative rights:
   terry:other-dept rlidwka
```

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` indicates that anyone in the group *terry:other-dept* is not allowed to access this directory in any way; *terry* has denied them all seven rights. Note that any authenticated user not in the *terry:other-dept* group has access to the subdirectory as *system:authuser*.

See section 5.5.2 to learn more about denying access to directories by using normal and negative rights. See chapter 6 to learn more about creating groups.

Example: Listing ACLs for Multiple Directories

This example shows the output when more than one directory is specified and how it is legal to combine different types of pathnames. Interpretation of the lists is the same as for the other examples.

Suppose *terry* is in a subdirectory of his home directory and wants to see the ACLs of three directories: the current directory, the home directory of his manager (*pat*), and another subdirectory of his home directory called **plans**. At the command shell prompt, he would type

```
% fs listacl . /afs/comp.com/usr/pat ../plans
Access list for . is
Normal rights:
   system:anyuser rl
   pat:dept rliw

Access list for /afs/comp.com/usr/pat is
Normal rights:
   system:anyuser rl
   pat rlidwka
   terry rliw

Access list for ../plans is
Normal rights:
   terry rlidwka
   pat rlidw
```

5.5. Changing an ACL

You are allowed to change any ACL for which you have the ADMINISTER right. You always have ADMINISTER rights on your home directory and all its subdirectories. Even if your ADMINISTER right is removed from the ACLs for your directories, you can still administer the ACLs in your home directory and its subdirectories.

To determine if you have the ADMINISTER right for any other directory, issue the **fs listacl** command with the appropriate directory. Look for the letter `a` in the list of rights following your user name or the name of a group to which you belong.

To change a directory's ACL, use the **fs setacl** command.

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

where

directory specifies each directory whose ACL you want to change. Abbreviated pathnames are interpreted relative to the current directory.

access list entries
defines a list of one or more ACL entries each of which specifies a user or group name and a corresponding set of access rights.

-clear removes all existing entries on the specified ACL prior to adding the new ones. Use caution with this flag because if one of the ACL entries does not grant all rights to the owner of the directory it can become awkward for the owner to access items in the directory.

-negative puts the indicated entries on the `Negative rights` list rather than the `Normal rights` list. Users and groups on the `Negative rights` list are specifically denied the listed rights.

If an ACL already grants certain rights to a user or group, the rights you specify with the **fs setacl** command replace the existing rights; they are not added to them. In other words, if you want a user or group to retain the rights they already have, include those rights when you enter the new set of rights.

See the following sections for information on making specific types of changes to ACLs:

- See section 5.5.1 to learn how to grant access.
- See section 5.5.2 to learn how to deny access.
- See section 5.5.3 to learn how to restore access after denying it.
- See section 5.5.4 to learn how to erase an ACL and replace it with a new one.

5.5.1. Granting Access to Directories

To grant access to a directory, use the **fs setacl** command. You must provide the name of the directory, the users or groups you want to add to its ACL, and their corresponding rights. An ACL may contain up to 20 entries. Section 5.5 explains more completely what values are acceptable for specifying users, groups, and access rights.

To Grant Access to a Directory – fs setacl

At the command shell prompt, type

% **fs setacl -dir** <*directory*>$^{+}$ **-acl** <*access list entries*>$^{+}$

where

directory specifies each directory whose ACL you want to change. Abbreviated pathnames are interpreted relative to the current directory.

access list entries
defines a list of one or more ACL entries each of which specifies a user or group name and a corresponding set of access rights.

Example: Adding a User to an ACL

Suppose *terry* wants to give *pat* READ and LOOKUP access to his *notes* subdirectory; from the *notes* subdirectory he would type

% **fs setacl . pat rl**

or

% **fs setacl . pat read**

To verify the ACL, *terry* uses the **fs listacl** command.

```
% fs listacl .
Access list for . is
Normal rights:
   pat rl
   terry rlidwka
```

Example: Adding Multiple Groups and Users to an ACL

Suppose *terry* wants to give the following access rights to his *notes* subdirectory:

- READ, LOOKUP, and INSERT for members of *system:anyuser*.
- All rights for his manager *pat*.
- All rights except ADMINISTER for his friend *indira*

In his *notes* subdirectory, he would type

% fs setacl . system:anyuser rli pat all indira write

or

% fs setacl . system:anyuser rli pat rlidwka indira rlidwk

To verify the ACL, *terry* uses the **fs listacl** command in his *notes* subdirectory:

```
% fs listacl .
Access list for . is
Normal rights:
   system:anyuser rli
   pat rlidwka
   terry rlidwka
```

5.5.2. Denying Access to Directories

There are two ways to deny someone access to a directory: remove the user or group from the `Normal rights` list or add them to the `Negative rights` list. Users that do not appear on the `Normal rights` list do not have access to that directory unless they are granted access by belonging to a group on the access control list. In particular, everyone on the system has the access rights of the group *system:anyuser*.

To specifically deny a user or group access, put an entry on the `Negative rights` list of the ACL. This denies access because the file server computes negative rights after normal rights so an entry on the `Negative rights` list cancels out any rights someone may gain under `Normal rights`.

Exception: `Negative rights` do not cancel out *system:anyuser* normal rights. If *system:anyuser* has access rights on the `Normal rights` list, any user on the `Negative rights` list can unlog and use *system:anyuser* access rights.

Putting users on the `Negative rights` list is most useful when you want most members of a group to have a given set of access rights but want to restrict access to a few group members.

To Remove an Entry from the Normal rights List – fs setacl

To remove a user or group from the `Normal rights` list, type

% fs setacl -dir <*directory*>$^+$ **-acl <***user or group name*> **none**

where

directory specifies each directory whose ACL you want to change. Abbreviated pathnames are interpreted relative to the current directory.

user or group name
specifies the user or group part of the ACL entry; the word **none** indicates that *user or group name* will be given no rights.

You can add someone new to an access list at the same time you remove someone; see the example below.

Example: Removing a User From the Normal rights List

Suppose *terry* wants to remove *pat*'s entry from the `Normal rights` part of his current directory's ACL, he would type

```
% fs setacl . pat none
% fs listacl .
Access list for . is
Normal rights:
  terry:dept rli
  system:anyuser ri
  terry rlidwka
```

Note: Keep in mind that if *system:anyuser* or another group to which *pat* belongs appears on the list, *pat* will still have those access rights. You might also want to deny *pat* access by putting her on the `Negative rights` list and remove *system:anyuser* from the `Normal rights` list.

Example: Removing a User While Adding Another

It is legal to remove a user from the `Normal rights` list while simultaneously adding a user to it. Suppose *terry* wants to remove *pat* from the `Normal rights` list and grant *indira* rights in the same command, he would type

```
% fs setacl . pat none indira rlidw
% fs listacl .
Access list for . is
Normal rights:
   terry:dept rli
   system:anyuser ri
   indira rlidw
   terry rlidwka
```

To Add an Entry to the Negative rights List – fs setacl -negative

To add an entry to the `Negative rights` list, you must use the **-negative** flag with the **fs setacl** command. The **-negative** flag indicates that all of the *access list entries* should go on the `Negative rights` list of the ACL. When you assign users to the `Negative rights` list, do not give *system:anyuser* more rights than the least privileged user because users can unlog and obtain *system:anyuser* access rights.

At the command shell prompt, type

`%` **fs setacl -dir** <*directory*>$^{+}$ **-acl** <*access list entries*>$^{+}$ **-negative**

where

directory specifies each directory whose ACL you want to change. Abbreviated pathnames are interpreted relative to the current directory.

access list entries
defines a list of one or more ACL entries each of which specifies a user or group name and a corresponding set of access rights.

-negative puts the indicated entries on the `Negative` rights list rather than the `Normal rights` list.

Example: Adding a User to the Negative rights List

Suppose *terry* had given READ, LOOKUP, INSERT, DELETE, and WRITE access to his *plans* subdirectory to all the members of a group *terry:dept*. The ACL for his directory might look like this:

```
% fs listacl /afs/comp.com/usr/terry/plans
Access control list for /afs/comp.com/usr/terry/plans is
Normal rights:
   system:anyuser rl
   terry:dept rlidw
   terry: rlidwka
```

However, *terry* notices that one of his co-workers, *indira*, has been making changes to files without permission. While *terry* wants to deny *indira* WRITE and DELETE rights, he does not want to remove *indira* from the *terry:dept* group entirely. In his *plans* subdirectory, *terry* would type

```
% fs setacl . indira wd -neg
% fs listacl
Access control list for . is
Normal rights:
   system:anyuser rl
   terry:dept rlidw
   terry: rlidwka
Negative rights:
   indira wd
```

5.5.3. Restoring Access to Directories after Denying It

It is possible to restore rights you have taken away. If you simply used the **none** access right group to remove a user or a group from the `Normal rights` list, use the regular **fs setacl** command to restore the right (see section 5.5.1).

If you added a user or a group to the `Negative rights` list, you must remove the user or group from that list and then grant them rights on the `Normal rights` list. If you forget to remove them from the `Negative rights` list, the user or group will still be locked out because negative rights cancel out normal rights.

To Remove an Entry from the Negative rights List – fs setacl -negative

To remove a user or group from the `Negative rights` list, assign the user or group the **none** shorthand grouping of rights on the `Negative rights` list. Use the **-negative** flag and provide the directory, the user or group name, and the **none** rights grouping.

Issuing this command returns the specified negative ACL entry to the default condition of not appearing on the ACL at all. You can then grant the user or group normal rights or a different set of negative rights with the **fs setacl** command (see sections 5.5.1 and 5.5.2).

At the command shell prompt, type

% **fs setacl -dir** <*directory*>+ **-acl** <*user or group name*> **none -negative**

where

directory specifies each directory whose ACL you want to change. Abbreviated pathnames are interpreted relative to the current directory.

user or group name
specifies the user or group part of the ACL entry; the word **none** indicates that *user or group name* will be removed from the list.

-negative specifies that the entry should be removed from the `Negative rights` list rather than the `Normal rights` list.

Example: Restoring Access (Negative rights)

In the example on page 5-14, *terry* put *indira* on the `Negative rights` list for the *plans* subdirectory. Now, *indira* has promised not to change *terry*'s files without telling him, so *terry* wants to restore *indira*'s normal rights by removing her from the `Negative rights` list. He types

```
% fs setacl /afs/comp.com/terry/plans indira none -negative
% fs listacl /afs/comp.com/terry/plans
Access control list for /afs/comp.com/usr/terry/plans is
Normal rights:
  system:anyuser rl
  terry:dept rlidw
  terry rlidwka
```

Note that *indira* no longer appears on the `Negative rights` list. So *terry* can now grant her normal rights if he chooses; however, this is unnecessary because *indira* has `rlidw` rights by belonging to *terry:dept*.

5.5.4. Replacing an ACL

You may sometimes want to clear an ACL completely and define new rights for the directory. Use the **-clear** flag on the **fs setacl** command to erase both the `Normal rights` and `Negative rights` lists. The list of new rights you define may be either normal or negative and each list must be added separately.

Be sure to grant yourself all seven rights when you issue this command. You can get into difficulty if you take yourself off an ACL. Without the LOOKUP right, you will be unable to use short-hand directory notation, such as interpreting "**.**" or "**..**", though you can still perform these commands if you provide a complete pathname. For example, you will not be able to use "**fs listacl .**" or "**cd ..**", but "**fs listacl $HOME**" or "**cd /afs/comp.com/usr/terry**" will still work.

Note: If you lock yourself out of a directory you own, you can always restore your rights to it because you always retain implicit ADMINISTER rights on your home directory and its subdirectories. To restore your rights, use the **fs setacl** command.

To Replace an ACL – fs setacl -clear

Use the **-clear** flag to replace a directory's ACL completely. The **-clear** flag erases the current ACL for the directory, replacing it with the new ACL entry. You can set the `Negative rights` list by including the **-negative** flag or the `Normal rights` list by omitting it.

At the command shell prompt, type

% **fs setacl -dir** <*directory*>+ **-acl** <*access list entries*>+ **[-clear] [-negative]**

where

directory specifies each directory whose ACL you want to change. Abbreviated pathnames are interpreted relative to the current directory.

access list entries
defines a list of one or more ACL entries each of which specifies a user or group name and a corresponding set of access rights.

-clear removes all existing entries on the specified ACL prior to adding the new ones. Use caution with this flag because if an ACL does not grant all rights to the owner of the directory, it can become awkward for the owner to access items in the directory.

-negative puts the indicated entries on the `Negative rights` list rather than the `Normal rights` list.

Example: Replacing an ACL

Suppose that the ACL on the current directory has become so crowded with groups and individual users with different combinations of rights that *terry* is no longer sure what access he is giving or denying. He can return the ACL to a simpler state, where only he and *pat* have rights, by typing

```
% fs setacl . terry all pat write -clear
% fs listacl .
Access control list for . is
Normal rights:
  terry rlidwka
  pat rlidwk
```

5.6. Copying ACLs Between Directories

The **fs copyacl** command allows you to copy an ACL from one directory to one or more other directories. You can use the command to copy the ACL from any directory for which you have the LOOKUP right to the ACL of any directory for which you have the ADMINISTER right. You can specify a filename as the directory from which the ACL is to be copied, in which case the ACL of the file's parent directory is used; but you must have both READ and LOOKUP rights on that directory. Use the **fs listacl** command to determine the rights you have for a directory.

The **fs copyacl** command does not affect entries on the ACL of the *source directory*; however, it 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*, it is copied to the ACL of the *destination directory*.
- If an entry is on the ACL of the *destination directory* but not on the ACL of the *source directory*, one of the following occurs:
  - If the **-clear** flag is not included with the command, the entry on the *destination directory* is not affected.
  - If the **-clear** flag is included with the command, the entry on the *destination directory* is removed from the ACL.

To Copy an ACL Between Directories – fs copyacl

To copy an ACL from one directory to one or more other directories, type

% fs copyacl -fromdir <*source directory*> **-todir** <*destination directory*>$^+$ **[-clear]**

where

source directory
: specifies the directory whose ACL is to be copied to each *destination directory*. It is legal to specify a directory name or a filename. If you specify a filename, the parent directory of the file is used.

destination directory
: specifies each *destination directory* to receive the ACL from the *source directory*.

-clear
: 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.

Example: Copying an ACL from One Directory to Another

Suppose *terry* is in his home directory and wants to copy its ACL to the ACL of his *plans* subdirectory. The current ACLs of the two directories can be listed with the **fs listacl** command by typing

% fs listacl . plans
```
Access list for . is
Normal rights:
   terry rlidwka
   smith rl
   jones rl

Access list for plans is
Normal rights:
   terry rlidwk
   pat rlidk
```

The **fs copyacl** command can be used to copy the ACL from the current directory to the *plans* subdirectory by typing

% fs copyacl -from . -to plans

The ACLs of the two directories can again be listed with **fs listacl**:

% fs listacl . plans
```
Access list for . is
Normal rights:
   terry rlidwka
   smith rl
   jones rl

Access list for plans is
Normal rights:
   terry rlidwka
   pat rlidwk
   smith rl
   jones rl
```

5.7. Using the UNIX Mode Bits in AFS

In a standard UNIX file system, the UNIX mode bits associated with each file and directory determine who can access that file or directory and in what manner. The UNIX mode bits appear in the output line for the standard **ls -l** command. The first character of the output from the **ls -l** command tells the type of the listed element (`-` for a file, `d` for a directory, `l` for a link). The remaining characters form three `rwx` groupings, that specify the read (`r`), write (`w`) and execute (`x`) privileges for each grouping (that is, the *owner*, *group*, and *other*).

AFS calculates file access based on two factors: the ACL entries of the parent directory and the UNIX *owner* mode bits.

- **The ACL entries:** A user may only access a file in a given way if he or she has been granted the appropriate rights on the ACL (whether directly, or through membership in a group).
- **The UNIX *owner* mode bits:** AFS ignores the UNIX *group* and *other* mode bits and looks only at the *owner* mode bits. If the *owner* mode bits on a file or directory forbid a specific type of access (for example, writing to the file), no one may access the file in that way no matter what type of ACL access rights they have.

Therefore, the following rules apply when working on AFS files:

- A user with appropriate AFS rights can only read a file if its UNIX `r` *owner* mode bit is turned on.
- A user with appropriate AFS rights can only write to a file if its UNIX `r` and `w` *owner* mode bits are turned on.
- A user with appropriate AFS rights can only execute a file if its UNIX `r` and `x` *owner* mode bits are turned on.

Note: AFS does not support write-only files; it requires the `r` mode bit to copy a file to the Cache Manager on a client workstation.

You can use the standard UNIX **chmod** command to toggle the *owner*, *group*, and *other* mode bits. (Of course, the *owner* mode bits are the only ones of consequence inside AFS.) You must have the WRITE and LOOKUP rights on the ACL of a directory housing a file to change the file's mode bits.

Note: UNIX also associates three additional mode bits, the "SUID," "SGID," and "sticky" bits, with a file. This documentation does not address these additional mode bits.

Example: Turning Off Write Permission for a File

Suppose *terry* is chairing a committee that is writing a proposal. As each section is approved, he turns off write access to that file. Committee members can continue to write and edit the unapproved sections and can still read and refer to the approved sections, but *terry* wants no one to alter an approved section. To turn off write access to the approved *proposal.chap2* section, *terry* types the following UNIX command in the *proposal* subdirectory:

```
% chmod -w proposal.chap2
% ls -l
-rw-r--r--  1 terry     5732 Dec  1 19:57 conclusion
-rw-r--r--  1 terry     5732 Dec  1 19:57 intro
-r--r--r--  1 terry     5732 Dec  1 19:57 proposal.chap2
-rw-r--r--  1 terry     5732 Dec  1 19:57 proposal.chap3
-rw-r--r--  1 terry     5732 Dec  1 19:57 proposal.chap4
```

Notice that the file *proposal.chap2* does not have the `w` (write) bit. No user that accesses the *proposal* subdirectory, including the owner, *terry*, can write to the file, even if the user has the WRITE and LOOKUP rights on the ACL of the directory.

6. Using Groups

This chapter explains how to create your own groups and suggests different ways to use them. It contains the following sections:

6.1. About Groups

An AFS ***group*** is a defined list of individual users that you can place on the access control lists (ACLs) so that you can grant the same access rights to a number of people at once. Groups make the job of ACL maintenance much easier. Instead of adding users to (and removing them from) ACLs separately, users can be added to and removed from groups, thus updating all ACLs on which the groups are included.

When you create a group, you automatically become its ***owner***. A group's owner is the only one allowed to ***administer*** the group. Administering a group includes adding members to it, removing members from it, renaming it, changing its owner, or deleting it entirely.

See chapter 5 to learn about ACLs, how AFS file protection works in general, and how to add individuals and groups to access control lists.

Note: Specific client machines can also be members of a group. If a machine is a member of a group, anyone logged in to that machine has the access rights of the group. Most cells that use "machine groups" use them for very specific purposes (like adhering to software license agreements). Talk with your system administrator before putting a client machine in a group or using a machine group on an ACL.

6.1.1. Suggestions for Using Groups Effectively

There are three typical ways to use groups, each suited to a particular purpose: private use, shared use, and group use. The following are only suggestions – you are free to use groups in any way you choose.

6.1.1.1. Private Use

When you create a group for your own convenience and put it on your own access control lists, without informing the group's members, that group is for your ***private use***. The only thing the group's members notice is that they are granted (or denied) certain kinds of access to a directory.

The existence of a private use group and the identity of its members is not necessarily secret. Anyone can learn that a group exists by seeing it on an ACL when they issue the **fs listacl** command (see chapter 5). And anyone can add any group to a directory's ACL, assuming they have ADMINISTER rights on that directory.

However, as the group's owner, you retain sole administrative control over the group. Furthermore, you can restrict who can list the members of a group (see section 6.6).

6.1.1.2. Shared Use

When you create a group for ***shared use***, you inform the group's members that you have formed a group with them in it. Like private use groups, anyone can add a shared use group to their own access control lists; however, under shared use, you specifically sanction this use. Again, though, as the group's owner, you retain sole control over its membership.

When you add a group owned by someone else to 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 to your files in a way you did not intend. See section 6.6 to learn how to protect group information.

For example, assume the manager of a department creates a group consisting of all the employees in the department. The employees can add the group to the directories that store department-related files so that everyone in the department can look at the files. However, the manager retains control over the group's membership.

6.1.1.3. Group Use

When you create a group for group use, there are two possible variations:

- ***Group-owned*** – One group owns another. All members of the owner group can administer the owned group; the members of the owned group have no administer rights.
- ***Self-owned*** – A group owns itself, so the membership of the owned and owner groups is identical.

Once a group is created, the owner can change the owner to a group, even the group itself (see section 6.7). 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; if the original creator leaves the group, he or she does not need to remember to transfer ownership to someone else.

Keep in mind that everyone in an owner group can make changes that affect others negatively: removing members from the group, adding members that do not belong, or changing ownership to themselves exclusively. These problems can be especially sensitive in a self-owned group. Using an owner group works best if all members know and trust each other, so it is probably best to keep the number of people in an owner group small.

6.1.2. Group Names

Most group names have two parts, separated by a colon, as follows:

owner-name:*group-name*

The *owner-name* is the name of the owner of the group; the *group-name* is the actual name of the group. You may also encounter groups that do not have an owner prefix; these are special groups created by system administrators. All of the groups you create must have an *owner-name* and a *group-name*, separated by a colon.

Together, *owner-name* plus *group-name* can amount to a maximum of 63 characters. The *group-name* can contain lowercase letters, numbers, or any punctuation except the colon; uppercase letters or spaces are not allowed.

6.1.3. Group Quota

Your cell's system administrators set a ***group quota*** for the number of groups you are allowed to create. When you create a group, your group quota decreases by one. For information on determining your quota, see section 6.3.

When a group that you created is deleted, your quota increases by one, even if you are no longer the owner. Transferring ownership of a group does not increase your group quota because you are still the creator. If you exhaust your group quota and need to create more groups, contact your system administrator to have your group quota increased.

6.2. Listing Information About Groups

You can use the following commands to determine group-related information about groups:

- Use the **pts membership** command to determine who belongs to a group.
- Use the **pts examine** command to determine who owns or who created a group.
- Use the **pts listowned** command to determine the groups that a group owns.

Note: The standard system groups *system:anyuser* and *system:authuser* are not listed in the output from these commands.

You can also check on group-related information about particular users, such as the groups they own (see section 6.3).

To List the Members of a Group – pts membership

You must specify the complete name or AFS User ID (UID) of each group whose members you want to list. At the command shell prompt, type

% pts membership -nameorid <*user or group name or id*>$^{+}$

where *user or group name or id* specifies the complete name or AFS UID of each group about which group membership information is to be displayed.

Example: Listing Members of a Group

Suppose *terry* wants to make sure that *pat* belongs to the group *terry:dept*. He would check this by typing

% pts membership terry:dept

```
Members of terry:dept (id: -286) are:
  terry
  smith
  pat
  johnson
```

The output indicates that *pat* does belong to the group.

To List Who Owns/Created a Group – pts examine

You must specify the complete name or AFS UID of each group about which owner and creator information is to be displayed. At the command shell prompt, type

% **pts examine -nameorid** <*user or group name or id*>$^+$

where *user or group name or id* specifies the complete name or AFS UID of each group about which owner and creator information is to be displayed.

The output from the **pts examine** command displays information in the following fields:

- **Name** – Confirms that the group exists in the system.
- **id** – Specifies the group's "AFS UID," a number that AFS servers use internally. This number is negative for groups, positive for users.
- **owner** – Indicates which user or group owns the group.
- **creator** – Indicates which user created the group.
- **membership** – For user entries, indicates the number of groups to which the user belongs; for group entries, indicates the number of users who belong to the group.
- **flags** – Indicates who can perform certain actions on this group entry (see section 6.6).
- **group quota** – For user entries, indicates the user's group quota; it is meaningless for group entries and should be `0`.

Example: Listing a Group's Owner and Creator

Suppose the user *terry* knows that *pat* created a group called *pat:staff*, which includes members of the department *pat* manages. *terry* might want to use this group on access control lists after finding out who can administer it (who owns it). To do this, *terry* could type

```
% pts examine pat:staff
Name: pat:staff, id: -673, owner: pat:staff, creator: pat,
  membership: 15, flags: S-M--, group quota: 0,
```

It turns out that *pat:staff* is self-owned. Because everyone in the group can change its membership, *terry* needs to decide if he wants to use a self-owned group on his ACLs.

To List the Groups a Group Owns – pts listowned

You must specify the complete name or AFS UID of each group about which ownership information is to be displayed. At the command shell prompt, type

% pts listowned -nameorid <*user or group name or id*>$^{+}$

where *user or group name or id* specifies the complete name or AFS UID of each group about which group ownership information is to be displayed.

Example: Listing the Groups a Group Owns

To find out if *terry:dept* owns any other groups, *terry* would type

```
% pts listowned terry:dept
Groups owned by terry:dept (id: -567) are:
  terry:other-dept
  terry:friends
```

The group owns two other groups: *terry:other-dept* and *terry:friends*.

6.3. Listing Group-Related Information About Users

You can use the following commands to determine group-related information about users:

- Use the **pts membership** command to determine the groups to which a user belongs.
- Use the **pts listowned** command to determine the groups a user owns.
- Use the **pts examine** command to determine how many additional groups a user can create.

To List Groups to Which a User Belongs – pts membership

You must specify the user name or AFS UID of each user about which group membership information is to be displayed. At the command shell prompt, type

% **pts membership -nameorid** <*user or group name or id*>+

where *user or group name or id* specifies the user name or AFS UID of each user about which group membership information is to be displayed.

Example: Listing the Groups to Which a User Belongs

Suppose *terry* wants to know the groups to which both he and *pat* belong. He would type

```
% pts membership terry pat
Groups terry (id: 1022) is a member of:
  pat:staff
  indira:friends
  pat:accounting
Groups pat (id: 1845) is a member of:
  pat:staff
  sam:managers
```

To List the Groups a User Owns – pts listowned

You must specify the user name or AFS UID of each user about which group ownership information is to be displayed. At the command shell prompt, type

% **pts listowned -nameorid** <*user or group name or id*>+

where *user or group name or id* specifies the user name or AFS UID of each user about which group ownership information is to be displayed.

Example: Listing the Groups a User Owns

Suppose *terry* wants to know the groups that both he and *pat* own. He can display this information by typing

```
% pts listowned terry pat
Groups owned by terry are:
   terry:dept
   terry:other-dept
   terry:friends
   terry:confidential
Groups owned by pat are:
   pat:staff
   pat:plans
   pat:managers
```

To List a User's Group Quota – pts examine

You must specify the user name or AFS UID of each user about which group quota information is to be displayed. At the command shell prompt, type

% **pts examine -nameorid** <*user or group name or id*>+

where *user or group name or id* specifies the user name or AFS UID of each user about which group quota information is to be displayed.

The **pts examine** command displays the same types of information for users that it displays for groups. It provides information in the following fields:

- **Name** – Confirms that the user exists in the system.
- **id** – Specifies the user's "AFS UID," a number that AFS servers use internally. This number is negative for groups, positive for users.
- **owner** – Indicates which user or group owns the user entry. This is usually *system:administrators* for users.
- **creator** – Indicates which user created the user entry.
- **membership** – For user entries, indicates the number of groups to which the user belongs; for group entries, indicates the number of users who belong to the group.
- **flags** – Indicates who can perform certain actions on this user entry (see section 6.6).
- **group quota** – For user entries, indicates the user's group quota; it is meaningless for group entries and should be `0`.

Example: Listing a User's Group Quota

Suppose *pat* has created several groups and wants to know how many more she can create. She would type

```
% pts examine pat
Name: pat, id: 1045, owner: system:administrators, creator: admin,
membership: 15, flags: S-M--, group quota: 17,
```

The relevant field is `group quota`, which shows that *pat* can create another 17 groups (her current group quota is `17`).

6.4. Creating Groups and Adding Members

You can use the following commands to create groups and add members to them:

- Use the **pts creategroup** command to create a group.
- Use the **pts adduser** command to add a user to a group.

Before you can add members to a group, you must create the group. When you create a group, you become its owner automatically. As explained in section 6.1, this means you are allowed to add and delete members, change the group's name, transfer ownership of the group, or delete the group entirely.

A newly created group has no members. You may add users to a group you own at any time. You cannot make a group a member of another group.

To Create a Group – pts creategroup

You must specify the name of each group to be created. At the command shell prompt, type

% **pts creategroup -name** <*group name*>$^{+}$ **[-owner** <*owner of the group*>**]**

where

group name
: specifies the name of each group to be created. A name must be of the form *owner-name*:*group-name*; see section 6.1.2 for information on group naming restrictions.

owner of the group
: specifies the user or group to own each group specified with **-name** (if the group(s) is to be owned by a group or a user other than the command issuer). Specify the user name of an individual or the complete name of a group. By default, the issuer of the command will be the group owner.

Issuing this command causes your group quota to decrease by one for each group you create.

Example: Creating a Group

Suppose user *terry* wants to create a group that contains all members of his department, he would type

```
% pts creategroup terry:dept
```

The **pts examine** command can be used to display information about the group.

```
% pts examine terry:dept
Name: terry:dept, id: -104, owner: terry, creator: terry,
  membership: 0, flags: S----, group quota: 5.
```

To Add Members to a Group – pts adduser

You can add members to any group you own. Specify the names of all users to be added to a group and the complete name of the group to which they are to be added. At the command shell prompt, type

% **pts adduser -user** <*user name*>+ **-group** <*group name*>+

where

user name
: specifies the user name of each individual to be added to the groups specified with **-group**. Groups cannot be members of other groups.

group name
: specifies the complete name of each group to which you want to add the users specified with **-user**.

Example: Adding Members to a Group

Suppose *terry* now wants to add *pat*, *indira*, and *smith* to the group *terry:dept*. He would type

% **pts adduser -user pat indira smith -group terry:dept**

The **pts membership** command can be issued to list the members of the group.

```
% pts members terry:dept
Members of terry:dept (id: -286) are:
  pat
  indira
  smith
```

Note that *terry* must add himself to the group to be included in its membership.

6.5. Removing Users from a Group and Deleting a Group

You can use the following commands to remove members from a group, delete a group, or remove deleted groups from ACLs:

- Use the **pts removeuser** command to remove a user from a group.
- Use the **pts delete** command to delete a group entirely.
- Use the **fs cleanacl** command to remove deleted groups from ACLs.

You must own a group to remove a user from it or to delete it. Use the **pts examine** command to determine if you own a group (see section 6.2). Use the **pts listowned** command to display a list of all the groups you own (see section 6.3).

When a group that you created is deleted, your group quota increments by one. This is true even if you no longer own the group. Use the **pts examine** command to check your group quota.

When a user or group is deleted, its AFS UID appears on ACLs in place of its AFS name. Use the **fs cleanacl** command to remove deleted user or group entries from ACLs.

Note: Although the **pts delete** command is also used to delete user entries as well as group entries, to delete a user entry, you must be a member of *system:administrators*.

To Remove a User from a Group – pts removeuser

You must specify the names of the users to be removed from groups and the complete names of the groups from which they are to be removed. At the command shell prompt, type

% **pts removeuser -user** <*user name*>+ **-group** <*group name*>+

where

user name
: specifies the user name of each user to be removed from each group specified with **-group**.

group name
: specifies the complete name of each group from which each user specified with **-user** is to be removed.

Example: Removing a User from a Group

Suppose *terry* wants to remove *pat* from the group *terry:dept*, he would type

% **pts removeuser pat terry:dept**

To remove *pat* from both *terry:dept* and *terry:friends* with the same command, *terry* would type

% **pts removeuser pat -group terry:dept terry:friends**

Note: The **-group** switch must be used with the second **pts removeuser** command because multiple group names are specified.

To Delete a Group – pts delete

You must specify the complete name or AFS UID of each group to be removed. At the command shell prompt, type

% **pts delete -nameorid** <*user or group name or id*>+

where *user or group name or id* specifies the complete name or AFS UID of each group to be deleted.

Example: Deleting a Group

If *terry* decided to delete the group *terry:dept* from the system, he would type

% **pts delete terry:dept**

Assuming the group *terry:dept* was included on the ACL for the directory *plans*, deleting the group would cause its entry on the ACL to be replaced by its AFS UID, as follows:

% **fs listacl plans**

```
Access list for plans is
Normal rights:
  -409 rlidwka
  terry:colleagues rl
  pat rliw
```

To Remove a Deleted Group from an ACL – fs cleanacl

You must specify the name of each directory (or the name of a file in each directory) from which the AFS UIDs of deleted groups (and users) are to be removed. At the command shell prompt, type

% **fs cleanacl [-path** <*dir/file path*>**+]**

where *dir/file path* specifies each directory from whose ACL deleted groups (and users) are to be removed. If a filename is specified, deleted groups are removed from the ACL of the file's parent directory.

Example: Removing a Deleted Group from an ACL

After *terry* removes the group *terry:dept*, the group's AFS UID still appears on any ACLs on which the group was included.

```
% fs listacl plans
Access list for plans is
Normal rights:
  -409 rlidwka
  terry:colleagues rl
  pat rliw
```

To remove the group's AFS UID from the ACL of the *plans* subdirectory, *terry* would type

```
% fs cleanacl plans
```

The group's AFS UID no longer appears on the ACL.

```
% fs listacl plans
Access list for plans is
Normal rights:
  terry:colleagues rl
  pat rliw
```

6.6. Protecting Group-Related Information

You can use ***privacy flags*** to limit the types of access various people have to information about your groups. Five privacy flags, each with a corresponding letter, are used to protect group information. You can use the following commands to set and view the privacy flags for a group:

- Use the **pts setfields** command to set the privacy flags for a group.
- Use the **pts examine** command to display the privacy flags for a group.

6.6.1. Using Privacy Flags

The following list explains the privacy flags in the order in which the **pts examine** command displays them:

- STATUS (**s**) – Controls use of the **pts examine** command to obtain status information about a group.
- OWNED (**o**) – Controls use of the **pts listowned** command to determine the groups that a user or group owns.
- MEMBERS (**m**) – Controls use of the **pts membership** command to determine the groups to which a user belongs or the users who belong to a group.
- ADD (**a**) – Controls use of the **pts adduser** command to add a user to a group.
- REMOVE (**r**) – Controls use of the **pts remove** command to remove a user from a group.

The default privacy flags for groups are `S-M--`. These flags allow everyone to obtain status information about a group and list its members. The owner of a group always has all of the rights granted by the five privacy flags. The owner can assign privacy flags to three categories of users:

- The owner of the group – provide rights to only the owner of the group by specifying a hyphen (-).
- The members of the group – provide rights to only the owner and the members of the group by specifying a lowercase letter.
- Everyone (equivalent to *system:anyuser* on ACLs) – provide rights to everyone by specifying a capital letter.

To List a Group’s Privacy Flags – pts examine

You must specify the complete name or AFS UID of each group whose privacy flags you want to display. At the command shell prompt, type

% **pts examine -nameorid** <*user or group name or id*>$^+$

where *user or group name or id* specifies the complete name or AFS UID of each group whose privacy flags are to be displayed.

Example: Listing a Group’s Privacy Flags

If *terry* wants to list the privacy flags for his group *terry:dept*, he would type

```
% pts examine terry:dept
Name: terry:dept, id: -567, owner: terry, creator: terry,
  membership: 6, flags: S-m--, group quota: 0.
```

The `flags: S-m--` field indicates the following:

- Everyone can obtain status information.
- Group members can list member information.
- Only the owner (*terry*) can list owners, add members, and remove members.

To Set the Privacy Flags on a Group – pts setfields

You must specify the complete name or AFS UID of each group whose privacy flags you want to set. At the command shell prompt, type

% **pts setfields -nameorid** <*user or group name or id*>$^+$ **-access** <*set privacy flags*>

where

user or group name or id
: specifies the complete name or AFS UID of each group whose privacy flags are to be set.

set privacy flags
: specifies the privacy flags to be set for each group specified with **-nameorid**. Specify the privacy flags according to the following guidelines:
  - Specify the flags in the order indicated previously (`somar`).
  - The hyphen is equivalent to lowercase letters for self-owned group entries, because the members of a self-owned group are owners of the group.
  - Include a letter or hyphen for all five privacy flags except the first slot (`s`) which must be a letter.

- Do not type the lowercase `o` in the second slot. Members of a group are automatically the owners of any groups owned by the group, so the lowercase `o` is equivalent to the hyphen in this slot.
- Do not type the uppercase `A` in the fourth slot or an uppercase `R` in the fifth slot; these flags would allow anyone to add or remove users from the group.

Note: Do not use this command to change the group quota for a group; a group cannot create another group.

Example: Setting the Privacy Flags on a Group

Suppose *terry* wants to allow the following access to information about his group *terry:dept*:

- `S` to allow everyone to list status information about *terry:dept*.
- `O` to allow everyone to list the groups owned by *terry:dept*.
- `m` to allow the members of *terry:dept* to list their fellow members.
- `-` to allow only *terry* to add members.
- `-` to allow only *terry* to remove members.

He would type

```
% pts setfields terry:dept -access SOm--
```

The **pts examine** command can then be used to display the privacy flags for the group.

```
% pts examine terry:dept
Name: terry:dept,  id: -567, owner: terry, creator: terry,
membership: 6, flags: SOm--, group quota: 0.
```

6.7. Changing a Group’s Owner or Name

You can use the following commands to change the owner or name of a group:

- Use the **pts chown** command to change the owner of a command.
- Use the **pts rename** command to change the name of a group.

You must own a group (or be a member of *system:administrators*) to change the owner or name of a group. You can change the owner of a group to another user, another group, or the group itself. If you are going to transfer ownership of a group and you wish to be a member of the group, make sure to add yourself to the group before you relinquish ownership.

The **pts chown** command automatically changes the *owner-name* prefix of the group name to the new owner. If the new owner is a group, only the *owner-name* from the name of the new owner group is included in the *owner-name* field. The command does not, however, automatically change the *owner-name* prefixes of any groups that the group owns. (See the examples below for clarification.)

The **pts rename** command is typically used to change only the *group-name* (the part that follows the colon). A group’s *owner-name* can be changed only if the owner of the group has been changed.

To Change a Group’s Owner – pts chown

You must type both the complete name of the group whose owner is to change and the name of the user or group that is to own it. At the command shell prompt, type

% pts chown -name <*group name*> **-owner** <*new owner*>

where

group name
: specifies the complete name of the group whose owner is to be changed to the user or group specified with **-owner**.

new owner
: specifies the user name of the user or the complete name of the group that is to own the group specified with **-name**.

Example: Changing a Group’s Owner to Another User

Suppose *pat* is leaving the system and *terry* wants to assume administration of the group *pat:staff*, of which *pat* is the current owner. Before leaving, *pat* can make *terry* the owner of *pat:staff* by typing

% pts chown pat:staff terry

The **pts examine** command can be used to display ownership information about the group.

```
% pts examine terry:staff
Name: terry:staff, id: -534, owner: terry, creator: pat,
membership: 15, flags: SOm--, group quota: 0.>
```

Not only is *terry* the new owner, but the name of the group has changed to *terry:staff*. However, the names of any groups owned by *terry:staff* (the former *pat:staff*) remain *pat:group-name*.

Example: Changing a Group's Owner to Itself

The user *terry* could make *terry:dept* a self-owned group by typing

```
% pts chown terry:dept terry:dept
```

The **pts examine** command can be used to list ownership information about the group.

```
% pts examine terry:dept
Name: terry:dept, id: -678, owner: terry:dept, creator: pat,
  membership: 6, flags: SOm--, group quota: 0.
```

In this case, the name of the group does not change because the owner prefix of the new owner group is also *terry*. If *terry* then decides that *smith:cpa* should own *terry:dept*, he would type

```
% pts chown terry:dept smith:cpa
```

Displaying information about the group with the **pts examine** command shows that the *owner-name* of the group's name is now *smith*.

```
% pts examine smith:dept
Name: smith:dept, id: -678, owner: smith:cpa, creator: pat,
  membership: 15, flags: SOm--, group quota: 0.
```

The group name changed to *smith:dept*, because the *owner-name* of the new owner group is *smith*.

To Change a Group's Name – pts rename

You must specify the complete versions of the group's current and new names. At the command shell prompt, type

% **pts rename -oldname** <*old name*> **-newname** <*new name*>

where

old name specifies the complete name of the group whose name is to be changed.

new name
: specifies the complete name the group is to receive.

Example: Changing a Group's Name with the Same Owner

Suppose *terry* decides that the group *terry:dept* should be called *terry:cpa*. He would type

```
% pts rename terry:dept terry:cpa
```

Note that *terry* appeared before the colon in both the old name and the new name, even though that field did not change; the **pts chown** command would need to be used to change the owner of the group before the *owner-name* of the group could be changed. The group's new name can now be specified with the **pts examine** command.

```
% pts examine terry:cpa
Name: terry:cpa, id: -678, owner: terry:cpa, creator: pat,
  membership: 15, flags: SOm--, group quota: 0.
```

Example: Changing a Group's Name with a Different Owner

Suppose the group *pat:staff* owned a group named *pat:plans*. The name of the group *pat:staff* changed to *terry:staff* when *terry* became the owner of the group. However, the name of *pat:plans*, which is now owned by *terry:staff*, did not change accordingly. To change its name, *terry* could type

```
% pts rename pat:plans terry:plans
```

This command is possible because the owner group of *terry:plans* (*terry:staff*) has *terry* as its *owner-name* prefix. The command could also have been used to change the name from *plans* to something else at the same time. The **pts examine** command can be used to list information about the group, which is now named *terry:plans*.

```
% pts examine terry:plans
Name: terry:plans, id: -678, owner: terry:dept, creator: pat,
  membership: 6, flags: SOm--, group quota: 0.
```

Troubleshooting

Replace this page with the tab separator for

Troubleshooting

7. Troubleshooting

This chapter discusses some of the more common problems you may encounter when you work in AFS and some typical ways of solving these problems. To use this section, find the heading that corresponds to the problem you are having or the error message you received and follow the directions. If you have problems with any of these steps, talk with your system administrator.

This chapter includes the following sections:

7.1. Problem: Cannot Save File

Step 1: If you cannot save a file, check if the file servers are running by issuing the **fs checkservers** command (see section 4.3):

% **fs checkservers &**

- If the output reads

  `These servers are still down: fileserver_x`

  then you should check to see if the file you are attempting to save is on the downed file server by running the **fs whereis** command (see section 4.2):

  % **fs whereis [-path** <*dir/file path*>**+]**

  If your file is on a downed file server (for example, *fileserver_x*), then you must wait until that file server is running again before you can save the file. If it is not on a downed file server, refer to the next step.

- If the output reads

  `All servers are running.`

  then a file server crash is not the cause of your problem; refer to the next step.

Step 2: If all relevant file servers are running, check to make sure you have valid tokens by issuing the **tokens** command (see section 3.1.2):

% **tokens**

- If your tokens are invalid for this cell, expired, or non-existent, you must authenticate by issuing the **klog** command (see section 3.1.2):

  % **klog**

- If your tokens are valid, refer to the next step.

Step 3: If the file servers are running and you have tokens, you should check to see if you have exceeded your volume quota by issuing the **fs quota** command (see section 4.1). You cannot save a file if the new file will put the underlying volume over its volume quota.

% **fs quota [-path** <*dir/file path*>**+**]

- If your volume quota is above 95% used, you may be exceeding your quota when you attempt the save. You should delete unnecessary files or ask your system administrator to increase your quota.

- If your volume quota is below 95% used, you are probably not exceeding your quota (unless you are attempting to save a very big file or you have a very small quota). Refer to the next step.

Step 4: Although you have not exceeded your volume quota, the partition that houses the volume containing the destination directory of the file copy may have exceeded its quota. Issue the **fs listquota** command to find out (see section 4.1):

% **fs listquota [-path** <*dir/file path*>$^+$]

- If the partition is full (near 99%), contact your system administrator immediately.
- If the partition is not full, refer to the next step.

Step 5: If you have room to save the file, check to make sure that you have the required rights to save the file (that is, the WRITE right to save an existing file, the WRITE and INSERT rights to save a new file). Use the **fs listacl** command (see section 5.4):

% **fs listacl [-path** <*dir/file path*>]

- If you do not have the necessary rights, contact the owner of the directory or the system administrator to get the necessary access rights; you automatically have ADMINISTER rights on your home directory and its subdirectories.

  Note: Remember, if you are granted new access rights, you will need to re-authenticate (issue **klog**) to use those new rights.

- If you have the necessary rights, contact your system administrator. He or she has additional tools that may solve your problem.

7.2. Problem: Cannot Access a Directory or File

Step 1: If you cannot access a directory or file, check if the file servers are running by issuing the **fs checkservers** command (see section 4.3):

`%` **fs checkservers &**

- If the output reads

  `These servers are still down: fileserver_x`

  then you should check to see if the directory or file you are attempting to access is on a downed file server by running the **fs whereis** command (see section 4.2):

  `%` **fs whereis [-path** <*dir/file path*>$^{+}$**]**

  If your directory or file is on a downed file server (for example, *fileserver_x*), then you must wait until that file server is running again before you can access it. If it is not on a downed file server, refer to the next step.

- If the output reads

  `All servers are running.`

  then a file server crash is not the cause of your problem; refer to the next step.

Step 2: If all relevant file servers are running, check to make sure you have valid tokens by issuing the **tokens** command (see section 3.1.2):

`%` **tokens**

- If your tokens are invalid for this cell, expired, or non-existent, you must authenticate by issuing the **klog** command (see section 3.1.2):

  `%` **klog**

- If your tokens are valid, refer to the next step.

Step 3: Check to make sure that you have the required rights to access the directory or file by issuing the **fs listacl** command. You must have LOOKUP rights on a directory's ACL to access the files and subdirectories in any way. You automatically have ADMINISTER rights on your home directory and its subdirectories. (See section 5.2 for information on the other rights.)

`%` **fs listacl [-path** <*dir/file path*>]

- If you do not have the necessary rights, contact the owner of the directory or the system administrator to get the necessary access rights.

  Note: Remember, if you are granted new access rights, you will need to re-authenticate (issue **klog**) to use those new rights.

- If you have the necessary rights, contact your system administrator. He or she has additional tools that may solve your problem.

7.3. Problem: Cannot Copy a File

Step 1: If you cannot copy a file, check if the file servers are running by issuing the **fs checkservers** command (see section 4.3):

% **fs checkservers &**

- If the output reads

  ```
  These servers are still down: fileserver_x
  ```

  then you should check to see which file servers store the directories you want to copy to and from by by issuing the **fs whereis** command (see section 4.2):

  % **fs whereis [-path** <*dir/file path*>**+]**

  If either directory is on a downed file server (for example, *fileserver_x*), then you must wait until that file server is running again before you can copy the file. If neither directory is on a downed file server, refer to the next step.

- If the output reads

  ```
  All servers are running.
  ```

  then a file server crash is not the cause of your problem; refer to the next step.

Step 2: If all relevant file servers are running, check to make sure you have valid tokens by issuing the **tokens** command (see section 3.1.2):

% **tokens**

- If your tokens are invalid for this cell, expired, or non-existent, you must authenticate by issuing the **klog** command (see section 3.1.2):

  % **klog**

- If your tokens are valid, refer to the next step.

Step 3: If the file servers are running and you have tokens, you should check to see if you have exceeded your volume quota in the directory you are copying the file to by issuing the **fs quota** command (see section 4.1). You cannot copy a file if the copy will put you over your volume quota.

% **fs quota [-path** <*dir/file path*>**+**]

- If your volume quota is above 95% used, you may be exceeding your quota when you attempt the copy. You should delete unnecessary files or ask your system administrator to increase your quota.

- If your volume quota is below 95% used, you are probably not exceeding your quota (unless you are attempting to copy a very big file or you have a very small quota). Refer to the next step.

Step 4: Although you have not exceeded your volume quota, the partition that houses the volume containing the copy destination directory may have exceeded its quota. Issue the **fs listquota** command to find out (see section 4.1):

% **fs listquota [-path** <*dir/file path*>$^+$]

- If the partition is full (near 99%), contact your system administrator immediately.
- If the partition is not full, refer to the next step.

Step 5: If you have room to copy the file, check to make sure that you have the required rights to copy the file (that is, the READ right on the source directory, the INSERT right on the destination directory). Use the **fs listacl** command (see section 5.2):

% **fs listacl [-path** <*dir/file path*>]

- If you do not have the necessary rights, contact the owner of each directory or the system administrator to get the necessary access rights. You automatically have ADMINISTER rights on your home directory and its subdirectories.

  Note: Remember, if you are granted new access rights, you will need to re-authenticate (issue **klog**) to use those new rights.

- If you have the necessary rights, contact your system administrator. He or she has additional tools that may solve your problem.

7.4. Problem: Accidental Removal from an ACL

Step 1: If you accidentally removed yourself from an ACL, check the ACL to see if you have any remaining rights by issuing the **fs listacl** command (see section 5.2). You must have the LOOKUP right in order to issue this command or list the contents of a directory.

% **fs listacl [-path** <*dir/file path*>]

- If the output reads

  `fs: You don't have the required access rights on '.'`

  or

  `Access list for` <*dir/file path*> `is`

  then you do not have the necessary rights, contact the owner of the directory or the system administrator to get the necessary access rights. <u>You automatically have the ADMINISTER right on your home directory and its subdirectories, so if you remove yourself you can always restore your access rights</u>. Refer to the next step.

  Note: Remember, if you are granted new access rights, you will need to re-authenticate (issue **klog**) to use those new rights.

- If you have the necessary rights, contact your system administrator. He or she has additional tools that may solve your problem.

Step 2: If you do not have the necessary access rights to your home directory or one of its subdirectories, you can always add the rights. If someone else owns the directory, the owner or a system administrator will need to add the rights. Use the **fs setacl** command (see section 5.2):

% **fs setacl -dir** <*directory*>$^+$ **-acl** <*access list entries*>$^+$

Note: If you are restoring your own rights to a directory from which you removed all rights, you will need to specify the full pathname of the directory because without the LOOKUP right, pathname abbreviations are not valid.

7.5. Problem: Cannot Execute Commands

Step 1: If you cannot execute a command, you must first determine the directory location of that command by issuing the UNIX **which** command:

% **which** <*command*>

- If the output reads

  `<command>:  Command not found`

  then the file server(s) containing this command may be down; refer to the next step.

- If the output reads

  `/<pathname>/command`

  then refer to the next step.

Step 2: Check if the file servers are running by issuing the **fs checkservers** command (see section 4.3):

% **fs checkservers &**

- If the output reads

  ```
  These servers are still down:
  fileserver_x
  fileserver_y
  ```

  then you should check to see if the command you are attempting to execute is on the downed file server by running the **fs whereis** command (see section 4.2) and specifying the pathname containing the command you want to execute:

  % **fs whereis [-path** <*dir/file path*>$^+$**]**

  If all of the file servers containing the command are down (for example, *fileserver_x* and *fileserver_y*), then you must wait until at least one of the file servers is running again before you execute the command. If at least one of the file servers containing the command is running, refer to the next step.

- If the output reads

  `All servers are running.`

  then a file server crash is not the cause of your problem; refer to the next step.

Step 3: If at least one relevant file server is running, check to make sure you have valid tokens by issuing the **tokens** command (see section 3.1.2):

% **tokens**

- If your tokens are invalid for this cell, expired, or non-existent, you must authenticate by issuing the **klog** command (see section 3.1.2):

% **klog**

- If your tokens are valid, refer to the next step.

Step 4: If the file servers are running and you have tokens, you should check to see if you have the required access rights to the directory containing the command. You must have LOOKUP and READ rights on a directory's ACL to execute the commands the directory contains. Use the **fs listacl** command (see section 5.2):

% **fs listacl [-path** <*dir/file path*>]

- If you do not have the necessary rights, contact the owner of the directory or the system administrator to get the necessary access rights.

  Note: Remember, if you are granted new access rights, you will need to re-authenticate (issue **klog**) to use those new rights.

- If you have the necessary rights, refer to the next step.

Step 5: Check if your path variable contains the command's path by using the UNIX **echo** command:

echo $PATH

- If your path does not contain the command's path, you should add it to the path variable definition. Talk with your system administrator if you do not now how to do this.
- If your path contains the command's path, see you system administrator. He or she has other tools that may solve your problem.

7.6. Problem: Cannot Grant Access to a Directory

Step 1: If you cannot grant access to a directory, check if you have the ADMINISTER right on the ACL of that directory by issuing the **fs listacl** command (see section 5.2):

% **fs listacl [-path** <*dir/file path*>]

- If you do not have the ADMINISTER right, contact the owner of the directory or the system administrator to have the necessary right granted.

  Note: Remember, if you are granted new access rights, you will need to re-authenticate (issue **klog**) to use those new rights.

- If you have the necessary rights, refer to the next step.

Step 2: If you have the ADMINISTER right but cannot grant access to a directory, check to make sure you have valid tokens by issuing the **tokens** command (see section 3.1.2):

% **tokens**

- If your tokens are invalid for this cell, expired, or non-existent, you must authenticate by issuing the **klog** command (see section 3.1.2):

  % **klog**

- If your tokens are valid, see your system administrator. He or she has other tools that may solve your problem.

7.7. Diagnosing Common Error Messages

This section contains information on some of the more common error messages and suggestions for solving the associated problems.

Error Message: "afs: Lost contact with fileserver"

Check if there is a network problem or if the servers are restarting or rebooting by issuing the **fs checkservers** command (see section 4.3):

`%` **fs checkservers &**

- If the output reads

  ```
  These servers are still down:
  fileserver_x
  fileserver_y
  ```

  then you must wait until the file servers are back up. You can continue to work on files in your local cache.

- If the output reads

  ```
  All servers are running.
  ```

  then a file server crash is not the cause of your problem; see your system administrator.

Error Message: "<command>: Connection timed out"

Check if there is a network problem or if the servers are restarting or rebooting by issuing the **fs checkservers** command (see section 4.3):

`%` **fs checkservers &**

- If the output reads

  ```
  These servers are still down:
  fileserver_x
  fileserver_y
  ```

  then you must wait until the file servers are back up. You can continue to work on files in your local cache.

- If the output reads

  ```
  All servers are running.
  ```

  then a file server crash is not the cause of your problem; see your system administrator.

Error Message: "fs: You don't have the required access rights on "."

You do not have any access rights to the current directory. If you feel you should have access rights, contact the owner of the directory or the system administrator. If you are the owner of the directory and you accidentally removed yourself from the ACL, see section 7.4.

Error Message: "afs: failed to store file"

Step 1: Check if you have exceeded your volume quota by issuing the **fs quota** command (see section 4.1). You cannot copy or save a file if the new file causes the affected volume to exceed its quota.

% **fs quota [-path** <*dir/file path*>$^{+}$]

- If your volume quota is above 95% used, you may be exceeding your quota when you attempt the copy or save. You should delete unnecessary files or ask your system administrator to increase your quota.
- If your volume quota is below 95% used, you are probably not exceeding your quota (unless you are attempting to copy or save a very big file or you have a very small quota). Refer to the next step.

Step 2: Although you have not exceeded your volume quota, the partition that houses your volume (and other volumes) may have exceeded its quota. Issue the **fs listquota** command to find out (see section 4.1):

% **fs listquota**

- If the partition is full (near 99%), contact your system administrator immediately.
- If the partition is not full, see your system administrator. He or she has other tools that may solve your problem.

Appendix A: Command Reference

Replace this page with the tab separator for

Appendix A: Command Reference

Appendix A. Command Reference

This chapter describes in detail the AFS commands presented in the *AFS User's Guide*. It is intended for advanced AFS users. It does not contain explanations or procedures. If you have further questions, contact your System Administrator or refer to the *AFS System Administrator's Guide*.

A.1. AFS Command Reference Entries

Each entry in this chapter contains the following components:

- COMMAND NAME and a brief description of the command's function appears on the first line.
- COMPLETE COMMAND SYNTAX, specifying the required order for all arguments and flags, appears on the second line. The syntax matches the online help and uses the symbols described on page 2-2.
- ACCEPTABLE ABBREVIATIONS/ALIASES lists the shortest legal abbreviation for the command's operation code and switches. If the operation code has an alias, it also appears here. Note that any abbreviations longer than those shown are also legal.
- DESCRIPTION explains the command's function in detail.
- ARGUMENTS describes the function and required form of each argument and flag.
- WARNINGS indicates potential complications or damaging (that is, irreversible) effects of issuing the command. This section appears only when necessary.
- OUTPUT describes the on-screen output of the command, if any.
- EXAMPLES provides one or more sample commands and resulting output, if any.
- PRIVILEGE REQUIRED lists the privilege required to perform the command.
- MORE INFORMATION points to related commands.

A.2. About the fs Commands

Some **fs** commands extend UNIX file system semantics by invoking file-related functions that UNIX does not provide (setting access control lists, for example). Other **fs** commands help users control the performance of the Cache Manager running on their local client workstation. When using **fs** commands, pay particular attention to the kind of privilege required, as it varies from command to command.

A.2.1. Common Arguments and Flags on fs Commands

All **fs** commands accept the following optional flag. It is listed in the command descriptions and is described in detail here:

[-help]

This flag has the same function as the **fs help** command: it prints a command's online help message on the screen. No other arguments or flags should be provided at the same time as this flag. If they are, this flag overrides them, and the only effect of issuing the command is that the help message appears.

A.2.2. The Privileges Required for fs Commands

The privileges required for **fs** commands vary more than those required for commands in other suites. Pay special attention to the PRIVILEGE REQUIRED section of each command description.

The necessary privileges for the **fs** commands in this chapter include:

- Having certain rights on a directory's access control list. For example, creating and removing mount points requires ADMINISTER, INSERT, and DELETE rights for the directory in which the mount point resides.
- Belonging to the *system:administrators* group in the Protection Database. See the **fs delete** command for an example.
- No privilege. Many **fs** commands simply list information and so do not require any special privilege.

A.3. About the pts Commands

The **pts** command interface allows system administrators and regular system users to create, modify, and delete entries in the Protection Database by interacting with the Protection Server. Members of *system:administrators* can manipulate any user, machine, or group entry. Regular users may manipulate the group entries they own or for which they have the necessary privileges. Consult each command description to learn about necessary privileges.

A.3.1. File and Directory Protection under AFS

AFS augments and refines the standard UNIX scheme for controlling access to files and directories. Instead of relying only on the mode bits that define access rights for individual files, AFS associates an ***access control list (ACL)*** with each directory. The ACL lists the user and group access rights to the files in the associated directory. Each ACL applies to all of the files in its associated directory. Using ACLs allows AFS to define seven rights rather than just the standard three. (It is still possible to set the mode bits, but AFS interprets them in a different way; see the chapter on protection in the *AFS System Administrator's Guide* for details.)

Another refinement to the standard UNIX protection scheme is that users can define their own protection ***groups*** and then place the groups on ACLs as though they were individual users. This makes it easy to assign the same rights to many people simultaneously. Groups make it possible to add someone to many ACLS by adding them to a group that already exists on those ACLs (and it is just as easy to remove someone the same way). Machines can also be members of a group. When logged into a machine that is a member of a group, a user is given all of the access rights of that group.

A.3.2. The Protection Database

The Protection Database contains the data required to protect the files and directories in AFS file space. The majority of the **pts** commands are used to access or edit the information in these the Protection Database. A listing of the fields that make up each record in the Protection Database follows:

- ***name***. For users, this is the character string typed when logging in. For machines, the name is the IP (Internet Protocol) 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.

- ***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 the **pts listmax** and **pts setmax** commands for more information about the AFS UID counters.

- ***owner***. This is the user or group that owns the entry (that is, 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 the discussion of the **pts setfields** command for more information.

- ***creator***. The name of the user who originally created the entry using the **pts createuser** and **pts creategroup** command. This field is mainly useful as an audit trail, and cannot be changed.
- ***membership count***. For users and machines, this indicates how many groups the user/machine belongs to. For groups, it indicates how many members belong to the group. This number cannot be set explicitly.
- ***privacy flags***. This field indicates who is allowed to list certain information about the entry or change it in certain ways. See the discussion of the **pts examine** and **pts setfields** commands for more information.
- ***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 for machines is meaningless because it is not possible to authenticate as a machine. The quota is also meaningless for groups and should have the value `0`.

  Only members of *system:administrators* may reset a group-creation quota using the **pts setfields** command. 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.
- ***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 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*.
- ***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.

A.3.3. The Standard System Groups

In addition to the groups that users and administrators can create, AFS defines three standard groups:

- ***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 everyone able to access the local cell, authenticated or not, including people who have logged into a local machine as "root," or AFS users from foreign cells who have telnetted to a local machine.
- ***system:authuser***. This group also lacks a stable membership—its membership changes as people authenticate and unauthenticate in the local cell. It includes everyone who is currently authenticated (has a valid token) in the cell. This means that they have proved their identities to the Authentication Server by providing the correct passwords, either during login or by issuing the **klog** command at another time.

- ***system:administrators***. This group includes the few users authorized to administer the cell. Members of this group have implicit ADMINISTER rights on the ACL of every directory in the system. Only members of *system:administrators* can issue privileged **pts** commands. Because of the extensive power granted by membership in this group, it is recommended that only a few high-level administrators be included in this group.

When the Protection Server initializes the Protection Database for the first time in a cell, it automatically creates entries for these groups in the Protection Database and allocates them AFS UIDs. (The IDs assigned are standard and should not be altered, although they may be examined with **pts examine**.) The counter that the Protection Server uses when allocating AFS UIDs is set beyond these standard IDs so that there will be no conflict when other groups are created later.

In addition to having no stable membership, the *system:anyuser* and *system:authuser* groups are unique in other ways. It is not possible to use **pts adduser** to add members to these groups, nor can their membership be listed with **pts membership**. Like other groups, these two can appear on ACLs, which provides a convenient way to allow all users (or all authenticated users) certain rights on a directory. As each AFS volume is created, the ACL on its root directory is set to grant READ and LOOKUP rights to *system:anyuser*.

The *system:administrators* group does have a stable membership, which only current members of it may alter. The person who installs a cell's first file server machine should add himself or herself to this group or create a "dummy" administrative account while the Protection Server is still running in "no authorization checking" mode. See the *AFS Installation Guide* for details. It is possible to place *system:administrators* on ACLs, but that is generally not necessary because members of this group have implicit ADMINISTER rights on every directory.

A.3.4. Common Arguments and Flags used by pts Commands

All **pts** commands accept optional arguments and flags. They are listed in the command descriptions and are described here in detail:

[-cell <*cell name*>]

This argument indicates that the command should be run in the cell specified by *cell name*. The issuer may abbreviate *cell name* to the shortest form that distinguishes it from the other cells listed in */usr/vice/etc/CellServDB* on the client machine on which the command is issued. By default, commands are executed in the local cell as defined

- First, by the value of the environment variable AFSCELL. (This variable is normally undefined. If you are working in another, non-local cell for an extended period of time, you may want to define AFSCELL to be the name of that cell.)

- Second, in */usr/vice/etc/ThisCell* on the client machine on which the command is issued.

[-noauth]

This flag instructs the Protection Server not to authenticate the user of the command, and thus establishes an unauthenticated connection between the user and the Protection Server (the user is recognized as the unprivileged user *anonymous*). It is useful only when authorization checking is disabled on the file server machine (during the installation of a file server machine or when **bos setauth** has been used during other unusual circumstances). In normal circumstances, the Protection Server allows only authorized (privileged) users to issue commands that change the status of a server or configuration file, and will refuse to perform such an action even if the **-noauth** flag is used.

[-test]

This flag directs the **pts** command interpreter to refer to the *CellServDB* file in */usr/afs/etc*, not */usr/vice/etc*, where it normally would. This is useful when a developer is testing the Protection Server by running an isolated instance on his or her own workstation. By including only that workstation (which is not a database server machine for the cell) in */usr/afs/etc/CellServDB* and then using this flag, the developer makes sure that the test does not affect the real Protection Server and Database in use in the remainder of the cell.

[-force]

This flag directs the **pts** command interpreter to continue executing the command, if possible, even if it encounters problems during the command's execution. The command interpreter performs as much of the requested operation as possible, rather than aborting if it encounters a problem. The command interpreter reports any errors it encounters during the command's execution. This flag is especially useful if you list many arguments with a command's switch; if one of the arguments is invalid, the command reports the error and proceeds with the remaining arguments.

[-help]

This flag has the same function as the **pts help** command: it prints the command's online help message on the screen. No other arguments or flags should be provided at the same time. Even if they are, this flag overrides them, and the only effect of issuing the command is that the help message appears.

A.3.5. Privileges Required for pts Commands

To issue most of the **pts** commands on a user or group Protection Database entry, the user issuing the command must have the required privileges. The exact privileges required, as well as any conditions that must be met, are listed in the PRIVILEGE

REQUIRED section of each **pts** command listed in this chapter. Below is a summary of privilege requirements.

Members of the *system:administrators* group are permitted to issue privileged **pts** commands and use privileged command arguments on all user and group Protection Database entries. (Only someone who already belongs to *system:administrators* may add a user to the *system:administrators* group. See the **pts adduser** command.)

Regular users (that is, user that are not members of *system:administrators*) can always list the information associated with the group entries they own, as well as, their own user entries. Regular users are not permitted to use certain commands and command arguments reserved for members of the *system:administrators* group. Depending on how a group owner has defined the privacy flags field for his or her owned group, regular users may also be permitted to access and manipulate the Protection Database entry for that group. (The **pts setfields** command is used to define the privacy flags field.)

A.4. About Non-Command Suite Commands

This appendix also contains information on commands not associated with the AFS command suites (that is, **fs**, **pts**, etc.) These commands are used for a wide variety of purposes including logging in, authenticating, and unauthenticating. The commands described include

klog
knfs
kpasswd
pagsh
tokens
unlog

fs apropos — show each help entry containing keyword.

fs apropos -topic <*help string*> [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs ap -t <*help string*> **[-h]**

DESCRIPTION

Displays the first line of the online help entry for any **fs** command that has *help string* in its name or short description.

ARGUMENTS

-topic specifies the keyword string for which to search. If it is more than a single word, surround it with double quotes or other delimiters. This argument is case-sensitive; type *help string*s for **fs** commands in lowercase letters.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The first line of a command's online help entry names the command and briefly describes what it does. The **fs apropos** command displays that first line for any **fs** command where *help string* is part of the command name or first line.

To see the remaining lines in a help entry, which provide the command's alias (if any) and syntax, use the **fs help** command.

EXAMPLES

The following lists all **fs** commands that have the word "*cache*" in their operation codes or short online descriptions:

```
% fs apropos -topic cache
setcachesize: set cache size
flush: flush file from cache
getcacheparms: get cache usage info
monitor: set cache monitor host address
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs help

fs checkservers — check status of file server machines.

fs checkservers [-cell <*cell to check*>] [-all] [-fast] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs checks [-c <*cell to check*>**] [-a] [-f] [-h]**

DESCRIPTION

Lists any file server machines in each specified cell that meet two conditions:

1. The Cache Manager has been in contact with the *fileserver* process running on the machine, and/or may need to contact it in future. (Reasons for wanting to contact a file server machine might include holding a callback from that machine or having locked files on it.)

2. The *fileserver* process on the machine is not currently responding to Cache Manager probes (implying that it is not responding to Cache Manager file requests either).

The Cache Manager constantly maintains a list of file server machines that meet the first condition, updating it every four to ten minutes by attempting to contact the *fileserver* process on each machine in the list. When a process does not respond to the probe, the Cache Manager marks it as non-functioning. If a machine that previously did not respond begins to respond again, the Cache Manager erases the "not functioning" mark.

This command forces the Cache Manager to update its information immediately (rather than waiting the standard interval). The Cache Manager probes the *fileserver* process on the machines in the specified cell that meet the first condition above, records those that do not respond, and reports the result. If the issuer includes the **-fast** flag, the Cache Manager outputs the list it already has at the time the command is issued instead of probing the machines again.

By default, the Cache Manager probes machines in the local cell only. If the **-all** flag is used, it probes all machines (from all cells) that meet the first condition. If a cell name is specified with **-cell**, The Cache Manager probes the machines in that cell only. The **fs checkservers** command will not use the AFSCELL environment variable.

WARNING

It can take quite a while for this command to produce its entire output if a number of machines in the Cache Manager's list are down when the command is issued. The delay may occur because after issuing the probe the Cache Manager waits a standard timeout period before concluding that the *fileserver* is not responding; this allows for the possibility of slow cross-network communication. If it is important that the command shell prompt return quickly, the issuer may wish to put this command in the background. It is harmless to interrupt the command (with **Ctrl-C** or another interrupt signal).

This command is not guaranteed to check the status of all file server machines in a cell. The Cache Manager probes only those machines that meet the first condition mentioned earlier.

ARGUMENTS

-cell specifies the complete name of the cell whose file server machines the Cache manager should probe (shortened forms are not acceptable). Provide this argument or **-all**; it may be combined with **-fast**.

-all causes the Cache Manager to probe all machines that meet the first condition mentioned earlier. Provide this argument or **-cell**; it may be combined with **-fast**.

-fast tells the Cache Manager to display its current list of down machines, rather than probing any machines. The displayed output may be up to 10 minutes old.

-dir is obsolete, but can still be provided on the command line. Previous versions of this command required a directory argument. If the issuer includes it by accident, a warning message appears, but the command still executes correctly.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

If the Cache Manager gets a response from all of the machines that it probes (that is, all such machines are functioning normally), the output is

```
All servers are running.
```

(Remember that this message does not imply that all file server machines in the cell are running. It reports the status of only those that the Cache Manager tries to probe.)

If a machine fails to respond to the Cache Manager's probe within the timeout period, the output displays its name. The format of a machine name (name in uppercase, name in lowercase, or Internet address in four-field decimal form) depends on the state of the local cell's name server at the time the command is issued.

EXAMPLES

In the following example, the issuer chooses to see the Cache Manager's current list of down machines that belong to the local cell, rather than waiting for it to probe them again. The output indicates that all machines responded to the previous probe.

% **fs checks -f**
```
All servers are running.
```

The following example checks file server machines in all cells that the Cache Manager has previously contacted. It reports that the machines *fs1.transarc.com* and *vice3.andrew.cmu.edu* did not respond to the machine's probe.

% **fs checkservers -all &**
```
These servers are still down:
     fs1.transarc.com
     VICE3.ANDREW.CMU.EDU
```

The following example checks machines that the Cache Manager has previously contacted in the *athena.mit.edu* cell only:

% **fs checks athena.mit.edu &**
```
%All servers are running.
```

PRIVILEGE REQUIRED

None

fs cleanacl — remove obsolete entries from access control list.

fs cleanacl [-path <*dir/file path*>$^+$**] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

fs cl [-p <*dir/file path*>$^+$**] [-h]**

DESCRIPTION

Removes from the access control list of each specified directory or file any entries that specify a user or group no longer found in the Protection Database. When a user/group is removed from the Protection Database, its AFS UID appears on access control lists rather than its name. This command removes such "abandoned" AFS UIDs from access control lists.

Cleaning access control lists in this way not only keeps them from becoming crowded with irrelevant information, but also prevents the new possessor of a recycled AFS UID from obtaining access intended for the former possessor of the ID. (Note that recycling IDs is not recommended in any case.)

ARGUMENTS

-path specifies a file or directory for which the associated access control list is to be cleaned. If a filename is specified, the ACL of the file's parent directory is cleaned. If the issuer omits this switch, the current working directory is assumed.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

If there are no obsolete AFS UIDs on the ACL, the following message appears:

`Access list for` *dir/file path* `is fine.`

Otherwise, the output reports the resulting state of the ACL, following the header:

`Access list for` *dir/file path* `is now`

EXAMPLES

In the following example, the user *pat* cleans the ACL on the current directory and its subdirectories called *reports* and *sources*. The ACLs for the first two have no obsolete AFS UIDs on them, but *sources* does.

```
% fs cl . ./reports ./sources
Access list for . is fine.
Access list for ./reports is fine.
Access list for ./sources is now
Normal rights:
   system:authuser rl
   pat rlidwka
```

PRIVILEGE REQUIRED

Issuer must have ADMINISTER rights to the directory; by default, the owner of the directory and members of *system:administrators* do.

MORE INFORMATION

fs listacl

fs copyacl — copy access control list from one directory to one or more other directories.

fs copyacl -fromdir <*source directory*>
-todir <*destination directory*>+
[-clear] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs co -f <*source directory*> **-t** <*destination directory*>+ **[-c] [-h]**

DESCRIPTION

Copies the access control list (ACL) from the *source directory* to each *destination directory*. The command does not affect entries on the ACL of the *source directory*. It affects entries on the ACL of each *destination directory* as follows:

- If an entry is unique to the ACL of the *source directory*, it is copied to the ACL of the *destination directory*.
- If an entry exists on the ACLs of both directories, it is changed on the ACL of the *destination directory* to match the rights granted on the ACL of the *source directory*.
- If an entry is unique to the ACL of the *destination directory* and the **-clear** flag is omitted, the entry is not affected.
- If an entry is unique to the ACL of the *destination directory* and the **-clear** flag is included, the entry is removed.

Use the **-clear** flag to completely replace the ACL of each *destination directory* with that of the *source directory*.

ARGUMENTS

-fromdir specifies the *source directory* whose ACL is to be copied to each *destination directory*. Abbreviated pathnames are interpreted relative to the directory in which the command is issued. If a filename is provided, the file's parent directory is used as the *source directory*.

-todir specifies one or more *destination directories* to receive the ACL from the *source directory*. Abbreviated pathnames are interpreted relative to the directory in which the command is issued. A filename cannot be specified with this argument.

-clear removes all existing entries from the ACL of each *destination directory* before copying the ACL from the *source directory*. The ACL of each *destination directory* is thus completely replaced with the ACL of the *source directory*. If the issuer omits this flag, entries that exist on the ACL of a *destination directory* but not on the ACL of the *source directory* are not affected.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

EXAMPLES

The following example uses the **fs copyacl** command to copy the ACL from the current directory to the subdirectory named *reports*. Entries on the ACL of the current directory are not affected. Because the **-clear** option is not used, entries on the ACL of the *reports* directory that are not on the ACL of the current directory remain unaffected as well.

```
% fs la . reports
Access list for . is
Normal rights:
   pat rlidwka
   smith rlidwk

Access list for reports is
Normal rights:
   pat rl
   pat:friends rl
Negative rights
   jones rlidwka

% fs co . reports

% fs la . reports
Access list for . is
Normal rights:
   pat rlidwka
   smith rlidwk

Access list for reports is
Normal rights:
   pat rlidwka
   pat:friends rl
   smith rlidwk
Negative rights
   jones rlidwka
```

PRIVILEGE REQUIRED

Issuer must have the LOOKUP right to the *source directory* and the ADMINISTER right to each *destination directory*. To issue the command with a filename used for *source directory*, the issuer must have both the LOOKUP and READ rights on the ACL of the file's parent directory.

MORE INFORMATION

fs listacl

fs setacl

fs examine — show information about volume containing specified directory.

fs examine [-path <*dir/file path*>$^+$**] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

fs exa [-p <*dir/file path*>$^+$**] [-h]**

fs listvol [-p <*dir/file path*>$^+$**] [-h]**

fs lv [-p <*dir/file path*>$^+$**] [-h]**

DESCRIPTION

Displays information about the volume containing each specified directory or file. This information includes the file's quota and current size. See the OUTPUT section for a complete explanation of the information provided. While this command provides the most information about a volume, the **fs listquota** and **fs quota** commands are also available to display information about a volume.

ARGUMENTS

-path specifies each file and/or directory for which information about the host volume is desired. Omit this argument to display information about the volume that contains the current working directory.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The output reports the following information about each volume that contains a specified directory or file:

- The volume ID number (abbreviated in the output as "`vid`") of the volume.
- The volume's name.
- The current "offline" message associated with the volume, as set by a system administrator using the **fs setvol** command.
- The current "message of the day" associated with the volume, as set by a system administrator using the **fs setvol** command.
- The volume's maximum size quota, in kilobyte blocks.
- Its current size, in kilobyte blocks.

- The number of kilobyte blocks still available on the disk partition that houses the volume and the partition's total size.

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 main reason is that 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 5 minutes old, as the Cache Manager polls the File Server for partition information at that frequency. Another potential difference: 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 is likely to be about 10% larger.

EXAMPLES

The following shows the output for the volume `user.smith` (and the partition housing it) in the Transarc Corporation cell:

```
% fs exa /afs/transarc.com/usr/smith
Volume status for vid = 50489902 named user.smith
Current maximum quota is 15000
Current blocks used are 5073
The partition has 46383 blocks available out of 333305
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs listquota

fs quota

fs setquota

fs help — show syntax of specified **fs** commands or list functional descriptions of all **fs** commands.

fs help [-topic <*help string*>$^+$] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs h [-t <*help string*>$^+$] [-h]

DESCRIPTION

Displays the first line (name and short description) of every **fs** command's online help entry if no *help string* is provided. For each operation code specified with **-topic**, it outputs the entire help entry. See the OUTPUT section.

ARGUMENTS

-topic specifies the operation codes for which syntax is to be provided. If the issuer omits this argument, the output instead provides a short description of all **fs** commands.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The online help entry for each **fs** command consists of two or three lines:

- The first line names the command and briefly describes what it does.
- The second line displays any aliases the command has (this line does not appear for every command).
- The final line, which begins with "`Usage:`," lists the command's arguments and flags in the prescribed order. Online help entries use the same symbols (for example, brackets) as the command definitions in this manual. For an explanation of their meaning, see chapter 2.

EXAMPLES

The following displays the online help entry for the **fs setacl** command:

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

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs apropos

fs listacl — show access control list.

fs listacl [-path <*dir/file path*>$^+$] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs la [-p <*dir/file path*>$^+$**] [-h]**

DESCRIPTION

Displays the access control list (ACL) associated with each *directory*. It is legal to provide a filename rather than a directory name for *directory*, in which case the ACL of the file's parent directory is displayed (because it is not possible to set an ACL for an individual file, the file is inheriting the ACL from its parent directory). Omit this argument to display the ACL of the current working directory.

Users who possess the ADMINISTER right on an ACL may change the ACL with the **fs setacl** command or copy the ACL from a different directory to it with the **fs copyacl** command.

WARNING

The appearance of a user/group on the `Negative rights` list does not guarantee that the person is denied those rights. If *system:anyuser* is granted any rights on the `Normal rights` list, a user need only issue **unlog** to obtain those rights.

ARGUMENTS

-path specifies each file and/or directory for which to display the associated ACL. If this argument is omitted, the output displays the ACL associated with the current working directory. If it is a filename, the ACL displayed is associated with the file's parent directory.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The first line of the output names the directory associated with the access control list. If the issuer used shorthand notation (such as "." for the current directory) when indicating the *directory*, it may appear here rather than the full pathname of the directory.

The "`Normal rights:`" header indicates the list of users who have normal rights to the directory. Each additional line lists a user/group name and the set of rights the user/group may exercise. The possible rights and their meanings are listed below:

- `r` = READ the contents of files in the directory.
- `w` = WRITE modify the contents of files in the directory.
- `l` = LOOKUP status information about the files in the directory.
- `d` = DELETE files from the directory.
- `i` = INSERT new files into the directory.
- `k` = LOCK set read or write locks on the files in the directory.
- `a` = ADMINISTER change the rights on the access control list.
- `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H`
  By default, these have no meaning to AFS server processes. Administrators and application programs may assign meanings to them and place them on ACLs to control access to the directory's contents in new ways. The letters must be uppercase.

A "`Negative rights:`" header may appear next, if any negative rights have been specified for this directory. The format of this list is the same as that of the `Normal rights` list. The difference is that each user/group listed is <u>denied</u> rather than granted the specified rights.

EXAMPLES

The following displays the ACL associated with user *pat*'s home directory and its *private* subdirectory when the **fs listacl** command is issued in the home directory:

```
% fs la . private
Access list for . is
Normal rights:
   system:authuser rl
   pat rlidwka
   pat:friends rlid
Negative rights:
   smith rlidwka

Access list for private is
Normal rights:
   pat rlidwka
```

PRIVILEGE REQUIRED

To issue this command with a directory name argument, the issuer must have the LOOKUP right on the directory's ACL. To issue this command with a filename argument, the issuer must have both the LOOKUP and READ rights on the ACL of the file's parent directory.

MORE INFORMATION

fs cleanacl

fs copyacl

fs setacl

fs listcells — show database server machines in cell(s) known to Cache Manager.

fs listcells [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs listc [-h]

DESCRIPTION

Formats and displays the Cache Manager's kernel-resident list of the database server machines in its home cell and foreign cells. At each reboot of the workstation, the Cache Manager copies the contents of */usr/vice/etc/CellServDB* into the kernel. It is possible to modify the kernel-resident list between reboots using **fs newcell**.

ARGUMENTS

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The output contains a line for each cell for which the kernel has a list of database server machines. The cell name is followed by a list of its database server machines (referred to as "hosts"). The format of each machine name (name in uppercase, name in lowercase, or Internet address in four-field decimal form) depends on the state of the local cell's name server at the time the command is issued.

EXAMPLES

The following shows output for several cells as illustrations of the different formats for machine names:

```
% fs listc
Cell transarc.com on hosts fs1.transarc.com fs2.transarc.com
Cell andrew.cmu.edu on hosts VICE11.FS.ANDREW.CMU.EDU
   VICE2.FS.ANDREW.CMU.EDU VICE7.FS.ANDREW.CMU.EDU.
Cell athena.mit.edu on hosts 18.80.0.2 orf.mit.edu
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs newcell

fs listquota — show quota information for the volume containing a file/directory.

fs listquota [-path <*dir/file path*>$^+$] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs lq [-p <*dir/file path*>$^+$**] [-h]**

DESCRIPTION

Displays information about the size and quota of the volume containing each specified directory or file. See the OUTPUT section for a complete explanation of the information provided.

ARGUMENTS

-path specifies each file and/or directory for which information about the host volume is desired. If the issuer omits this argument, the current directory is assumed.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The output reports the following information about each volume that contains a specified directory or file:

- The name of the volume.
- Its maximum size quota, in kilobytes.
- Its current size, in kilobytes.
- The percentage of its quota that its current size represents.
- The percentage of the volume's disk partition that is full. This is usually unrelated to how much of the user's quota is used, since it depends on all the volumes on the partition. A large value may nevertheless prevent a user from being able to store more data on the partition.

EXAMPLES

The following shows the output for the volume `user.smith` in the Transarc Corporation cell:

```
% fs lq /afs/transarc.com/usr/smith
Volume Name      Quota    Used    % Used   Partition
user.smith       15000    5071       34%         86%
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs diskfree

fs examine

fs quota

fs setquota

fs setvol

fs quota — show percent of quota used for volume containing directory/file.

fs quota [-path <*dir/file path*>$^+$] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs q [-p <*dir/file path*>$^+$**] [-h]**

DESCRIPTION

Displays the percent of maximum quota currently used by the volume that contains each specified directory or file. This is the least informative but quickest **fs** command that provides quota information about a volume. The **fs examine** and **fs listquota** commands provide more complete information. Only the system administrator may set quota.

ARGUMENTS

-path specifies each file and/or directory for which quota information about the host volume is desired. If the issuer omits this argument, the current directory is assumed.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The output reports the percent of quota used. It does not name the host volume.

EXAMPLES

The following lists the percent quota used of the volume housing the current working directory:

```
% fs quota
17% of quota used.
```

The following lists the percent quota used of both the volume housing the current working directory's parent directory and the volume housing the directory named */afs/transarc.com/usr/smith*:

```
% fs quota .. /afs/transarc.com/usr/smith
43% of quota used.
92% of quota used.
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs examine

fs listquota

fs setquota

fs setvol

fs setacl — sets access control list for a directory.

fs setacl -dir <*directory*>+ **-acl** <*access list entries*>+ **[-clear] [-negative] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

fs sa -d <*directory*>+ **-a** <*access list entries*>+ **[-c] [-n] [-h]**

DESCRIPTION

Puts the specified *access list entries* on the access control list (ACL) of each specified *directory*. Only user and group entries can be placed on an ACL. Machine entries (IP addresses) cannot be placed directly on an ACL; instead, the machine entry must be made a member of a group and the group can then be put on the ACL.

WARNING

If the ACL already grants certain rights to a user or group, the rights specified with *access list entries* replace them, rather than just being added to them.

Setting negative rights is generally unnecessary and not recommended. Simply omitting a user or group from the `Normal rights` list is normally adequate to prevent access. In particular, note that it is futile to deny rights that are granted to *system:anyuser* on the same ACL; all the user needs to do is issue the **unlog** command to receive the denied rights.

ARGUMENTS

-dir specifies each directory for which the access control list is to change. Abbreviated pathnames are interpreted relative to the directory in which the command is issued.

-acl defines a list of one or more entries, in the following order separated by a space:

- A user name or group name (in lowercase letters).
- The access right(s) to be associated with the user/group.

This argument is unusual in requiring two parts for each instance. The accepted abbreviation of each right and the meaning of the right follows:

`r` READ. Allows the possessor to read the contents of files in the directory and to "stat" (issue **ls -l** for) file and subdirectory elements in the directory.

`w` WRITE. Allows the possessor to modify the contents of files in the directory and to change their UNIX mode bits with **chmod**.

`l` LOOKUP. Allows the possessor to list the names of files and subdirectories in the directory (for example, by issuing **ls**). The possessor may "stat" (issue **ls -l** for) the directory itself (but not for files and subdirectories in it) and may examine the directory's ACL.

`d` DELETE. Allows the possessor to remove files from the directory.

`i` INSERT. Allows the possessor to create new files in the directory or move existing files into it.

`k` LOCK. Allows the possessor to run programs that need to issue the "flock" system call on files in the directory.

`a` ADMINISTER. Allows the possessor to change the directory's ACL.

`A`, `B`, `C`, `D`, `E`, `F`, `G`, *H*;
By default, these have no meaning to AFS server processes. Administrators and application programs may assign meanings to them and place them on ACLs to control access to the directory's contents in new ways. The letters must be uppercase.

`all` all seven standard rights (*rlidwka*).

`none` no rights. Removes the user/group from the ACL, but may not guarantee they have no rights if they belong to groups that remain on the ACL.

`read` both *r* and *l*.

`write` all rights except ADMINISTER (*rlidwk*).

It is legal to mix the individual letters and the words within *access list entries*, but not within an individual pairing of user/group and rights.

-clear removes all existing entries on each access control list before placing *access list entries* on it. This should be used with caution: if *access list entries* does not grant all rights to the owner of the directory, it can become awkward for the owner to access items in the directory. In particular, not having the LOOKUP right makes it impossible to resolve the (`.`) and (`..`) shorthand from within the directory.

-negative puts the specified *access list entries* in the `Negative rights` section of each access control list. The user/group is thus explicitly denied the indicated rights, even if entries on the accompanying `Normal rights` section of the access control list grant them rights. However, it is possible to unlog to obtain rights granted to *system:anyuser* on the `Normal rights` section of the same ACL; see the WARNING above.

This flag affects all directories and access list entries specified. Its use is not recommended; see the WARNING section above. If the issuer omits this flag, the *access list entries* go into the `Normal rights` section of the access control list.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

EXAMPLES

The following example adds two entries to the `Normal rights` part of the current working directory's ACL: the first entry grants READ and LOOKUP rights to *pat:friends*, while the other (using the **write** shorthand) gives all rights except ADMINISTER to *smith*.

% fs sa . pat:friends rl smith write

The following shows the effect of the **-clear** flag on the ACL of the subdirectory *reports* by showing the ACL before and after the command is issued:

```
% fs la reports
Access list for reports is
Normal rights:
   system:authuser rl
   pat:friends rlid
   smith rlidwk
   pat rlidwka
Negative rights:
   terry rl

% fs sa -clear reports pat all smith write system:anyuser rl
% fs la reports
Access list for reports is
Normal rights:
   system:anyuser rl
   smith rlidwk
   pat rlidwka
```

The following shows how the **-dir** and **-acl** switches are necessary when more than one directory is specified. The new entry granting READ, LOOKUP, and INSERT rights to *pat:friends* is added to the ACL for both the current directory and its *public* subdirectory.

% fs sa -d . public -a pat:friends rli

PRIVILEGE REQUIRED

Issuer must have ADMINISTER rights to the directory; the directory's owner and members of *system:administrators* always do.

MORE INFORMATION

fs copyacl

fs listacl

fs whereis — report name of file server machine(s) housing specified file/directory.

fs whereis [-path <*dir/file path*>$^+$**] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

fs whe [-p <*dir/file path*>$^+$**] [-h]**

DESCRIPTION

Returns the name of the file server machines that house each specified directory or file. See the OUTPUT section for a description of the information displayed.

ARGUMENTS

-path specifies each file or directory whose location is to be returned. Each specified file or directory must reside in AFS (not on a local disk). If the issuer omits this argument, the location of the working directory is returned.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The output includes a line for each specified directory or file. It names the file server machine on which the volume that houses the specified directory or file resides. A list of multiple machines indicates that the directory or file is in a replicated volume.

Machine names usually have a suffix indicating their cell membership. If some question remains, the **fs whichcell** command names the cell in which a directory or file resides.

EXAMPLES

The following indicates that the directory */afs/transarc.com* resides in a replicated volume located on both *fs1.transarc.com* and *fs3.transarc.com*:

```
% fs whe /afs/transarc.com
File /afs/transarc.com is on hosts fs1.transarc.com
  fs3.transarc.com
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs whichcell

fs wscell

fs whichcell — return name of cell to which specified file/directory belongs.

fs whichcell [-path <*dir/file path*>$^+$**] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

fs whi [-p <*dir/file path*>$^+$**] [-h]**

DESCRIPTION

Returns the name of the cell in which the volume that houses each indicated directory or file resides. See the OUTPUT section for a description of the information displayed.

ARGUMENTS

-path specifies each file and/or directory whose cell membership is to be returned. Each specified directory or file must reside in AFS (not on a local disk). If the issuer omits this argument, the cell of the working directory is returned.

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The output includes a line for each specified directory or file, naming the cell in which the directory or file resides.

EXAMPLES

The following shows that the current directory resides in a volume in the Transarc Corporation cell:

```
% fs whi
File . lives in cell 'transarc.com'
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs whereis

fs wscell

fs wscell — return name of cell to which workstation belongs.

fs wscell [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

fs ws [-h]

DESCRIPTION

Returns the name of the home cell for the workstation from which the command is issued.

ARGUMENTS

-help prints the online help entry for this command. Do not provide any other arguments or flags with this one. See section A.2.1 for more details.

OUTPUT

The reported cell name comes from the file */usr/vice/etc/ThisCell* on the workstation's local disk.

EXAMPLES

The following results when the **fs wscell** is issued on a machine in the Transarc Corporation cell:

```
% fs wscell
This workstation belongs to cell 'transarc.com'
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

fs whereis

fs whichcell

kas examine — display information from an Authentication Database entry.

kas examine -name <*name of user*>
[-admin_username <*admin principal to use for authentication*>**]**
[-password_for_admin <*admin password*>**] [-cell** <*cell name*>**]**
[-servers <*explicit list of authentication servers*>$^+$**] [-noauth]**
[-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

kas e -na <*name of user*> **[-a** <*admin principal to use for authentication*>**]**
[-p <*admin password*>**] [-c** <*cell name*>**]**
[-s <*explicit list of authentication servers*>$^+$**] [-no] [-h]**

DESCRIPTION

Formats and displays information from the Authentication Database entry for *name*. See the OUTPUT section for details.

ARGUMENTS

-name	specifies the Database entry from which to display information.
-admin_username	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.
-password_for_admin	specifies the issuer's password. If provided here, the password is visible on the screen. If the issuer does not provide it, it will be prompted for and not be visible on the screen.
-cell	specifies the cell in which to run the command, if not the local cell.
-servers	specifies the database server machine(s) with which to establish a connection.
-noauth	establishes an unauthenticated connection between the Authentication Servers and issuer, whom they assign the unprivileged identity *anonymous* rather than attempting mutual authentication.
-help	prints the online help for this command. Do not provide any other arguments or flags with this one.

OUTPUT

The output reports, in this order:

- The name of the entry.
- One or more status flags, which will only appear if a system administrator has used the **kas setfields** command to change a flag from its default value. A plus sign (+) will separate the flags if more than one appears. The non-default values which may appear, and their meanings, are:
-
  - `ADMIN`. the user is allowed to issue privileged **kas** commands (Default: `NOADMIN`.)
  - `NOTGS`. the Ticket Granting Service will refuse to issue tickets to the user (Default: `TGS`.)
  - `NOSEAL`. the Ticket Granting Service cannot use the contents of this entry's key field as an encryption key (Default: `SEAL`.)
  - `NOCPW`. the user or server cannot change his/her/its own password or key (Default: `CPW`.)
- The word "`key`" followed by the key checksum in parentheses.

  The octal key itself appears only if authorization checking is disabled on the database server machine to which the **kas examine** command is directed with the **-servers** argument (see the EXAMPLES section). The reasoning behind this requirement is two-fold. First, it implies that only someone authorized to issue the **bos setauth** command or with "root" access to the database server machine's local disk is able to see actual keys from the Authentication Database. Second, it makes it clear that the system is in a compromised state of security while keys are being displayed on the screen. Both turning off authorization checking and displaying keys on a screen are serious security risks.

  In the normal cases when authorization checking is enabled on the database server machine, a "checksum" appears instead of the key. This is a decimal number derived by encrypting a constant with the key. In the case of the "*afs*" key, this number can be compared to the checksum with the corresponding key checksum in the output of the **bos listkeys** command.
- The date that the user changed his/her own password, indicated as "`last cpw`" (which stands for "last change of password")
- The expiration date of the password. If an expiration date appears here, the user must change his or her password before that expiration date. A regular user cannot authenticate using an expired password; a system administrator can get an admin ticket for up to 30 days after expiration. If there is no expiration date on the password, the entry will say: `password will never expire`. The password expiration date is set using the **kas setfields** command.

- The limit on the number of consecutive failed authentication attempts permitted before the user account will lock. This number is set by the **kas setfields** command.
- The lock out time for the user account if the limit on the number of consecutive failed authentication attempts is exceeded. (This entry will only appear if there is a limit on the number of failed authentication attempts.) The lockout time is set by the **kas setfields** command.
- The lock state of the user account. (This entry will only appear if there is a limit on the number of failed authentication attempts.) If the account is locked, the entry will say `User is locked until` *day/date/time*. If the account is not currently locked, the entry will say `User is not locked`. A system administrator can unlock a locked account by issuing the **kas unlock** command.
- The date on which the entry expires. If this is a user entry, the user will be unable to authenticate with the Authentication Server after this date.
- The maximum length of time that tickets issued for this entry may be valid
- The date of the last modification to the entry, indicated as "`last mod`," and the user name of the person who issued the modifying command. Password changes made by the user himself/herself are recorded as "`last cpw`" instead.

EXAMPLES

The following shows the privileged user *smith* examining her own Authentication Database entry. Note the `ADMIN` flag, which shows that *smith* is privileged.

```
% kas e smith
Password for smith:
User data for smith (ADMIN)
 key (0) cksum is 3414844392,  last cpw: Thu Dec 23 16:05:44 1993
 password will expire:  Fri Jul 22 20:44:36 1994
 5 consecutive unsuccessful authentications are permitted.
 The lock time for this user is 25.5 minutes.
 User is not locked.
 entry never expires. Max ticket lifetime 100.00 hours.
 last mod on Thu Jul 1 08:22:29 1993 by admin
```

In the following example, the user *erz* examines her Authentication Database entry to determine if her account is locked. Sure enough, it is.

```
% kas e erz
Password for erz:
User data for erz
 key (0) cksum is 73829292912,  last cpw: Wed Dec 8 11:23:01 1993
 password will expire:  Sat Jul 9 11:23:01 1994
 5 consecutive unsuccessful authentications are permitted.
 The lock time for this user is 25.5 minutes.
 User is locked until Tue Sep 20 12:25:07 1993
 entry expires on never. Max ticket lifetime 100.00 hours.
 last mod on Thu Jul 1 08:22:29 1993 by admin
```

In the following an administrator logged in as the privileged user *admin* uses **bos setauth** to turn off authorization checking on the database server machine *db1.transarc.com* so that he can look at the key in the *afs* entry. He enters interactive mode to open a connection with the Authentication Server on *db1.transarc.com* only and uses the **-noauth** flag to prevent that server from attempting to authenticate him.

```
% bos setauth db1.transarc.com off
% kas i -servers db1.transarc.com -noauth
ka> examine afs -servers db1.transarc.com
User data for afs
 key (12): \357\253\304\352a\236\253\352, last cpw: no date
 entry never expires. Max ticket lifetime 100.00 hours.
 last mod on Thu Jan 11 14:53:29 1990 by admin
```

PRIVILEGE REQUIRED

A user may examine his or her own entry. To examine others' entries, the issuer must have the `ADMIN` flag set in his or her Authentication Database entry.

To look at actual keys, authorization checking must be disabled on the database server machine with **bos setauth**, which implies being listed in */usr/afs/etc/UserList*; it is not necessary to have the `ADMIN` flag in addition.

klog — authenticate with Authentication Server to obtain token.

klog [-x] [-principal <*user name*>**]**
[-password <*user's password*>**] [-tmp]**
[-cell <*cell name*>**]**
[-servers <*explicit list of servers*>$^+$**]**
[-pipe] [-lifetime <*ticket lifetime in hh[:mm[:ss]]*>**]**
[-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

klog [-x] [-pr <*user name*>**] [-pa** <*user's password*>**] [-t]**
[-c <*cell name*>**] [-s** <*explicit list of servers*>$^+$**]**
[-pi] [-l <*ticket lifetime in hh[:mm[:ss]]*>**] [-h]**

DESCRIPTION

Authenticates *user name* in the specified cell. By default, the issuer of the command is *user name*. If the issuer of the command successfully identifies *user's password*, he or she obtains a token accepted by the AFS server processes in that cell. This command does not change the identity under which the issuer is logged into the local UNIX file system.

The Cache Manager acting on behalf of *user name* stores the token in a credential structure associated with *user name*. If *user name* already has a token for the cell, the token resulting from this command replaces the existing token in the credential structure.

Note: If you are using the Kerberos version of this command (that is, *klog.krb*), the **klog** command will also write a copy of *user name*'s ticket-granting ticket to the */tmp* directory on the local machine. If you have defined the environment variable KRBTKFILE, the ticket-granting ticket will be written to the directory specified by that variable.

By default, the token generated by **klog** is appropriate for the local cell (the one to which the local machine belongs). The command interpreter contacts an Authentication Server in the local cell, chosen at random from the list in */usr/vice/etc/CellServDB*, and requests a token for the issuer logged into the local machine.

Use the **-principal** argument to acquire a token for a *user name* other than the name under which you are identified by the system. Use the **-cell** argument to acquire a token for a cell other than the local cell. Use the **-servers** argument to specify a list of authentication servers to be used by the command interpreter to obtain the token.

The issuer (or user indicated with **-principal**) does not have to appear in the local password file (*/etc/passwd* or equivalent) to issue this command; in previous versions of this command, users had to add the **-x** flag if they did not appear in that file.

During a single login on a given machine, a user can be authenticated in multiple cells simultaneously; however, a single user can have only one token at a time for a given cell per login session or machine (that is, he or she can only authenticate under one identity per cell).

The lifetime of the token resulting from this command is the smallest of the following:

- The lifetime requested by the issuer with the **-lifetime** argument. If the issuer does not include this argument, the value defaults to 720 hours (30 days).
- 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).

If none of these defaults have been changed, then the standard token lifetime is 25 hours for users whose Authentication Database entries were created by the AFS 3.1 or later version of the Authentication Server, and 100 hours for users whose Authentication Database entries were created by the AFS 3.0 version of the Authentication Server. The user can issue **klog** to request a token with a different lifetime.

The maximum lifetime for any token is 720 hours (30 days), and the minimum is 5 minutes. Between these values, token lifetimes are not granted on a linear scale; only certain values are possible.

Lifetimes between 5 minutes and 10 hours 40 minutes are granted at 5 minute intervals, rounding up. For example, if the issuer requests a lifetime of 12 minutes, the token's actual lifetime is 15 minutes.

For token lifetimes greater than 10 hours 40 minutes, consult the following table, which presents all the possible times in units of *hours:minutes:seconds*. The number in parentheses is an approximation of the corresponding time in days and hours (*d h*). For example, 282:22:17 means 282 hours, 22 minutes, and 17 seconds, which translates to approximately 11 days and 18 hours (11d 18h).

If the issuer requests a lifetime between two listed values, the token's lifetime is rounded up to the next higher value. For example, if 11:24:16 was specified, the **klog** command would use 12:11:34.

11:24:15 (0d 11h)	33:14:21 (1d 09h)	96:52:49 (4d 00h)	282:22:17 (11d 18h)
12:11:34 (0d 12h)	35:32:15 (1d 11h)	103:34:45 (4d 07h)	301:53:45 (12d 13h)
13:02:09 (0d 13h)	37:59:41 (1d 13h)	110:44:28 (4d 14h)	322:46:13 (13d 10h)
13:56:14 (0d 13h)	40:37:19 (1d 16h)	118:23:54 (4d 22h)	345:05:18 (14d 09h)
14:54:03 (0d 14h)	43:25:50 (1d 19h)	126:35:05 (5d 06h)	368:56:58 (15d 08h)
15:55:52 (0d 15h)	46:26:01 (1d 22h)	135:20:15 (5d 15h)	394:27:37 (16d 10h)
17:01:58 (0d 17h)	49:38:40 (2d 01h)	144:41:44 (6d 00h)	421:44:07 (17d 13h)
18:12:38 (0d 18h)	53:04:37 (2d 05h)	154:42:01 (6d 10h)	450:53:46 (18d 18h)
19:28:11 (0d 19h)	56:44:49 (2d 08h)	165:23:50 (6d 21h)	482:04:24 (20d 02h)
20:48:57 (0d 20h)	60:40:15 (2d 12h)	176:50:01 (7d 08h)	515:24:22 (21d 11h)
22:15:19 (0d 22h)	64:51:57 (2d 16h)	189:03:38 (7d 21h)	551:02:38 (22d 23h)
23:47:38 (0d 23h)	69:21:04 (2d 21h)	202:08:00 (8d 10h)	589:08:45 (24d 13h)
25:26:21 (1d 01h)	74:08:46 (3d 02h)	216:06:35 (9d 00h)	629:52:56 (26d 05h)
27:11:54 (1d 03h)	79:16:23 (3d 07h)	231:03:09 (9d 15h)	673:26:07 (28d 01h)
29:04:44 (1d 05h)	84:45:16 (3d 12h)	247:01:43 (10d 07h)	
31:05:22 (1d 07h)	90:36:53 (3d 18h)	264:06:34 (11d 00h)	

WARNING

This command does not create a new PAG (process authentication group); see the description of the **pagsh** command in this chapter to learn about PAGs. Users in cells not using the AFS version of **login** should issue **pagsh** before issuing this command, so that the tokens get stored in a credential structure that is identified by PAG rather than UNIX UID.

A potential security problem exist when a token is associated with a credential structure identified by a UNIX UID because anyone logged in as "root" can assume any other identity by issuing the **su** command. And when "root" assumes another UNIX UID it can use use any tokens associated with that UID. Use of a PAG as an identifier eliminates this problem.

If the issuer entered the current session by issuing the AFS **login** command, his or her credential structure is already identified by a PAG. Issuing **klog** during the same session creates a new token associated with the existing PAG.

ARGUMENTS

-x appears only for backwards compatibility. Its former function is now the default behavior of this command, as mentioned in the DESCRIPTION section.

-principal
is the user name under which the issuer wishes to authenticate. By default, the Authentication Server attempts to authenticate the user logged into the local machine's UNIX file system. This argument allows the issuer to specify a different user name.

-password
specifies the issuer's password (or that of *user name* if **principal** is provided). Use of this argument is strongly discouraged, as it makes the password visible on the command line. If the issuer omits this argument, **klog** prompts for the password and does not echo it visibly:

Password: <*user's password***>**

-tmp indicates that a copy of the ticket-granting ticket used to create the token should be placed in a file on the local machine's */tmp* directory; the file has the format */tmp/tkt**UNIX_UID***. (For example for user with UNIX UID 1000, the file would be named */tmp/tkt1000*).

The ticket-granting ticket is an intermediate ticket that the Ticket Granting Service requires of clients who desire server tickets (the extra level of indirection increases security). Putting the ticket-granting ticket into */tmp* allows standard Kerberos applications to access it and use it for obtaining server tickets. If this flag is omitted, the Cache Manager discards the ticket-granting ticket after it obtains the AFS server ticket. Do not use this flag with the *klog.krb* version of this command because, by default, *klog.krb* automatically writes the ticket-granting ticket to */tmp*.

-cell specifies the cell in which the issuer wishes to authenticate, by directing the command to that cell's Authentication Servers. During a single login on a given machine, a user may be authenticated in multiple cells simultaneously, but can have only one token at a time for each of them (that is, can only authenticate under one identity per cell per machine). The issuer may abbreviate *cell name* to the shortest form that distinguishes it from the other cells listed in */usr/vice/etc/CellServDB* on the client machine on which the command is issued.

If this argument is omitted, the command is executed in the local cell, as defined

- First, by the value of the environment variable AFSCELL.
- Second, in */usr/vice/etc/ThisCell* on the client machine on which the command is issued.

-servers causes the command interpreter to establish a connection with the Authentication Server running on each specified database server machine. It then chooses one of these at random to execute the command. The command accepts shortened machine names, but exactly which abbreviations are acceptable depends on the state of the cell's name server at the time the command is issued. This option is useful for testing specific servers if problems are encountered.

If this argument is omitted, the command interpreter establishes a connection with each machine listed for the indicated cell in the local workstation's copy of */usr/vice/etc/CellServDB*, and then chooses one of those at random for command execution.

-pipe indicates that the command should run without printing anything on the screen, including prompts or error messages. The **klog** command interpreter Server expects to receive the password from standard input (stdin). The issuer is discouraged from using this argument; it is for use by application programs rather than human users.

-lifetime indicates the lifetime that the issuer wishes the token to have. The value provided is considered in the lifetime calculation described in the DESCRIPTION section, along with the maximum ticket lifetimes mentioned there.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

EXAMPLES

Most often, this command is issued without arguments. The appropriate password is for the person currently logged into the local UNIX file system. The ticket's lifetime is calculated as described in the DESCRIPTION section above (if no defaults have been changed, it is 25 hours for a user whose Authentication Database entry was created by the AFS 3.1 or later version of the Authentication Server, 100 hours for a user whose Authentication Database entry was created with the AFS 3.0 version).

```
% klog
Password:
```

The following allows the issuer working on a machine in the Transarc cell to authenticate as *admin* in the Transarc test cell, even though he or she is logged into the Transarc machine under a different name.

```
% klog admin -c test.transarc.com
Password: <admin's password>
```

In the following, the issuer requests a ticket lifetime of 104 hours 30 minutes (4 days 8 hours 30 minutes). Presuming that this lifetime is allowed by the maximum ticket lifetimes and other factors described in the DESCRIPTION section, the token will have an actual lifetime of 110:44:28, which is the next largest possible value above 104:30:00 (see page A-46).

```
% klog -life 104:30
Password:
```

PRIVILEGE REQUIRED

None. An entry for the issuer must exist in the Authentication Database, and the issuer must supply the correct password.

MORE INFORMATION

login

pagsh

tokens

knfs — enable authenticated access to AFS from an NFS client using NFS/AFS Translator.

knfs -host <*host name*> **[-id** <*user ID (decimal)*>**]**
[-sysname <*host's '@sys' value*>**] [-unlog] [-tokens]**
[-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

knfs -ho <*host name*> **[-i** <*user ID (decimal)*>**]**
[-s <*host's '@sys' value*>**] [-u] [-t] [-he]**

DESCRIPTION

Allows a user to authenticate with AFS and work in the AFS file space while working on an NFS client machine – even when using a system type for which AFS binaries are not available. If AFS defines a system type for the corresponding NFS client, the authenticated user(s) on that machine will be able to access the AFS file tree and issue AFS commands (by using the **-sysname** argument). It AFS does not define a system type for the NFS client, the authenticated user(s) on that machine will be able to access the AFS file tree, but will not be able to issue AFS commands.

To properly execute this command, establish a connection to the machine that will act as the NFS/AFS translator machine for the NFS client machine (*host name*). Next, issue the **klog** command to obtain AFS tokens on the translator machine for every cell that will be contacted from the NFS client machine. Finally, issue the **knfs** command specifying the NFS client machine with the **-host** argument, and optionally the specific user who will use the NFS client to access AFS with the **-id** argument.

Issue this command only over a connection (perhaps established at the console or via **telnet**) to the AFS client machine serving as the NFS/AFS translator machine for *host name*, never on the NFS client machine itself. The Cache Manager on the translator machine then uses the credential structure identified by the new PAG (process authentication group) when accessing AFS on behalf of the issuer working on the NFS client machine. (If the issuer dos not use the **-id** argument, he or she may actually be sharing the credential structure with other users on the NFS client machine; see the WARNINGS section for details.)

When issued on an NFS/AFS Translator machine, this command creates a PAG associated with the indicated NFS client machine (*host name*), and optionally, the user ID (*user ID*) that the issuer is using on *host name*. It associates the issuer's existing AFS token(s) with a new credential structure identified by the new PAG.

The **-unlog** flag discards the tokens in the credential structure identified by the PAG, but does not destroy the credential structure itself. The Cache Manager on the translator machine retains the credential structure until the next reboot, and uses it each time the issuer accesses AFS through the translator machine. The credential structure only has tokens in it if the issuer re-issues **knfs** on the translator machine each time he or she logs into the NFS client machine.

Users working on NFS client machines of system types for which AFS binaries are available (and for which the cell has purchased a license) should use **klog** rather than the **knfs** command.

WARNINGS

Although the **-id** argument is optional, not using it can create potential security problems and confusion about which users are using which tokens. The security problems arise because if the issuer does not use **-id**, then his or her tokens are associated with a "generic" credential structure identified by a PAG that is associated only with *host name*, not with the specific issuer. Every user working on *host name* who also does not use the **-id** argument shares the same generic credential structure and so uses the issuer's tokens. This includes users on *host name* who do not issue **knfs** at all. If another user subsequently issues **knfs** without **-id**, his or her tokens replace the previous issuer's tokens in the credential structure, and everyone shares the new tokens instead. Similarly, when an issuer discards tokens with the **-unlog** flag and does not use **-id**, then everyone using the generic credential structure becomes unauthenticated at the same time.

The sharing of tokens that result from not using **-id** is acceptable from a security standpoint only if

- The *host name* is used by only one person.
- It is acceptable that all of *host name*'s users who do not have their own credential structure have the same access to AFS files. They should be aware that they are subject to token sharing.

The DESCRIPTION section mentioned that using the **-unlog** flag does not destroy the credential structure, but only discards the tokens associated with it. The Cache Manager on the translator machine retains the credential structure until the next reboot and uses it whenever the issuer accesses AFS through the translator machine. One implication is that once the issuer issues **knfs** using the **-id** argument, he or she cannot use the generic credential structure until the machine is rebooted.

This command does not make it possible for users working on non-supported machine types to issue AFS commands. This is possible only on NFS clients of a system type for which AFS is available (and for which the cell has purchased AFS).

ARGUMENTS

-host names the NFS client machine the issuer is working on. A full name is safest, but abbreviated forms are acceptable depending on the state of the cell's name server at the time the command is issued.

-id specifies the issuer's user ID on the NFS client (a UNIX UID or equivalent), which NFS passes to the translator machine to identify the user.

-sysname specifies the value of the *@sys* variable if the NFS client machine is running a defined AFS operating system (that is, one for which AFS binaries are available). This allows users on NFS client machines using defined system types to issue AFS commands.

-unlog discards the tokens in the credential structure identified by the PAG associated with **-host** and, optionally, **-id**.

-tokens allows users who do not have access to AFS binaries to obtain information about their AFS tokens.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

EXAMPLES

The following shows a typical use of this command. The issuer *smith* is working on *nfs-client1.transarc.com* and has user ID 1020 on that machine. He is accessing AFS through the translator machine called *translator4.transarc.com*. He telnets to *translator4* and uses AFS **login** as *smith* in the Transarc cell. He uses **klog** to obtain tokens in the Transarc test cell (*test.transarc.com*) as *admin*. The **knfs** command instructs the Cache Manager on *translator4* to associate both tokens with the PAG identified by *nfs-client1* and user ID *1020*. He exits the telnet connection and works normally on *nfs-client1*.

```
% telnet translator4.transarc.com
. . .
login: smith
Password:
AFS 3.2 (R) login
% klog admin -c test.transarc.com
Password:
% knfs nfs-client1.transarc.com 1020
% exit
```

The following shows *smith* telnetting again to *translator4* and discarding his tokens.

```
% telnet translator4.transarc.com
. . .
login: smith
Password:
AFS 3.2 (R) login
% knfs nfs-client1.transarc.com 1020 -unlog
% exit
```

PRIVILEGE REQUIRED

The issuer must provide correct password(s) when obtaining tokens on the NFS/AFS translator machine.

MORE INFORMATION

klog

login (AFS version)

pagsh

kpasswd — change password in Authentication Database.

kpasswd [-x] [-principal <*user name*>**]**
[-password <*user's password*>**]**
[-newpassword <*user's new password*>**]**
[-cell <*cell name*>**]**
[-servers <*explicit list of servers*>+**]**
[-pipe] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

kpasswd [-x] [-pr <*user name*>**] [-pa** <*user's password*>**]**
[-n <*user's new password*>**] [-c** <*cell name*>**]**
[-s <*explicit list of servers*>+**]**
[-pi] [-h]

DESCRIPTION

Changes password for indicated user in Authentication Database. By default, the change occurs in the local cell's Authentication Database for the user logged into the local machine's UNIX file system.

The issuer (or user indicated with **-principal**) does not have to appear in the local password file (*/etc/passwd* or equivalent) to issue this command; in previous versions of this command, users had to add the **-x** flag if they did not appear in that file.

ARGUMENTS

-x appears only for backwards compatibility. Its former function is now the default behavior of this command, as mentioned in the DESCRIPTION section.

-principal
names the Authentication Database entry in which to change the password. If this argument is omitted, the change is made to the command issuer's password.

-password
specifies the current password. It is not recommended that the issuer provide this argument on the command line. If it is omitted, the new password is prompted for and does not echo visibly:

`Old password:` <*user's password*>

-newpassword
specifies the new password, which the **kpasswd** command interpreter converts into an encryption key (string of octal numbers) before sending it to the Authentication Server for storage in the user's Authentication Database entry. Unlike the UNIX **passwd** command, **kpasswd** does not restrict passwords to 8 characters; it accepts passwords of virtually any length. All

AFS commands that require passwords (**klog**, the **kas** suite, **kpasswd**, and the AFS version of **login**) can handle passwords longer than 8 characters, but some non-AFS programs cannot. This is a consideration if non-AFS programs handle AFS passwords in the local environment.

It is not recommended that the issuer provide this argument on the command line. If it is omitted, it is prompted for and does not echo visibly:

```
New password (RETURN to abort): <new password>
Retype new password: <new password>
```

-cell specifies the cell in which to change the password, by directing the command to that cell's Authentication Servers. The issuer may abbreviate *cell name* to the shortest form that distinguishes it from the other cells listed in */usr/vice/etc/CellServDB* on the client machine on which the command is issued.

By default, the command is executed in the local cell, as defined

- First, by the value of the environment variable AFSCELL.
- Second, in */usr/vice/etc/ThisCell* on the client machine on which the command is issued.

-servers causes the command interpreter to establish a connection with the Authentication Server running on each specified database server machine. It then chooses one of these at random to execute the command. The issuer may abbreviate the machine name to the extent the cell's name server will accept.

By default, the command interpreter establishes a connection with each machine listed for the indicated cell in the local workstation's copy of */usr/vice/etc/CellServDB*, and then chooses one of those at random for command execution.

This option is useful for testing specific servers if problems are encountered.

-pipe indicates that the command should run without printing anything on the screen, including prompts or error messages. The **kpasswd** command interpreter expects to receive all necessary arguments, each on a separate line, from standard input (stdin). The issuer is discouraged from including this argument; it is for use by application programs rather than human users.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

EXAMPLES

The following shows the typical use of this command, when the issuer wishes to change his or her own password.

```
% kpasswd
Changing password for 'user' in cell 'cellname'.
Old password:
New password (RETURN to abort):
Verifying, please re-enter new_password:
```

PRIVILEGE REQUIRED

Issuer must provide correct current password.

MORE INFORMATION

klog

pagsh — create new process authentication group (PAG).

pagsh

DESCRIPTION

Creates a new command shell (owned by the issuer of the command) and associates a new PAG (process authentication group) with the shell and the user. A PAG is a number guaranteed to identify the issuer of commands in the new shell uniquely to the local Cache Manager. The PAG is used, instead of the issuer's UNIX UID, to identify him or her in the credential structure that the Cache Manager creates to track each user.

Any additional tokens issued to the user (presumably for other cells) become associated with the PAG, rather than with the user's UNIX UID. This method for distinguishing users has two advantages.

- It means 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*.
- It closes a potential security loophole: UNIX allows anyone already logged in as "root" on a machine to assume any other identity by issuing **su**. If the credential structure were identified by a UNIX UID rather than a PAG, then a root user could assume a UNIX UID and use any tokens associated with that UID. Use of a PAG as an identifier eliminates that possibility.

Note: If you are using the Kerberos version of this command (that is, *pagsh.krb*), the PAG credential structure created will support the use of tokens stored in the Cache Manager kernel and the ticket-granting tickets stored in */tmp* (or in the directory specified by the environment variable KRBTKFILE). This command supports the placement of ticket-granting tickets by the Kerberos version of the **klog** (that is, *klog.krb*) and **login** (that is, *login.krb*) commands.

WARNINGS AND REQUIREMENTS

Each PAG created uses two of the memory slots that the kernel uses to record the UNIX groups associated with a user. If none of these slots are available, the **pagsh** command fails. This is not a problem with most operating systems, which make at least 16 slots available per user.

Users in cells not using the AFS version of **login** should issue this command before issuing **klog**. If they do not, then the Cache Manager stores the token in a credential structure identified by UNIX UID rather than PAG. This creates the potential security loophole described in the DESCRIPTION section.

If the command is to be issued on NFS client machines for which AFS is available, the **pagsh** binary accessed by the NFS client must be owned by, and grant set user ID privilege to, "root." The complete set of mode bits should be `-rwsr-xr-x`. This is not a requirement when the command is issued on AFS client machines.

EXAMPLES

The following shows the only possible use of the command:

% **pagsh**

PRIVILEGE REQUIRED

None

MORE INFORMATION

klog

login (AFS version)

tokens

pts adduser — add users and machines to groups.

pts adduser -user <*user name*>$^{+}$ **-group** <*group name*>$^{+}$
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts ad -u <*user name*>$^{+}$ **-g** <*group name*>$^{+}$ **[-c** <*cell name*>**] [-n] [-t] [-f] [-h]**

DESCRIPTION

Makes each specified user and machine entry a member of each specified *group name* entry. While there is theoretically no limit to the number of members in a group, the **pts membership** command can list no more than 5000 members, even if more exist.

WARNING

The effects of adding a user to a group are not immediate. Before a new user member of a group gets the access rights of that group, he or she must re-authenticate (issue **klog**). As long as a user retains old tokens, he or she retains the access rights acquired with those tokens.

ARGUMENTS

-user specifies the user name of each user entry and/or the IP address of each machine entry to be added to each group.

-group specifies the complete group name (including the owner field if applicable) of each group to which each indicated user and/or machine entry should be added.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

EXAMPLES

The following adds user *smith* to *system:administrators*.

% **pts ad smith system:administrators**

The following adds user *pat* to *terry:friends* and to *staff*:

% **pts ad pat terry:friends staff**

The following adds users *jones*, *terry* and *pat* to *smith:colleagues*. The switches are necessary because there is more than one instance of the first list-taking argument.

% **pts ad -u jones terry pat -g smith:colleagues**

The following adds the machine entries in the Transarc Corporation subnet to the group *bin-prot*. Because of the IP address range of the Transarc Corporation subnet, the system administrator was able to group the machines into three machine entries (using the wildcard scheme discussed under the **pts createuser** command).

% **pts ad 158.98.0.0 192.55.207.0 192.54.226.0 -gr bin-prot**

PRIVILEGE REQUIRED

Depends on how the group's fourth privacy flag is set:

- If it is a hyphen, only the group's owner and members of *system:administrators* may add members.
- If it is a lowercase a, members of the group may add other members.
- If it is an uppercase A, everyone on the system may add members.

The **pts setfields** command sets the privacy flags and the **pts examine** command displays them.

MORE INFORMATION

pts examine

pts membership

pts removeuser

pts setfields

pts apropos — show each help entry containing keyword.

pts apropos -topic <*help string*> **[-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts ap -t <*help string*> **[-h]**

DESCRIPTION

Displays the first line of the help entry for any **pts** command that has *help string* in its name or short description.

ARGUMENTS

-topic specifies the keyword string for which to search. If it is more than a single word, surround it with double quotes or other delimiters. Type all *help string*s for **pts** commands in lowercase letters.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

OUTPUT

The first line of a command's online help entry names the command and briefly describes what it does. The **pts apropos** command displays that first line for any **pts** command where *help string* is part of the command name or first line.

To see the remaining lines in a help entry, which provide the command's alias (if any) and syntax, use the **pts help** command.

EXAMPLE

The following lists all **pts** commands that have the word "*create*" in their operation code or short online description.

```
% pts ap create
creategroup: create a new group
createuser: create a new user
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

pts help

pts chown — change owner of entry in Protection Database.

pts chown -name <*group name*> **-owner** <*new owner*>
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts ch -na <*group name*> **-o** <*new owner*> **[-c** <*cell name*>**] [-no] [-t] [-f] [-h]**

DESCRIPTION

Changes the owner of *group name* to *new owner*, by putting the latter in the owner field of *group name*'s Database entry. The new owner may be either an individual or another group.

In the case of regular groups, this command automatically changes the owner-name field (before the colon) of the group name to match *new owner*. If *new owner* is itself a group, then only its owner-name field, not its whole name, is put before the colon in the new name.

The change in the owner-name field of a regular group's name does not propagate down to any groups owned by the group. To change the owner names of these groups, use **pts rename**.

Note: Although a machine can theoretically own a group entry, this ownership confers no functionality and is not recommended. All discussion of this command will refer only to user and group ownership.

ARGUMENTS

-name specifies the current name of the group whose owner is to change.

-owner specifies the name of the user or group which is to become the new owner of the entry.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

EXAMPLES

The following shows user *terry* transferring the ownership of her group *terry:friends* to *pat*. A side effect is that the group name changes to *pat:friends*.

% pts chown terry:friends pat

The following shows user *terry* transferring the ownership of her group *terry:friends* to the group *pat:buddies*. A side effect is that the group name changes to *pat:friends*.

pts chown terry:friends pat:buddies

PRIVILEGE REQUIRED

Issuer must be *group name*'s current owner or belong to the *system:administrators* group.

MORE INFORMATION

pts rename

pts creategroup — create (empty) group entries in Protection Database.

pts creategroup -name <*group name*>$^{+}$
[-owner <*owner of the group*>**]**
[-id <*id (negated) for the group*>$^{+}$**] [-cell** <*cell name*>**]**
[-noauth] [-test] [-force] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

pts cg -na <*group name*>$^{+}$ **[-o** <*owner of the group*>**]**
[-i <*id (negated) for the group*>$^{+}$**] [-c** <*cell name*>**] [-no] [-t]**
[-f] [-h]

DESCRIPTION

Creates an entry in the Protection Database for each specified *group name*. The issuer of the command is recorded as the group's creator. The issuer is also recorded as the group's owner unless the **-owner** argument is provided to name a different individual or a group as the owner. By default, the Protection Server allocates the next available AFS UID (as determined by the `GroupMax` counter), but members of *system:administrators* can assign a different ID.

Two types of groups can be created: ***regular*** and ***prefix-less***; though only a member of *system:administrators* may create the latter type. The rules for naming the two types of groups are discussed in the ARGUMENTS section.

Creating a group lowers the issuer's group-creation quota by one. See the **pts setfields** or **pts examine** command for more information.

ARGUMENTS

-name specifies the name of each group. A group name may be up to 63 characters in length and contain lowercase letters, numbers, and punctuation (except the colon); uppercase letters are not allowed.

When creating regular groups, *group name* must have the following format:

owner-name:group-name

The *owner-name* field must match the true owner of the group, as follows:

- If the issuer does not include the optional **-owner** argument, *owner-name* must match his or her user name.
- If the issuer does use **-owner** to name an individual, then *owner-name* must match the owner of the group's user name.
- If the issuer uses **-owner** to name a regular group as the owner, then *owner-name* must match the *owner-name* field of the owner

group. If the issuer uses **-owner** to name a prefix-less group as the owner, then *owner-name* must match the entire owner group name.

When creating prefix-less groups, the *owner-name* field is omitted, leaving only the *group-name* field; the colon is unnecessary. Only members of *system:administrators* can create prefix-less groups.

-owner specifies a user different from the issuer of the command or a group (regular or prefix-less) that is to own each specified group. If the specified owner is to be a group, that group must already have at least one member; this requirement prevents the creator from assigning self-ownership to a group during its creation.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

EXAMPLES

The following shows *pat* creating the groups *pat:friends* and *pat:colleagues*. Using the **-name** switch is required because more than one instance is provided but this argument is not the last list-taking one.

% **pts cg -name pat:friends pat:colleagues**

The following shows *smith*, who belongs to *system:administrators*, creating the prefix-less group *staff* and assigning its ownership to *system:administrators* rather than to himself.

% **pts cg staff system:administrators**

The following shows *pat* creating the group *smith:team-members*, which is allowed because the owner is set to *smith* at the same time. Note that creation of the group will be charged against *pat*'s group-creation quota, not *smith*'s.

% **pts cg smith:team-members smith**

PRIVILEGE REQUIRED

To assign AFS UIDs with the **-id** argument or create prefix-less groups, the issuer must belong to the *system:administrators* group. Any user may create regular groups, but he or she must

- Be authenticated. The command does not work if the **-noauth** flag is provided.
- Have a group creation quota greater than `0`. The **pts examine** command displays this quota.

MORE INFORMATION

pts examine

pts setfields

pts delete — delete entries from Protection Database.

pts delete -nameorid <*user or group name or id*>$^+$
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts del -na <*user or group name or id*>$^+$ **[-c** <*cell name*>**]**
[-no] [-t] [-f] [-h]

DESCRIPTION

Removes each specified user, machine, and/or group entry from the Protection Database. When a user or group entry is deleted, its AFS UID appears on ACLs instead of the user name or group name. (Use the **fs cleanacl** command to remove obsolete AFS UIDs from ACLs.)

Deleting a user or machine entry removes the user name or IP address from the group membership lists. Deleting a group entry causes:

- The group name to be removed from the membership list of every user and machine Protection Database entry on which it appeared.
- The group-creation quota of the group's creator to be incremented by one, even if the creator no longer owns the group. See the description of the **pts setfields** or **pts examine** command for more information about group creation quotas.

ARGUMENTS

-name specifies the name or AFS UID of each user or group, or the IP address or AFS UID of each machine entry to be deleted. Users, machines, and groups can be specified on the same command line, as can names (IP addresses for machines) and AFS UIDs. Precede the AFS UID of each group with a hyphen to indicate that it is negative.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

EXAMPLES

The following shows users *pat* and *terry* being removed from the Protection Database:

% **pts del pat terry**

The following shows the issuer removing the group with AFS UID -215 from the Protection Database. Using the **-i** switch is required because no instance is provided for the **-name** argument.

% **pts del -i -215**

PRIVILEGE REQUIRED

To delete user and machine entries, the issuer must belong to the *system:administrators* group. To delete group entries, the issuer must either be the group's owner or belong to the *system:administrators* group.

MORE INFORMATION

fs cleanacl

pts adduser

pts creategroup

pts examine

pts removeuser

pts setfields

pts examine — show entries from the Protection Database.

pts examine -nameorid <*user or group name or id*>$^{+}$
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts e -na <*user or group name or id*>$^{+}$ **[-c** <*cell name*>**]**
[-no] [-t] [-f] [-h]

DESCRIPTION

Displays information about the specified user, machine, and/or group entries, as described in the OUTPUT section below.

ARGUMENTS

-name specifies the name or AFS UID of each user or group, or the IP address or AFS UID of each machine whose Protection Database entry is to be displayed. Users, machines, and groups can be specified on the same command line, as can names (IP addresses for machines) and AFS UIDs. Precede the AFS UID of each group with a hyphen to indicate that it is negative.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

OUTPUT

The output of the command contains the following information about each specified entry:

- ***Name***. For users, this is the user name. For machines, it is the IP address. For groups, it is the complete group name with owner field included if applicable. This field may be changed with the **pts rename** command.

- ***AFS UID***. This number is the AFS UID of the entry. It is positive for users and machines, negative for groups.
- ***Owner***. This is the user/group entitled to administer the entry. For user and machine entries, this field is automatically set to *system:administrators* at creation time and should not be changed. For group entries, this field may be changed with the **pts chown** command.
- ***Creator***. This is the user who initially created the entry; its main use is as an audit trail.
- ***Memberships***. For user and machine entries, this number indicates the number of groups to which the user or machine belongs; for group entries, it indicates the number of users and/or machines belonging to the group. The **pts membership** command reveals the identities of the groups, users, and machines.
- ***Privacy Flags***. This field defines the access and manipulation privileges for the entry. The value may be changed with the **pts setfields** command.

  There are five slots, one for each of the five privacy flags. The flags must appear in the order listed here, and something must appear in all five slots. The meaning of the privacy flags and the single letter corresponding to each follows:

STATUS **(s)**:	list status information about an entry (issue the **pts examine** command).
OWNED **(o)**:	list the groups owned by a user or group (issue the **pts listowned** command).
MEMBERSHIP **(m)**:	list the groups to which a user or machine belongs or the users and machines who belong to a group (issue the **pts membership** command).
ADD **(a)**:	add a user or machine to a group (issue the **pts adduser** command). This makes sense only with group entries.
REMOVE **(r)**:	remove a user or machine from a group (issue the **pts removeuser** command). This makes sense only with group entries.

  One of three entries may appear in each flag slot; these entries define the users that possess the right associated with that slot as follows:

  – A ***hyphen*** in the slot indicates that only the owner and members of *system:administrators* have the right. For user entries, the user in question also has the right.

  – A ***lowercase letter*** in the slot indicates that members of the group have the right. This applies to group entries only.

  – An ***uppercase letter*** in the slot indicates that everyone has the right. This is equivalent to *system:anyuser* on standard directory ACLs.

  The default privacy flags are "`S----`" for user and machine entries and "`S-M--`" for group entries.

User and machine entries do not have members, so the last two slots, ADD and REMOVE, are meaningless for them. For the same reason, there is no sensible interpretation of any of the lowercase letters for user entries; they are equivalent to the hyphen. The lowercase letters are equivalent to the hyphen for self-owned groups, too, since all the members of a self-owned group are owners of the group.

- ***Group quota***. This number indicates how many additional groups the user can create. When a user or machine entry is created, the default value is `20`; however, a machine cannot create groups because it is not possible to authenticate as a machine. The **pts setfields** command allows system administrators to reset this number. This field is meaningless for groups and should always be `0` for them.

EXAMPLES

The following shows the output for *terry*'s entry:

```
% pts ex terry
Name: terry, id: 1045, owner: system:administrators, creator: admin,
  membership: 9, flags: S----, group quota: 15.
```

The following shows the output for the machine entry *158.98.4.10*:

```
% pts ex 158.98.4.10
Name: 158.98.4.10, id: 5151, owner: system:administrators, creator: kt,
  membership: 1, flags: S----, group quota: 20.
```

In the following, the issuer is interested in the groups with AFS UIDs -673 and -674:

```
% pts ex -673 -675
Name: terry:friends, id: -673, owner: terry, creator: terry,
  membership: 5, flags: S-M--, group quota: 0.
Name: smith:colleagues, id: -675, owner: smith, creator: smith,
  membership: 14, flags: SOM--, group quota: 0.
```

PRIVILEGE REQUIRED

Depends on how the entry's first privacy flag is set

- If it is lowercase `s` and this is a group entry, only members of the group and of *system:administrators* may examine the entry. If this is a user entry, only members of *system:administrators* and the user may examine the entry. If this is a machine entry, only members of *system:administrators* may examine the entry.
- If it is uppercase `S`, everyone on the system may examine the entry.

It is not possible to set this flag to the hyphen (the command interpreter would not be able to distinguish the hyphen from one that indicates a flag or switch).

The **pts setfields** command sets the privacy flags.

MORE INFORMATION

pts adduser

pts rename

pts chown

pts listowned

pts membership

pts removeuser

pts setfields

pts help — show syntax of specified **pts** command(s) or list functional description for all of them.

pts help [-topic <*help string*>$^+$] [-help]

ACCEPTABLE ABBREVIATIONS/ALIASES

pts h [-t <*help string*>$^+$**] [-h]**

DESCRIPTION

Displays the first line (name and short description) of every **pts** command's help entry, if no *help string* is provided. For each operation code specified with **-topic**, it outputs the entire help entry. See the OUTPUT section below.

ARGUMENTS

-topic specifies the operation codes for which syntax is to be provided. If it is omitted, the first line (name and short description) of every **pts** command is displayed.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

OUTPUT

The online help entry for each **pts** command consists of two or three lines:

- The first line names the command and briefly describes what it does.
- The second line shows any aliases the command has (it does not appear for every command).
- The final line, which begins with "`Usage:`", lists the command's arguments and flags in the prescribed order. Online help entries use the same symbols (for example, brackets) as the command definitions in this manual. For an explanation of their meaning, see chapter 2.

EXAMPLE

The following displays the online help entry for the **pts membership** command:

```
% pts help membership
pts membership:  list membership of a user or group
aliases: groups
Usage: pts membership -nameorid <user or group name or id>+
[-cell <cell name>] [-noauth] [-test] [-force] [-help]
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

pts apropos

pts listowned — show groups owned by specified users and/or groups.

pts listowned -nameorid <*user or group name or id*>$^{+}$
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts listo -na <*user or group name or id*>$^{+}$ **[-c** <*cell name*>**]**
[-no] [-t] [-f] [-h]

DESCRIPTION

Lists the names of the groups owned by the specified users and/or groups. If a value of *0* is provided for **-nameorid**, the command also lists all "orphaned groups" (groups that still exist in the database even though their owners have been deleted from it). Only system administrators may specify *0* with **-nameorid**.

Note: Although a machine can theoretically own a Protection Database entry, this ownership confers no functionality and is not recommended. All discussion of this command will refer only to user and group ownership.

ARGUMENTS

-nameorid
specifies the name or AFS UID of each user or group for which group ownership information is to be displayed. Users and groups can be specified on the same command line, as can names and AFS UIDs. Precede the AFS UID of each group with a hyphen to indicate that it is negative.

Providing a value of *0* (zero) lists entries for groups that still exist in the database even though the entries for their owners have been deleted. Only system administrators may specify 0 with **-nameorid**.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

OUTPUT

The command displays group ownership information for each specified user or group. The first line of the output displays the *name* and *AFS UID* of the user or group for which ownership information is to be displayed, as follows:

`Groups owned by` *name* `(id:` *AFS UID*`) are:`

A list of groups owned by the user or group follows. The list does not include groups owned by groups that the user or group owns or to which the user or group belongs. If the user or group does not own any groups, the preceding line appears with nothing after it.

If the second protection flag is the hyphen (as it is by default), most users will not be allowed to list the owned groups. A `Permission denied` message is displayed in this case.

EXAMPLES

The following shows the groups owned by *terry* and *terry:friends*:

```
% pts listo terry terry:friends
Groups owned by terry (id: 1045) are:
  terry:friends
  terry:project1
  terry:project2
Groups owned by terry:friends (id: -673) are:
```

The following shows the message that *smith* receives when he tries to list the groups owned by *pat*. The message is displayed because the second protection flag on *pat*'s entry is the hyphen.

```
% pts listo pat
pts: Permission denied so failed to get owner list for pat (id: 1144)
```

PRIVILEGE REQUIRED

Depends on how the entry's second privacy flag is set

- If it is the hyphen and this is a group entry, only the owner of the group and members of *system:administrators* may list the groups owned by this group. If this is a user entry, only the user himself/herself and members of *system:administrators* may list the groups owned by the user. The hyphen is the default.
- If it is uppercase `O`, everyone on the system may list the groups owned by this user/group.

It is not possible to set the second flag to lowercase `o` (which would be equivalent in meaning to the hyphen). The **pts setfields** command sets the privacy flags; the **pts examine** command displays them.

MORE INFORMATION

pts examine

pts setfields

pts membership — show membership list for user or group.

pts membership -nameorid <*user or group name or id*>$^+$
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts m -na <*user or group name or id*>$^+$ **[-c** <*cell name*>**]**
[-no] [-t] [-f] [-h]

pts groups -na <*user or group name or id*>$^+$ **[-c** <*cell name*>**]**
[-no] [-t] [-f] [-h]

DESCRIPTION

For each specified user name and/or machine IP address, this command lists the groups to which the user or machine belongs. For each specified group name, it lists the users and/or machines who belong to the group. The output is limited to 5000 names for any user, machine, or group, even if there are actually more in the list.

It is not possible to list the members of *system:anyuser* or *system:authuser*, nor do they appear in the list of groups to which a user belongs. The membership of these groups is not stable, as it depends on, respectively, each user's current login or authentication status.

The **pts adduser** command adds users and machines to groups, and the **pts removeuser** command removes them from groups

ARGUMENTS

-nameorid
specifies the name or AFS UID of each user or group, or the IP address or AFS UID of each machine about which group membership information is to be displayed. Users, machines, and groups can be specified on the same command line, as can names (IP addresses for machines) and AFS UIDs. Precede the AFS UID of each group with a hyphen to indicate that it is negative.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

OUTPUT

For each specified user and machine, the output contains the following header line followed by a list of the groups to which the user belongs:

`Groups` *user name* `(id:` *AFS UID*`) is a member of:`

For each specified group, the output contains the following header line followed by a list of the users and/or machines who belong to the group:

`Members of` *group name* `(id:` *AFS UID*`) are:`

The command will list up to 5000 members of a group, or 5000 groups to which an individual belongs.

EXAMPLE

The following shows the groups to which *pat* belongs and the members of *smith:friends*. Note that *pat*'s third protection flag must have changed from the default hyphen for a non-administrative user to elicit this type of listing.

```
% pts mem -name pat smith:friends
Groups pat (id: 1144) is a member of:
  smith:friends
  staff
  johnson:project-team
Members of smith:friends (id: -562) are:
  pat
  terry
  jones
  richard
  thompson
```

The following shows the groups to which machine *158.98.4.10* belongs (*bin-prot* is a binary protection group composed solely of machines for licensing purposes):

```
% pts mem 158.98.4.10
Groups 158.98.4.10 (id: 5151) is a member of:
  bin-prot
```

PRIVILEGE REQUIRED

Depends on how the entry's third privacy flag is set

- If it is the hyphen and this is a group or machine entry, only the owner of the group and members of *system:administrators* may list the members of the group. If it is a hyphen and this is a user entry, only the user himself/herself and members of *system:administrators* may list the groups to which the user belongs.
- If it is a lowercase `m` and this is a group entry, privilege extends to members of the group. If it is a lowercase `m` and this is a user or machine entry, the meaning is equivalent to the hyphen.
- If it is an uppercase `M`, privilege extends to everyone on the system.

The **pts setfields** command sets the privacy flags, and the **pts examine** command displays them.

MORE INFORMATION

pts examine

pts setfields

pts removeuser — remove users from groups in Protection Database.

pts removeuser -user <*user name*>$^{+}$ **-group** <*group name*>$^{+}$
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts rem -u <*user name*>$^{+}$ **-g** <*group name*>$^{+}$ **[-c** <*cell name*>**]**
[-n] [-t] [-f] [-h]

DESCRIPTION

Removes each user and/or machine entry from membership in the specified *group name* entry. It also removes each *group name* from the list of memberships stored with each user and/or machine entry.

The **pts adduser** command adds users and machines to groups. The **pts membership** command lists the members of a group or the groups to which a user or machine belongs.

WARNING

The effects of removing a user from a group are not immediate. Before a user member of a group is denied the access rights of that group, he or she must re-authenticate (issue **klog**). As long as a user retains old tokens, he or she retains the access rights acquired with those tokens.

ARGUMENTS

-name specifies the user name of each user entry or the IP address of each machine entry to be removed.

-group specifies the name of each group from which each user/machine entry should be removed.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

EXAMPLE

The following removes *smith* from the groups *staff* and *staff:finance*. Note that no switches are necessary because only a single instance is provided for the first argument (the user name).

% **pts rem smith staff staff:finance**

The following removes the three machine entries in the Transarc Corporation subnet from the group *bin-prot*:

% **pts rem 158.98.0.0 192.55.207.0 192.54.226.0 -gr bin-prot**

PRIVILEGE REQUIRED

Depends on how the group's fifth privacy flag is set:

- If it is the hyphen, only the group's owner and members of *system:administrators* may remove members.
- If it is lowercase `r`, privilege extends to members of the group.

It is not possible to set the fifth flag to uppercase `R`.

The **pts setfields** command sets the privacy flags and the **pts examine** command displays them.

MORE INFORMATION

pts adduser

pts membership

pts setfields

pts rename — change name of entry in Protection Database.

pts rename -oldname <*old name*> **-newname** <*new name*>
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts ren -o <*old name*> **-ne** <*new name*> **[-c** <*cell name*>**] [-no] [-t] [-f] [-h]**

DESCRIPTION

Changes the name of a user or group entry called *old name* to *new name*. It is not possible to change a user name to be like a regular group name (with a colon in it). Members of *system:administrators* may change regular group names into prefix-less names and vice versa. When changing a prefix-less group name into a regular group name or a regular group name to another regular group name, the owner-name field of the new name must correctly reflect the group's ownership.

WARNING

By convention, many aspects of an AFS user account have the same name as the user's Protection Database entry, including the Authentication Database entry, volume, and mount point. If this command is used to change a user name, it is recommended for consistency's sake that the names of all relevant related entities also be changed. See the chapter on user accounts in the *AFS System Administrator's Guide* for complete instructions.

ARGUMENTS

-oldname
specifies the current full name of the entry.

-newname
specifies the new full name for the entry. For regular groups, the *owner-name* (pre-colon) part of the new name must reflect the true ownership of the group.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

EXAMPLES

The following changes the name of the group *staff*, owned by the privileged user *admin*, to *admin:staff*:

% **pts ren staff admin:staff**

In the following example, *pat* renames *terry:plans*, a group owned by *pat:staff*; until recently, *pat:staff* was *terry:staff*. This type of change is necessary, when the owner of a group changes, because the groups owned by that group are not automatically renamed to reflect the new owner.

% **pts terry:plans pat:plans**

PRIVILEGE REQUIRED

To change a regular group name to a prefix-less name or vice versa, the issuer must belong to *system:administrators*. For regular-to-regular or prefix-less-to-prefix-less changes, the issuer must be the group's owner or belong to the *system:administrators* group. To change user entries, the issuer must belong to the *system:administrators* group.

MORE INFORMATION

pts chown

pts setfields — set privacy flags or group creation quota for a protection Database entry.

pts setfields -nameorid <*user or group name or id*>$^+$
[-access <*set privacy flags*>**]**
[-groupquota <*set limit on group creation*>**]**
[-cell <*cell name*>**] [-noauth] [-test] [-force] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

pts setf -na <*user or group name or id*>$^+$
[-a <*set privacy flags*>**] [-g** <*set limit on group creation*>**]**
[-c <*cell name*>**] [-no] [-t] [-f] [-h]**

DESCRIPTION

Sets group-creation quota and/or the privacy flags for each entry indicated. Only system administrators may change a group-creation quota. Only the owner of an entry or a system administrator may change privacy flags.

The **pts examine** command shows the current settings of the quota and flags.

Note: Group-creation quota is meaningless for machines and groups, because it is impossible to authenticate as a group or machine.

ARGUMENTS

-nameorid specifies the name or AFS UID of each user or group, or the IP address or AFS UID of each machine for which privacy flags or group creation quota is to be set. Users, machines, and groups can be specified on the same command line, as can names (IP address for machines) and AFS UIDs. Precede the AFS UID of each group with a hyphen to indicate that it is negative.

-access specifies a list of five characters, one for each of the protection rights on a Protection Database entry. The default rights for user and machine entries are `S----` and for group entries `S-M--`. It is necessary to provide this argument only if the default settings are not satisfactory.

There are five slots, one for each of the five privacy flags. The flags always appear in the order listed here, and something always appears in all five slots. The meaning of the privacy flags and the single letter corresponding to each follows:

- STATUS **(s)**: list status information about an entry (issue the **pts examine** command).

- OWNED **(o)**: list the groups a user or group owns (issue the **pts listowned** command).
- MEMBERSHIP **(m)**: list the groups to which a user or machine belongs or the users and machines that belong to a group (issue the **pts membership** command).
- ADD **(a)**: add a user or machine to a group (issue the **pts adduser** command). This is meaningful only for group entries.
- REMOVE **(r)**: remove a user or machine from a group (issue the **pts removeuser** command). This is meaningful only for group entries.

The issuer of the command determines who receives each right by typing one of three types of characters in each slot, as follows:

- To grant the right only to the owner of the entry and members of *system:administrators*, type a ***hyphen*** in the slot. For user entries, the hyphen also gives the user this right.
- To grant the right only to members of the group, type the ***lowercase version*** of the letter associated with the right (as defined just above). Do not type lowercase letters for user or machine entries; they are meaningless for them.
- To grant the right to everyone, type the ***uppercase version*** of the letter associated with the right. This is similar to putting *system:anyuser* on standard directory ACLs.

The following restrictions define how the issuer may set the rights:

- As mentioned above, it is always necessary to specify a value for all five flags.
- Do not set the flags (other than the first flag) in user or machine entries to lowercase letters. Lowercase letters have no sensible interpretation for these entries, because these entries cannot have members. 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`. The hyphen is not legal 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 hyphen is equivalent to the disallowed lowercase `o`, because members of a group own any groups owned by the group.
- Legal values for the third ("membership") flag are the hyphen, lowercase `m`, and uppercase `M`.

- Legal values for the fourth flag ("add members") are the hyphen, lowercase `a`, and uppercase `A`. This flag has no sensible interpretation for user and machine entries, but 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, but must be set nonetheless, preferably to the hyphen.

-groupquota specifies the number of additional groups a user can create (it does not matter how many they have created already). This number is meaningless for groups and machines, since it is not possible to authenticate as a group or machine; there is no reason to set or change this value for those entries.

-cell specifies the cell in which to run the command. See section A.3.4 for more details.

-noauth tells the Protection Server to assign the unprivileged identity *anonymous* to the issuer. See section A.3.4 for more details.

-test tells the command interpreter to contact the machine listed in */usr/afs/etc/CellServDB* for execution of the command. See section A.3.4 for more details.

-force directs the command to continue performing as much of an operation as it can, even if it encounters problems while executing. See section A.3.4 for more details.

-help prints the online help for this command. Do not provide any other arguments or flags with this one. See section A.3.4 for more details.

EXAMPLES

The following changes the privacy flags on the group *operators*. The group retains the default "`S`" and "`M`" flags, but it acquires the "`a`" and "`r`" flags so that its members can add and remove members.

% **pts setf operators -a S-Mar**

The following changes the privacy flags and sets group quota on the user entry *admin*. The default "`S`" protection flag is retained, and the "`O`" and "`M`" flags are enabled.

% **pts setf admin -a SOM-- -g 50**

PRIVILEGE REQUIRED

For group entries, the issuer must be the group’s owner or belong to the *system:administrators* group. For user entries, setting group-creation quota requires that the issuer belong to the *system:administrators* group; setting the privacy flags requires that the issuer be the owner of the group or belong to the *system:administrators* group.

MORE INFORMATION

pts adduser

pts examine

pts listowned

pts membership

pts removeuser

tokens — display all tokens.

tokens

DESCRIPTION

Instructs the Cache Manager on the local machine to display all tokens (tickets) it has for the issuer. AFS server processes require that their clients present a token as evidence that they have authenticated in the server's local cell.

Note: If you are using the Kerberos version of this command (that is, **tokens.krb**), the output will also list information the Kerberos-style tokens stored in the */tmp* directory.

OUTPUT

The output lists one token for each cell in which the user is authenticated. The output indicates the

- User's AFS UID, if it is available for display.
- Server for which the token is valid (normally, "*afs*"). This includes a cell specification.
- Day and time the token expires.

Note: If you are using **tokens.krb**, the output will also contain information on the Kerberos ticket-granting ticket including: the ticket owner, the Kerberos ticket-granting service that issued the ticket (for example, *krbtgt.TRANSARC.COM*), and the expiration date of the ticket.

An `--End of list--` message appears at the end of the output. If the user is not authenticated in any cell, this message is all that appears.

EXAMPLES

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 the 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 the Transarc Corporation cell, the Andrew cell at Carnegie Mellon University, and the 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--
```

The following shows the output when the issuer uses **tokens.krb** after authenticating in the Transarc Corporation cell using **klog.krb**.

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

User’s (AFS ID 1000) tokens for afs@transarc.com [Expires Jan 31 00:09]
User smiths tokens for krbtgt.TRANSARC.COM@transarc.com
[Expires Jan 31 00:09]
   --End of list--
```

PRIVILEGE REQUIRED

None

MORE INFORMATION

klog

unlog

unlog — discard all tokens.

unlog [-cell <*cell name*>$^+$**] [-help]**

ACCEPTABLE ABBREVIATIONS/ALIASES

There is no abbreviation or alias for the **unlog** command; however, you can abbreviate the switch and flag normally.

DESCRIPTION

Instructs the Cache Manager on the local machine to discard the specified token(s) currently held for the issuer. If no cell names are provided, the Cache Manager discards the token for the local cell and all tokens for foreign cells; if cell names are provided, using the **-cell** argument, only the tokens for those cells are discared.

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

WARNINGS

Specifying one or more *cell names* may cause brief "authentication outages," during which the issuer has no valid tokens in any cell. This is because the command actually discards all tokens and then restores those that the issuer did not specify with *cell name* (and so presumably wishes to retain). The authentication outage may interrupt the operation of jobs that require authentication.

ARGUMENTS

-cell specifies each cell for which the Cache Manager should discard the token. If omitted, all tokens are discarded. Abbreviated cell names are acceptable, but which abbreviations are legal depends on the naming service available in the cell at the time the command is issued.

-help prints the online help for this command. Do not provide any other arguments or flags with this one.

EXAMPLES

The following discards all tokens:

% unlog

The following discards only the tokens for the *transarc.com* and *athena.mit.edu* cells:

% unlog transarc.com athena

PRIVILEGE REQUIRED

None

MORE INFORMATION

klog

tokens

Appendix B: NFS/AFS Translator

Replace this page with the tab separator for

Appendix B: NFS/AFS Translator

Appendix B. NFS/AFS Translator

If your site uses the Network File System (NFS) and you are using an NFS client machine, you can still access the AFS file space by using the NFS/AFS Translator™. Furthermore, if your NFS client machine uses an AFS supported operating system, you can even issue many of the AFS commands from that machine over the NFS/AFS Translator.

This appendix includes the following sections:

B.1. Setting up the Components of the NFS/AFS Translator

To use the NFS/AFS Translator (hereafter, ***translator***), your site must have the following components:

- An NFS/AFS translator machine (hereafter, translator machine). A ***translator machine*** is an AFS client that simultaneously runs as an NFS server. Your system administrator can provide you with a list of the translator machines in your cell.
- An NFS client machine configured to use the services of the translator. The system administrator configures the NFS client by mounting AFS and by defining two AFS environment variables on the NFS client (AFSCONF and AFSSERVER). By convention, the NFS mount directory for AFS is named */afs* so that NFS pathnames to the AFS file space match the AFS pathnames.

Although an AFS user account is not required to access the AFS file space, it is recommended. If you do not have an AFS user account, you will only be permitted to access AFS as a member of the group *system:anyuser* and your access to AFS directories and files will be limited.

B.1.1. Establishing the NFS/AFS Connection

If your NFS machine uses an operating system supported by AFS (one for which AFS binaries are available), you can establish the translator connection by authenticating as if you were working on an AFS client. If your NFS machine uses an unsupported operating system (one for which no AFS binaries are available), then the procedure for establishing the translator connection is somewhat different. The instructions that follow discuss the two methods of establishing the connection.

To connect to AFS on a supported operating system

To establish a connection to AFS on an NFS client machine using a supported operating system, you must:

1. Log in to the NFS client machine under your NFS user name.
2. Authenticate with AFS by issuing **klog**; if you do not have an AFS account, this step is unnecessary. (See section 3.1.2 for information on authentication.)
3. (Optional) Issue the **knfs** command to allow use of the *@sys* variable on the NFS client. (See the **knfs** entry in Appendix A for an explanation of the **-sysname** argument.)

To connect to AFS on a non-supported operating system

To establish a connection to AFS on an NFS client machine using an unsupported operating system, you must

1. Log in to the NFS client machine under your NFS user name.
2. Access the translator machine (for example, via **telnet**).
3. Authenticate with AFS by issuing **klog**; if you do not have an AFS account, this step is unnecessary. (See section 3.1.2 for information on authentication.)
4. Issue the **knfs** command so that the Cache Manager on the translator machine associates the tokens you obtained in the previous step with your NFS identity.

   % **knfs** <*host name*> **[**<*user ID (decimal)*>]

   where

   host name
   : specifies the full name of the NFS client machine from which you will access AFS.

   user ID (decimal)
   : specifies the UNIX UID or equivalent (not your username) that is used to identify you as a unique user on the NFS client machine. If you do not know the *user ID (decimal)* or think you may not have one, contact your system administrator. If you do not have a *user ID (decimal*), you can omit it, but you run the risks associated with sharing tokens with other users on your NFS machine. (See the **knfs** entry in Appendix A for a complete explanation.)

5. (Optional) Log out from the translator machine without unauthenticating.
6. Work on the NFS client machine as you normally would.
7. When you are finished accessing AFS, issue the **knfs** command on the translator machine again, adding the **-unlog** flag to destroy your tokens. (If you logged out in step 5, you must first log in to the translator machine again.)

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

   Note: If you are sharing "generic" AFS tokens with other users on the same NFS client (because when you originally issued **knfs**, you did not specify *user ID (decimal)*), make sure that the other users are finished with those tokens before issuing the above command.

B.2. Using AFS Commands through NFS

You can issue AFS commands while working on an NFS client machine if both of the following are true:

- Your NFS client machine uses a supported operating system. Ask your system administrator if you are not sure about the status of your machine's operating system.
- Your system administrator has configured the NFS client machine and your account to enable you to issue AFS commands. Again, you should ask your system administrator about the status of your machine and account.

Note: You cannot issue AFS commands on an NFS client machine if AFS binaries are not available.

B.3. Troubleshooting the NFS/AFS Translator

Keep in mind that NFS/AFS translator performance depends for the most part on the stability of the NFS system. While file outages, broken connections, and inaccessible files may manifest themselves as AFS problems, they may be caused by NFS.

This troubleshooting section will help you diagnose some command problems. If other problems arise, contact your system administrator. Transarc Customer Support will try to supply work-arounds for NFS-related problems.

Note: AFS uses a delayed write mechanism. Changes made and written to AFS files may take up to 30 seconds to be visible to client machines using a different translator machine.

B.3.1. Your NFS Client Machine is Frozen

Your NFS client machine can "freeze" or "hang" when the translator machine goes down – if the translator was mounted with a hard mount. If the translator does not respond, you can interrupt the hung process (using **<Ctrl-c>** or a similar interrupt) and continue to work locally. If the translator machine will be down for an extended period, you can mount another translator.

Note: An interrupt will only work if the mount was set using the *intr* mount option. If the *intr* option was not used, you must unmount the translator from another NFS process.

B.3.2. NFS/AFS Translator Reboots

When your translator machine reboots, the system name for your NFS client machine needs to be redefined. And if you are authenticated to AFS, you must reauthenticate using the **klog** command.

B.3.3. System Error Messages

`stale NFS client` – Your translator machine was rebooted and cannot map the current directory. **cd** to your current directory using the full pathname (that is, */afs/cell/username/directory*) and the path will be re-established.

`Getpwd: can't read` – A translator machine has rebooted and cannot map the current directory. **cd** using the full pathname, and the path will be re-established.

`"NFS server <nfsserver> is not responding still trying"` – If the translator is mounted using a hard mount, you will get this warning message, but your machine will continue to retry the request. If the translator is mounted with a soft mount, the operation will abort after the requested number of transmissions (the default is 3).

Glossary

Replace this page with the tab separator for

Glossary

Glossary

Access Control List (ACL)	A list associated with an AFS directory that specifies what actions a user or group may perform on the directory and the files in it. There are seven access rights: READ, LOOKUP, INSERT, DELETE, WRITE, LOCK, and ADMINISTER, each with a corresponding letter abbreviation, `r`, `l`, `d`, `w`, `k`, `a`.
Access List Entry	An entry on an ACL that pairs one user or group of users with the specific access rights they possess.
Access Right	A certain type of access granted on an ACL. Anyone who possesses the right can perform the action.
Add Privacy Flag	Enables the possessor to add a user to a group. Letter: `a`.
Administer Right	An ACL right that allows the possessor to change the entries on the ACL. This protection flag enables the possessor to add and remove users in a group.
NFS/AFS Translator	Allows users on NFS client machines to access files in the AFS file space.
Alias	A shortened or alternative form of an AFS command.
"All" ACL Shorthand	A shorthand notation used in the **fs setacl** command to represent all seven access rights.
Argument	A portion of a command that names an entity to be affected by the command. Arguments consist of two parts: a ***switch*** and ***instance***(s). Some AFS commands take one or more arguments.
Authenticated	To become recognized as a valid AFS user by providing the correct password. Authentication occurs during AFS login or when the **klog** command is issued. A user is authenticated or recognized as a valid user by providing the correct password. Only authenticated users can perform most AFS actions.
Bytes, kilobyte	A unit of measure used to specify volume, quota, and disk usage. A kilobyte block is equal to 1024 bytes.

Cache Manager

A program on a client machine that accesses files stored in AFS on behalf of the client machine. When a user accesses a file, the Cache Manager requests the file from the appropriate File Server. The Cache Manager stores ("caches") a copy of the requested file on the client workstation's local disk. The local copy of the cached file is used, eliminating sending network requests to File Server machines for data from the stored file.

Cached File

A copy of a file the Cache Manager stores on a workstation's local disk.

Callback

A "promise" from the File Server that it will contact the Cache Manager if the centrally stored copy of the file is altered while the copy of the file is in use on a local machine. If the file is altered, the File Server "breaks" the callback. The next time a program on the workstation asks for data from the file, the Cache Manager notices the broken callback and retrieves an updated copy of the file from the File Server. Callbacks ensure the user is working with the most recent copy of a file.

Cell

An independently administered site running AFS, consisting of a collection of file server machines and client machines defined to "belonging to" the cell. A machine can belong to only one cell at a time.

Client Machines

Workstations which perform computations for users. In most cases, a user will work on a client machine, accessing files stored on a file server machine.

Client/Server Machines

The two types of computers (each with different specialized functions) that compose a computer network: Client Machines and Server Machines.

Command

A string of characters specified to an AFS server, indicating an action to perform. AFS commands can contain a command name, an operation code, arguments and flags, and as well as the switch and instance pairs for the arguments.

Complete Pathname

A full specification of a file's location in AFS. Beginning with */afs*, it specifies all the directories the Cache Manager must check to find the file. The names of the directories are separated by slashes.

Command Suite

The first part of a command string which indicates the group of commands the command belongs to. This is usually based on function.

Delete Access Right

Enables the possessor to remove elements from a directory. Letter: d.

Directory	A logical structure containing a collection of files and other directories.
Distributed File System	A file system that joins the file systems of individual machines. Files are stored (distributed) on different machines in the computer network but are accessible from all machines.
File Server Machines	A type of machine in AFS, used to store files and transfer requested files to client machines.
File	A collection of related information stored and retrieved under a single name.
Flag	Part of a command that determines how the command executes, or the type of output it produces.
Foreign Cell	All cells other than the local cell the client machine is assigned to. Users may authenticate or access files in foreign cells. The Cache Manager (through which the user is accessing the AFS file space) remains the same, and determines which cell is local.
Group	A group is a defined list of individual users who can be placed on the access control lists (ACLs) of directories. When using groups it is possible to grant the same access rights to a number of people at one time.
Group-owned Group	A group owned by another group. All members of the "owner" group can administer the "owned" group; the members of the owned group have no administer rights themselves.
Hierarchical File Structure	A method of organizing and storing data on different levels of directories.
Home Directory	A directory in a file system owned by a user and used by him/her to store personal files.
Insert Access Right	Enables the possessor to add files or directories to a directory. Letter: `i`.
Instance	That part of a command string defining a specific switch.
Local Cell	The cell the user's Cache Manager is assigned to.
Lock Access Right	Enables programs run under the user's name to place advisory locks on a file. Letter: `k`.
Login	The process by which a user accesses AFS or a generic operating system like UNIX.
Logout	The process by which a user signs off from a terminal.
Lookup Access Right	Enables the possessor to list contents of a directory and look at the ACL. Letter: `l`.

Mount Point	A special type of directory that connects a location in the AFS file space with a volume. A mount point looks like a standard UNIX directory; listing the directory shows the contents of the volume. Users can **cd** to it or use **ls** to list its contents. Each mount point corresponds to a single volume.
Members Privacy Flag	Enables the possessor to list the members of a group or groups to which a user belongs. Letter: `m`.
Mutual Authentication	A procedure through which two parties communicating across a network prove their identities to one another. AFS requires mutual authentication whenever a server and client talk to one another.
"None" ACL Shorthand	A shorthand notation used in the **fs setacl** command to delete an entry from an ACL.
Operation Code	The second word in an AFS command, indicating the command's function.
Owner (Group)	The person or group allowed (by default) to administer a group.
Owned Privacy Flag	Enables the possessor to list groups owned by the user or group. Letter: `o`.
Parent Directory	The directory in which a directory or file resides.
Partition	A logical section of a disk in a computer.
Password	A unique, user-defined string of characters validating the user's system identity. The user must correctly enter the password in order to be authenticated.
Quota	The size limit of a volume assigned by the system administrator and measured in kilobyte blocks.
"Read" ACL Shorthand	A shorthand notation used in the **fs setacl** command that grants both the READ and LOOKUP access rights.
Read Access Right	Enables the possessor to examine the contents of a file but not change them. Letter: `r`.
Relative Pathname	A file/directory's location, specified relative to the current working directory. It may include the "." or "..".
Remote Commands	Commands used to run programs on a remote machine without the need for a user to explicitly telnet to it.
Remove Privacy Flag	Enables the possessor to remove members from a group. Letter: `r`.
Self-owned Group	A group that owns itself. All members of the group are allowed to administer the group.

Server	A program that provides specialized service to other servers, such as storing and transferring files or performing authentication.
Status Privacy Flag	Enables the possessor to list status information about users or groups. Letter: `s`.
Subdirectory	A directory that resides in another directory in the file system hierarchy.
Switch	The part of a command string defining the type of an argument. It is preceded by a hyphen.
Syntax	The correct sequence in which a command's arguments must be keyed in on a command line.
System Administrator	A user who is authorized to operate and administer AFS in a cell.
System Groups	Groups created to enable a user to automatically grant access to predefined AFS groups.
System:administrators group	The people in the cell designated as system administrators, who are authorized to operate and administer AFS.
System:anyuser	A system:anyuser group includes everyone who can gain access to the cell, including unauthenticated AFS users, sometimes called anonymous.
System:authuser	A system:authuser group includes everyone who is currently "authenticated" in the cell.
Token	A set of data that indicates a user has authenticated. The AFS server programs accept the token as partial proof that the user is authorized to request a service. If a user does not have a token, the identity of *anonymous* is assigned and the user has *system:anyuser* access. If a user has a token, *authuser* access is assigned.
UNIX Mode Bits	A set of access rights associated with a file/directory in the UNIX file system which appear when **ls -l** is invoked. The rights are: read, write, and execute (r, w, x). AFS combines their effect with AFS access rights in order to determine what type of access someone has to the files.
User name	A short string of characters entered at login that uniquely identifies a person in the local cell.
Volumes	A "container" that keeps a set of related files and directories together on a disk partition that is specific to AFS.
"Write" ACL Shorthand	A shorthand notation used in the **fs setacl** command to grant all rights except ADMINISTER.

Write Access Right Enables the possessor to modify the contents of a file. Letter: `w`.

Index

Replace this page with the tab separator for

Index

Index

Table of Contents

Concepts

Protecting Directories

Troubleshooting

Appendix A: Command Reference

Appendix B: NFS/AFS Translator

Glossary

Index