

AFS TABLE OF CONTENTS

AFS IN THE SCHOOL OF COMPUTER SCIENCE	2
AFS OVERVIEW	2
AFS CELLS.....	3
AFS VOLUMES.....	3
AFS QUOTA	4
<i>How to Use the fs Command to Determine Quota Use</i>	<i>4</i>
<i>Using Jeeves to change AFS Quotas</i>	<i>5</i>
<i>AFS Groups</i>	<i>5</i>
<i>Access Control Lists (ACLs).....</i>	<i>9</i>
<i>AFS Authentication.....</i>	<i>13</i>
<i>Cross-realm Authentication.....</i>	<i>15</i>
<i>Authenticating Daemons to AFS.....</i>	<i>17</i>
AFS COMMAND COMMANDS AND TROUBLESHOOTING	19
<i>Basic Information and Common Commands.....</i>	<i>19</i>
<i>Troubleshooting and Problem Solving</i>	<i>21</i>

AFS in the School of Computer Science

AFS (the Andrew File System) is a distributed, client-server, file system used to provide most file-sharing services in SCS. An AFS client is installed by default on Facilitized Unix, and Windows hosts. Some advantages to using AFS are that it provides:

- Transparent access to files. Files in AFS may be located on different servers, but are accessed the same way as files on your local disk regardless of which server they are on.
- An uniform namespace. A file's pathname is exactly the same from any Unix host that you access it from.
- A secure, fine-grained access control for files. You can control exactly which users have access to your files and the rights that each one has on a directory-by-directory basis.

The School of Computer Science maintains documentation on the use of AFS in the SCS computing environment at <http://www.cs.cmu.edu/~help/afs/>.

OpenAFS is the client and server software deployed within the SCS computing environment. More information on both AFS and the specific OpenAFS implementation can be found at <http://www.openafs.org>.

AFS Overview

Certain terms and conventions used by AFS are:

Cells

AFS, at the top level, is organized into cells, which correspond to separate administrative groups. The cs.cmu.edu cell is administered by SCS Facilities, and there are other AFS cells on campus.

Volumes & quotas

Related directories in AFS are organized into units called volumes. For example, your home directory in AFS will be in its own volume. Each volume has its own quota that determines how much disk space it can use.

AFS Groups and ACLs

AFS groups make it much easier to manage ACLs for large directory trees, and allow the addition of large numbers of users to an ACL with a single command. AFS does not use standard Unix permissions to protect files, but attaches an access control list (ACL) to each directory (not each file) in AFS which controls most access according to the contents of that ACL. AFS directory ACLs supercede conventional UNIX file permissions.

Authentication

AFS access is based on limited-lifetime tokens that you get when you run kinit, login using your SCS Kerberos password, or provide authentication means to daemons running on a facilitized workstation. The global namespace that AFS encompasses also allows for configuring access rights for users in foreign cells via cross-realm authentication when supported in cooperating cells.

AFS Cells

AFS is organized into administrative units called cells. Each cell has its own accounts, file-space organization, and policies. Some of the AFS cells at CMU are:

cs.cmu.edu

Administered by SCS Facilities.

andrew.cmu.edu

Administered by CMU Computing Services.

ece.cmu.edu

Administered by ECE Facilities.

Hosts running AFS will belong to a particular cell. You can run the command:

```
fs wscell
```

to see which cell the host you are on belongs to.

By convention, on hosts running AFS, files for each cell are located under the directory `/afs/<cellname>`. For example, files in the `cs.cmu.edu` cell are located under `/afs/cs.cmu.edu/`. On hosts running the SCS Facilities environment, you can abbreviate the cellname and just use the first component of the name for AFS cells at CMU. For example: `/afs/cs/` or `/afs/andrew/`.

AFS Volumes

An AFS volume is a container for a subtree of related files and directories. Volumes are attached to mount points, which tell AFS where a volume is located in the AFS directory tree. For example, a typical SCS user named `bovik` would be contained in its own AFS volume called `user.bovik`, which would be mounted at `/afs/cs.cmu.edu/user/bovik`.

There are four main classifications of AFS space provided in the SCS computing environment.

User

Every individual with a School of Computer Science account has an AFS volume, typically accessible via a directory: `/afs/cs.cmu.edu/user/username`. Volume names are `user.username`.

Academic

Courses taught by the School of Computer Science may have AFS volumes associated with them, accessible as `/afs/cs.cmu.edu/academic/class/coursenumber-termyear`, e.g., `15100-f05`.

Faculty may also request “dropbox” space for students that will be accessible via separate `username` volumes mounted under `/afs/cs.cmu.edu/academic/class/coursenumber-termyear-users/username`.

Volumes names are `d.class.coursenum.termyear`.

Project

Research projects affiliated with the School of Computer Science may request an AFS volume, or volumes, that will be accessible from the parent directory `/afs/cs.cmu.edu/project/directory-name`.

Volumes names are `p.directoryname`.

Projects are required to list a primary and secondary SCS user account to serve as points of contact for administrative functions on project volumes.

Miscellaneous Software

The School of Computer Science maintains a distributed software environment for its "facilitized" *NIX workstations. The central repository for this software is in AFS and maintained by staff and volunteers. More information about software collections and how to become a maintainer is at http://www.cs.cmu.edu/~help/unix_linux/software_collections/.

AFS Quota

Each volume in AFS, including the volume that contains your user files, has a quota associated with it.

User volume quota

All user space classifications are allocated 1 GB of quota by default. Should additional capacity be required you may contact help+@cs.cmu.edu.

Academic volume quota

Course volume size is limited to a maximum quota of 25 GB.

Project volume quota

Project volume size is limited to a maximum quota of 25 GB; however, individual projects may request up to 50 GB of overall storage to be split across multiple volumes. Additional capacity may be acquired for a one-time fee by contacting help+@cs.cmu.edu.

If you are using an application that attempts to write to a volume that is full, it is possible that the write might fail and data, such as changes to a file you are editing, might be lost. You can use the fs command to see how much free quota is left in a volume, and Jeeves to change your AFS quota yourself, without having to contact Facilities.

How to Use the fs Command to Determine Quota Use

The fs command is in /usr/local/bin on "facilitized" Unix systems. The following commands will show current quota use for a volume:

```
fs lq <directory-name>
will list the quota information for the given directory. For example:
% fs lq /afs/cs/user/bovik
Volume Name          Quota      Used %Used  Partition
user.bovik           50000     42766   86%      59%
```

```
fs lv -dir <directory-name>
will list the current status of the named directory. The available remaining quota is the difference between the maximum quota and the number of blocks used. For example:
```

```
% fs lv /afs/cs/user/bovik
Volume status for vid = 25836 named user.bovik
Current disk quota is 50000
Current blocks used are 42766
The partition has 7023124 blocks available out of 17212287
```

Using Jeeves to change AFS Quotas

Jeeves is a telnet-based service that can be used to change AFS quotas and perform other AFS-related tasks. See the Jeeves documentation for details on how to connect to Jeeves and a list of all commands.

You can connect to Jeeves by telneting to `jeeves.srv.cs.cmu.edu` using a Kerberized telnet client, such as NiftyTelnet on a Windows PC, or the standard telnet client on Facilitized Unix hosts. Once connected, you will be presented with a menu-based interface.

All user accounts are created with a quota of 1 GB though should additional capacity be needed, please contact `help+@cs.cmu.edu`.

Project volume admins and miscellaneous software collection maintainers can also change quotas for the volumes that they are responsible for. See the Jeeves documentation for details.

An AFS volume resides on a single disk partition. If you ask for more additional quota than there is remaining space on the partition in which the volume is located, Jeeves will automatically move it to a partition with more space. Please do not increase your AFS quota to the maximum permitted if you do not need that space. We do not have enough free disk space on-line at any one time for everyone to do so.

AFS Groups

With the `pts` (protection server) command, you can create your own AFS groups and add them to AFS access control lists. AFS groups make it much easier to manage ACLs for large directory trees, and allow the addition of large numbers of users to an ACL with a single command. A typical use of AFS groups would be to create a new AFS directory that will be the root of a larger tree, and add the appropriate group to its ACL. Since a new AFS directory inherits its parent's ACL, sub-directories created in that tree will also have that group on their ACL. Adding or revoking a user's group membership will thus change access for that user throughout the entire directory tree.

System & Special Groups

In addition to user-created groups, the following system and special groups exist and have the listed membership:

system:anyuser	Anyone, anywhere.
system:campushost	Anyone on a CMU host.
system:friendlyhost	Anyone on an SCS host (except for a few exceptions --- the SCS web and anonymous FTP servers are not members of this group).
system:authuser	Anyone authenticated to the cs.cmu.edu AFS cell (people with valid SCS accounts).
system:administrators	Authorized AFS administrators (Facilities staff).
wwwsrv:http-ftp	The SCS Web and FTP servers and a few other authorized hosts.

The above special groups may be added to ACLs in the same way as user-created AFS groups.

How to create and manage AFS groups

Note: typing `pts help` will list the various `pts` commands. Most `pts` commands can be used with or without named arguments. For example,

```
pts creategroup bovik:colleagues
```

and

```
pts creategroup -name bovik:colleagues
```

will do the same thing.

Creating groups

AFS group names have the form `username:<identifier>` , and are created with the `pts creategroup` command. The username specified will be the owner of the group, and must be a valid AFS user name (you will usually want to use your own AFS username). For example, the command:

```
pts creategroup bovik:colleagues
```

would create a group called `bovik:colleagues` .

Adding and removing users

To add a user to a group, use the pts adduser command:

```
pts adduser jsmith bovik:colleagues
```

To remove a user from a group, use the pts removeuser command:

```
pts removeuser jsmith bovik:colleagues
```

Listing group members

To see a list of the members of a group, use the pts membership command:

```
pts membership bovik:colleagues
```

Examining and changing group privacy flags

You can use the pts examine command to find out information about a group (you can also use this command on a AFS username). The command:

```
pts examine bovik:colleagues
```

would produce the following output:

```
Name: bovik:colleagues, id: -3745, owner: bovik, creator:  
bovik,  
membership: 2, flags: S-M--, group quota: 0.
```

The above fields have the following meanings:

Name	The name of the group.
Id	A unique identification number for the group that AFS users internally.
Owner	The owner of the group
Creator	The person who originally created the group.
Membership	How many members belong to the group.
Flags	Group privacy flags that determine who can list group properties or make certain changes to the group. See below for details.
group quota	How many more groups a user is allowed to create.

The five group privacy flags appear in the following order:

Status (s)	Controls who can use <code>pts examine</code> to list status information about a group.
Owned (o)	Controls who can use <code>pts listowned</code> to list groups owned by a group or user.
Membership (m)	Controls who can use <code>pts membership</code> to list groups a user belongs to, or users that belong to a group.
Add (a)	Controls who can use <code>pts adduser</code> to add a user to a group.
Remove (r)	Controls who can use <code>pts removeuser</code> to remove a user from a group.

Each one of the flags, `somar`, has three possible values:

A hyphen, "-", gives rights only to the group's owner (along with members of the system group `system:administrators`, which only has SCS Facilities staff as members).

A lowercase version of the flag (eg a lowercase "s") gives rights to members of the group, in addition to those who have "hyphen" rights.

An uppercase version of the flag gives rights to anyone.

The default values of S-M-- gives anyone the ability to examine a group and see who belongs to a group, and only gives the owner of the group the other rights. You can use the `pts setfields` command to change these default values. Type `pts help setfields` for details about the syntax of this command.

Access Control Lists (ACLs)

AFS uses access control lists (ACLs) to determine permissions for accessing data. An ACL is a set of Kerberos instances, IP addresses, and/or AFS groups along with an associated AFS permission.

For example, the ACL for the directory `/afs/cs.cmu.edu/user/bovik` has entries for:

```
wwsrv:http-ftp rl
system:anyuser rl
bovik rlidwka
```

The above ACL gives just "read" and "lookup" rights to the special groups `wwsrv:http-ftp` and `system:anyuser`, and all AFS ACL permissions to the user "bovik".

ACLs allow very flexible control over who may access data in AFS. Some features of ACLs and AFS access permissions are:

- ACLs apply only to directories in AFS, not to files.
- AFS ignores standard Unix permissions (the ones you set with the `chmod` command), with the exception of the file owner mode bits (see the section below on protecting individual files for details).
- The owner of a directory can always change the ACL on that directory, no matter what the ACL is (so you can fix things if you accidentally remove yourself from the ACL of a directory you own).
- When you create a directory, it automatically inherits the ACL of its parent directory.
- In order to access a subdirectory, one must have "l" (lookup) permissions on all parent directories.

Note: Because top-level AFS user directories in SCS are created by default with fairly liberal ACLs, you may need to take special precautions to protect confidential information in AFS. For example, the "mbox" file created by some Unix mail programs and Emacs "shell.CKP" files will not be protected by the standard Unix permissions if they are created in AFS. You should make sure that directories that are used to store e-mail and other sensitive files have appropriate ACLs.

AFS Permissions and Their Meaning

There are seven standard AFS permissions, each referred to by one of the letters r, l, i, d, w, k and a. The lida permissions apply to directories and the rwk permissions apply to files.

Directory permissions

l (lookup)	Allows one to list the contents of a directory. It does not allow the reading of files.
i (insert)	Allows one to create new files in a directory or copy new files to a directory.
d (delete)	Allows one to remove files and sub-directories from a directory.
a (administer)	Allows one to change a directory's ACL. The owner of a directory can always change the ACL of a directory that s/he owns, along with the ACLs of any subdirectories in that directory.

File permissions

r (read)	Allows one to read the contents of file in the directory.
w (write)	Allows one to modify the contents of files in a directory and use chmod on them.
k (lock)	Allows programs to lock files in a directory.

Normal and negative AFS permissions

An ACL can be either "normal" or "negative". Normal rights grant the specified access permissions, while negative rights allow one to cancel specific permissions for a user or group on an ACL.

How to list and change AFS ACLs

The fs command (which should be in /usr/local/bin on Facilitized Unix hosts) is used to list and change ACLs. When specifying directory rights using fs, the following shortcuts may be used:

All	Means the same as rlidwka (all rights).
Read	Means the same as rl (read and lookup rights).
Write	Means the same as rlidwk (all rights except the ability to change the ACL).
None	Removes the entry from the ACL (removing both any positive and negative rights that may exist).

How to list an ACL

The command `fs listacl <directory-name>` will list the ACL of a directory ("listacl" is usually abbreviated to "la"). For example:

```
fs la /afs/cs/user/bovik
```

will produce the output:

```
Access list for /afs/cs/user/bovik is
Normal rights:
  wwwsrv:http-ftp rl
  system:anyuser rl
  bovik rlidwka
```

You can use the command `fs help listacl` to list the complete set of options.

How to add a user or group to an ACL

The command `fs setacl -dir <directory> -acl <acl entries>` will add the given ACLs to the given directory. For example:

```
fs setacl -dir /afs/cs/user/bovik -acl jsmith rl
```

will give the user "jsmith" read and lookup rights on the directory `/afs/cs/user/bovik`. As a shortcut, you can abbreviate "setacl" to "sa" and/or leave out the "-dir" and "-acl" as long as you maintain the arguments in the order given in the above example:

```
fs sa /afs/cs/user/bovik jsmith rl
```

is the same command as the previous example, but using abbreviated syntax.

You can also use the "-clear" switch on a `fs setacl` command to completely clear the previous ACL when setting new entries. Be careful not to remove your own administrative rights when doing so.

How to remove a user or group from an ACL

To remove a user or group from an ACL, give, assign the user or group the access permission "none". For example:

```
fs setacl -dir /afs/cs/user/bovik -acl jsmith none
```

would remove the user "jsmith" from the access list for `/afs/cs/user/bovik`.

How to set negative ACL entries

To set negative ACL entry, use the `-negative` switch to the appropriate `fs` command. For example:

```
fs setacl -dir /afs/cs/user/bovik -acl jsmith rl -negative
```

would set negative read and lookup rights for user "jsmith".

How to copy ACLs

To copy an ACL for *dir1* to *dir2* use the command:

```
fs copyacl -fromdir <dir1> -todir <dir2>
```

This command will copy all ACL entries from *dir1* to *dir2*, overwriting those on *dir2* that already exist and keeping the ones that don't conflict with ACL entries on *dir1*. You can use the "-clear" switch to completely replace the ACL of *dir2*.

How to change the ACLs of directory tree in AFS

If you want to set an ACL for a directory in AFS and all of its sub-directories, you can use one of the following commands:

```
find <directoryname> -type d -exec fs sa -acl <acl> -dir '{} ' ';' 
```

or

```
find <directoryname> -type d -print | xargs fs sa -acl <acl> -dir
```

For example, the commands:

```
find /afs/cs.cmu.edu/user/bovik/public -type d -exec fs sa -acl system:anyuser rl -dir '{} ' ';' 
```

and

```
find /afs/cs.cmu.edu/user/bovik/public -type d -print | xargs fs sa -acl system:anyuser rl -dir
```

would both do the same thing -- add the acl *system:anyuser rl* to the directory */afs/cs.cmu.edu/user/bovik/public* and all of its subdirectories. Be careful that the directory you give is not a symlink, since the find program will not follow symlinks by default. The second version of the command may be faster for large directory trees.

How to protect individual files

AFS ignores all but the owner Unix mode bits on files. Turning off the "r" bit removes read access to a file for everyone, including the owner. Turning off the "w" bit removes write access to a file for everyone, including the owner. Turning off the "x" bit disallows execution of a file for everyone, including the owner.

How to make an AFS directory private

To make an AFS directory so that only you can read & administer the contents, you should remove all entries ACLs except one for you. A quick way to do this is with the command:

```
fs setacl; <directory> your-username all&nbsp;clear.
```

For example:

```
fs setacl /afs/cs/user/bovik/private bovik all -clear
```

Note that this command will remove all access for others to all subdirectories of the given directory. If you want others to access subdirectories, but still prevent reading of files in the top-level directory, you should add "l" access for selected users or groups to the top-level directory's ACL.

How to make a "drop box" directory

A "drop box" directory is a directory that people can copy files to but not read, delete, or write to files that are already in the directory. For example, if one were teaching a class one could use such a directory as an upload area for homeworks. To create such a directory, create a private directory and then add "il" only rights for the pts group or user that should have rights to create files in that directory.

AFS Authentication

AFS does secure authentication through tokens that are usually obtained by interactively typing a password, either when logging in or by running a program such as kinit. The token you receive is used to verify your identity to the AFS servers when accessing files. Tokens have limited lifetimes (typically 25 hours) and need to be periodically renewed. Users and processes which are not authenticated to AFS typically only have the access rights system:anyuser. A user can only have one token per cell at any given time.

AFS & Kerberos

AFS uses Kerberos as the basis for its authentication tokens. Kerberos provides a mechanism where a given Kerberos name, such as "bovik" can have separate instances such as "bovik.root". AFS also supports the use of such instances. Each instance has its own password and AFS will use a unique ID for the corresponding AFS instance.

AFS IDs & Unix IDs

A Unix ID is a number that corresponds to a particular user on a Unix host. An AFS ID is a number that corresponds to a particular AFS username or instance name (eg "bovik" or "bovik.ftp"). AFS uses the AFS ID, instead of the name, internally. There is no necessary relation between AFS IDs and Unix IDs, but we keep them synchronized when creating AFS and Unix user IDs. A given Unix user ID may be authenticated to any AFS ID for which a token can be obtained.

Process authentication groups

Since AFS authentication tokens are associated with a Unix user, the AFS client uses the concept of a PAG to allow multiple sessions for the same Unix user to have different AFS tokens. Without a PAG, all Unix sessions for a given Unix user would share the same AFS tokens. In addition, a PAG is inherited by subprocesses, so a setuid program can use the AFS authentication token associated with the PAG.

By using a PAG, different sessions for the same Unix user can have a different set of AFS tokens. Obtaining and destroying such tokens does not interfere with the tokens of another Unix session for the same Unix user.

If a Unix user does not have a PAG, then the AFS tokens are associated with the Unix user ID (numeric). In this case, all sessions for the same Unix ID without a PAG will share the same set of tokens.

The AFS cache manager & authentication

Files on AFS are stored on servers (the SCS AFS servers are located in the SCS machine room). Access to these files is controlled by a cache manager which runs on your local host. The cache manager keeps an on-disk cache of files that have been previously accessed. When you attempt to access data on AFS, it checks to see if the data is in this cache, and if not it gets a copy of the data from a file server that has it. Consistency between the data in the local cache and the data on the file server is maintained through use of a callback mechanism where the server notifies the cache manager of any changes to the data.

The tokens obtained when authenticating to AFS are used by the cache manager when requesting data from AFS servers. To distinguish between one user's tokens and another's, the cache manager keeps track of tokens by using either the user's Unix ID or a process authentication group (PAG).

How to list your AFS tokens

The tokens command will list your AFS tokens and produce output like the following:

```
Tokens held by the Cache Manager:

User's (AFS ID 2102) tokens for afs@cs.cmu.edu [Expires Jun 13
22:04]
--End of list--
```

To see the name of the user that corresponds to the given AFS id, use the command:

```
pts examine <AFS ID>
```

For example:

```
pts examine 2102
```

How to get AFS tokens on Unix hosts

You will automatically get AFS tokens for the cs.cmu.edu AFS cell on a Facilitized Unix host when you login to the host by typing your password (as opposed to autologging in via telnet or SSH). You can use the command:

```
kinit <username>
```

to get tokens or renew tokens. For example:

```
kinit bovik
```

and then type your SCS Kerberos password at the prompt.

To get tokens for another AFS cell, use the klog command:

```
klog <username-in-foreign-cell> -c <cellname>
```

For example:

```
klog hb2q -c andrew.cmu.edu
```

and type your password for the foreign AFS cell at the prompt.

How to change your AFS password

Your AFS password for the cs.cmu.edu AFS cell is exactly the same as your SCS Kerberos password. You can use Jeeves or the command `passwd -k` to change this password. If you want to change your AFS password in another AFS cell, use the command:

```
vpasswd <username-in-other-cell> -c <cellname>
```

Cross-realm Authentication

Cross-realm AFS authentication allows users in one Kerberos realm (a Kerberos "realm" is an administrative domain such as CS or Andrew) to manipulate files in another realm without having to authenticate separately in each one. AFS cells that support cross-realm authentication are:

- cs.cmu.edu
- andrew.cmu.edu
- club.cc.cmu.edu
- dementia.org
- athena.mit.edu

Setting up cross-realm authentication

To set up cross-realm authentication, you need to run the `aklog` command, while authenticated to your local cell, giving it the name of the foreign AFS cell that you will be authenticating to. Then you will need to create an entry in the foreign cell's pts database. Previously, this happened automatically. For example, if you are on a host in the cs.cmu.edu cell and want to do cross-realm authentication with the andrew.cmu.edu cell, you should run:

```
aklog andrew.cmu.edu
```

If you are on a host in the andrew.cmu.edu cell and want to do cross-realm authentication with cs.cmu.edu, you should run:

```
aklog cs.cmu.edu
```

Running `aklog` does two things:

- It provides cross-realm "tokens" that can be used to access files in the foreign cell.
- It checks to see that there is an entry for these cross-realm tokens in the protection server (pts) database in the foreign cell. If such an entry doesn't already exist, one is created. The cross-realm entry will have the form *userid@foreigncell*.

After running `aklog`, run the command:

```
pts createuser <username>@<localcell> -cell <foreigncell>
```

This creates an entry for you in a foreign cell's protection database. If you want to verify that an entry for your cross-realm tokens exists in the foreign cell's pts database, you can enter the following command:

```
pts examine <username>@<localcell> -cell <foreigncell>
```

To see a list of which Andrew or ECE users have established cross-realm identities in the cs.cmu.edu cell, you can use the command:

```
pts members system:authuser@{ece or andrew}.cmu.edu
```

To see a list of CS users who have established cross-realm identities in the andrew or ece AFS cells, you can use the command:

```
pts members system:authuser@cs.cmu.edu -cell {ece or andrew}.cmu.edu
```

Note that you will need to run aklog to get cross-realm tokens before accessing files in the other cell. If you are frequently accessing files in another cell, you may wish to put:

```
aklog <foreigncell>
```

in your .login.

Adding a cross-realm ID to ACLs and groups

Once a cross-realm ID has been created, you can add it to AFS ACLs and groups the same way you would add a user in the local cell to ACLs and groups. For example, to add the Andrew username "hb0v" to an ACL:

```
fs sa my_directory hb0v@andrew.cmu.edu rl
```

and to a group:

```
pts adduser hb0v@andrew.cmu.edu bovik:colleagues
```

Security

By granting permission to access your files in the cs.cmu.edu cell to your username in another realm, you have created the possibility that somebody could break into your other account and access your CS files (the same concept applies if the andrew.cmu.edu cell is where you keep most of your important files and you have granted cross-realm access to your CS username). For this reason, it is suggested that you only add your other realm's username to directories when it is necessary to do so.

Authenticating Daemons to AFS

In normal situations, daemons that are (for example) run as background processes, do not have authenticated access to AFS. This section describes a mechanism for running background daemons so that they can obtain such access. The procedures described here may not work on non-facilitized hosts.

Some of the steps in this document may seem to be unnecessary, but the intent is to provide a simple and portable mechanism for running background daemons so that they can have authenticated AFS access.

How to do it

The basic procedure for running authenticated daemons involves the following steps (steps 1-3 are setup steps that should only have to be done once):

- Use Jeeves to create a `<username>.daemon` instance and associated AFS identity (where `<username>` is your SCS username). See the Jeeves documentation for complete instructions on using Jeeves. Note: you may need to ask the SCS Help Desk to do this step for you).
 - Connect to Jeeves.
 - Choose the **kerberos** menu item.
 - Select: **key <username>.daemon** and record the Kerberos key that Jeeves will print out. Contact the SCS Help Desk if you get an error when performing this step.
 - Choose **back** and go to the top-level Jeeves menu.
 - Choose then **afs** menu item and then the **account** menu item.
 - Select: **define <username>.daemon** to create the AFS identity.
- Add the `<username>.daemon` instance to AFS ACLs as necessary to minimize any potential security risks, you should only add this instance to the ACLs of directories that the daemon needs to access.
- **Create key and keytab files**
 - To do this step, you will need to know the 16 character hex string for the Kerberos key of the `.daemon` instance you created in step 1.
 - Login to the machine that will be running the daemon and run the following two commands:

```
umask 077
cd /etc/not-backed-up
```

This is to ensure that the keyfiles are not readable and will not be backed up (since backed-up data goes over the network in the clear). If you need to use a more convenient name or location for the files you will create in this directory, use a symbolic link.

- Rename/remove any existing `<username>.daemon.*` in that directory.
- Run the following command:

```
/usr/local/etc/ksrvhost <username> =daemon
```

and type in the Kerberos hex string you got from Jeeves at the prompt.

- Run the following command to create the Kerberos 5 keytab file:

```
/usr/local/etc/dokeytab <username>.daemon.keytab
<username>.daemon
```

You should now have created the following two files:

```
/etc/not-backed-up/<username>.daemon
/etc/not-backed-up/<username>.daemon.keytab
```

Both files should be readable only by the owner of the files, and the owner should be the same as the Unix user which the daemon will be running as. If that is not the case, use *chmod* and *chown* to change ownership and modes to the correct ones.

Running the daemon

It is important that the daemon is run in a way that it gets its own PAG before starting. If it does not do so, all processes without a PAG with the same Unix user ID as the daemon will share the same token and perhaps have undesired authenticated access to AFS. The general procedure to follow is: Before starting the daemon:

- Get a new PAG.
- Set unique names for KRBTKFILE and KRB5CCNAME in the process environment.
- Obtain the Kerberos tickets, including an AFS service ticket and store the AFS token.

Now execute the daemon. At this point, the daemon will inherit the Kerberos tickets (via the KRBTKFILE and KRB5CCNAME in the environment) and the AFS tokens (via the PAG which is shared by all descendents of the PAG creator).

After the daemon is running:

- Have some mechanism to periodically renew tickets before they expire (the default ticket lifetime is usually 25 hours).
- Have some mechanism to detect when the daemon has exited, either via looking for a */var/run/<daemon>.pid* file or through some other means.

As long as the process that is maintaining the tickets has the same PAG and the same KRBTKFILE and KRB5CCNAME then tickets for the daemon process can be maintained forever.

The following wrapper Perl script is an example of how to do the above steps: *run_with_tickets*

Before using this script, you should read through it and modify variables appropriately for your application. This script has not been extensively tested.

The sample "run_with_tickets" script uses a child watcher process to maintain the tickets. It exits when it detects that its parent process is gone. The parent process is assumed to be the daemon (or some indefinitely running process) which is exec-ed to replace the parent perl process.

It is common for daemon processes to use a paradigm where the daemon will fork, the parent will exit, and the child will continue execution thus placing itself in the background. Such daemon processes must be started with a switch so that the daemon does not perform this forking operation.

If the daemon does not support startup without placing itself in the background, then you will need maintain your own script using the above guidelines.

If the daemon does support the "run_with_tickets" script, then you can start your daemon with a "launch_daemon" script that contains the following:

```
/usr/mypkg/etc/run_with_tickets /usr/mkpkg/bin/my_daemon arg1  
arg2 >/some/log/file 2>&1 &
```

The above example assumes a sh-based script with associated files located under "/usr/mypkg"

AFS Command Commands and Troubleshooting

Basic Information and Common Commands

AFS access permissions

Directory permissions

l (lookup)	Allows one to list the contents of a directory. It does not allow the reading of files.
i (insert)	Allows one to create new files in a directory or copy new files to a directory.
d (delete)	Allows one to remove files and sub-directories from a directory.
a (administer)	Allows one to change a directory's ACL. The owner of a directory can always change the ACL of a directory that s/he owns, along with the ACLs of any subdirectories in that directory.

File permissions

r (read)	Allows one to read the contents of file in the directory.
w (write)	Allows one to modify the contents of files in a directory and use chmod on them.
k (lock)	Allows programs to lock files in a directory.

Special AFS groups

system:anyuser	Anyone, anywhere.
system:campushost	Anyone on a CMU host.
system:friendlyhost	Anyone on an SCS host (except for a few exceptions --- the SCS web and anonymous FTP servers are not members of this group).
system:authuser	Anyone authenticated to the cs.cmu.edu AFS cell (people with valid SCS accounts).
system:administrators	Authorized AFS administrators (Facilities staff).
wwwsrv:http-ftp	The SCS Web and FTP servers and a few other authorized hosts.

Using the fs command to list your AFS quota

```
fs lq <directory-name>
```

Using the fs command to list/change ACLs

fs help will list all fs commands. fs help <command> will list the syntax for the given command.

An ACL entry is of the form:

```
<user_or_group name> <permissions>
```

For example:

```
system:anyuser rl
```

To list a directory's ACL

```
fs la <directory-name>
```

To add a user or group to an ACL

```
fs sa -dir <directory> -acl <acl entries>
```

To remove a user or group from an ACL

```
fs sa -dir <directory> -acl <username_or_groupname> none
```

Using the pts command to manipulate AFS groups

pts help will list all pts commands. pts help <command> will list the syntax for a given command. User-created groups should have names of the form: your_userid:<name> .

To create a group

```
pts creategroup <name>
```

To add a user to a group

```
pts adduser -user <user;>&nbsp;&nbsp;&nbsp;-group <group>
```

To remove a user from a group

```
pts removeuser -user <user>&nbsp;&nbsp;&nbsp;-group <group>
```

To list the membership of a group

```
pts membership <group>
```

To list the groups a user belongs to

```
pts membership <user>
```

To find out information about a group

```
pts examine <group>
```

To find out information about a user

```
pts examine <user>
```

To delete a group

```
pts delete <group>
```

Troubleshooting and Problem Solving

Some of the common types of problems and possible solutions are listed here. However, if you are ever in doubt for how to deal with a specific problem contact the helpdesk by phone or email.

Permission & authentication problems

The most common reason for "permission denied" errors is that one's AFS tokens have expired. You can use the tokens command to see the AFS tokens that you hold. If you do not have valid tokens, you can run kinit to authenticate and get new tokens. If you do have valid tokens, you should double-check that you are in the correct ACLs for the directories that you are trying to access. Note that if you are added to an AFS group, you will need to re-authenticate before you can use the permissions for that group.

With very few exceptions all SCS hosts should be a member of the system:friendlyhost AFS group. If you have trouble accessing files from your machine that should be accessible by system:friendlyhost, contact Facilities, and we will add that machine to friendlyhosts. One consequence of being a friendlyhost is that, if you are running a webserver on your machine, you should not allow the server to access files in /afs/cs, as that would circumvent the purpose of the system:friendlyhost access controls.

Network problems

If attempts to access a directory in AFS result in hangs or the directory not being found, you should first check to see if your machine is seeing the network. You can use the ping command to check network connectivity to other hosts (choose a host in the SCS machine room, like *cs.cmu.edu*). If you know which fileserver the directory is located on, you can ping that fileserver.

Local caching problems

A common cause of AFS problems is either a lack of disk space on the partition where a host's AFS cache is located or corruption of the local AFS cache. If you have trouble accessing files in AFS from a particular host, but can access these files from other hosts, then local AFS cache corruption or disk space issues may be the problem. Note that you will need root access on your machine to do this, and you should contact the Help Desk if you do not have root and/or are not comfortable fixing this type of problem.

Occasionally, the AFS cache on a machine may become corrupted. Symptoms of this problem include an inability to access certain files or directories in AFS, or being unable to run a binary located on AFS without an immediate core dump. If these symptoms occur, you should verify that it is a local problem (as opposed to an AFS server problem) by seeing if it occurs on other machines of that type, or comparing checksums for the binaries between the machine having the problem and other machines. You can use the command

```
/usr/local/bin/fs checkservers
```

to check on the status of AFS servers that your local machine's cache manager has recently contacted. The command

```
fs checkvolumes
```

will check the status of alternate locations of volumes for replicated collections, and may fix some problems. If the problem is local to a particular machine, then it is very possibly caused by AFS cache corruption. There are a few things to try to fix the problem. If it is a single file or volume, you can run

```
fs flush path-to-file
```

or

```
fs flushv path-to-volume
```

Alternatively, you can try interactively reducing the cache size, and then increasing it back to the default.

To do so, run

```
fs setcachesize 20 (or some other small number)
```

wait until that command completes, and then run

```
fs setcachesize -reset
```

to reset it to its original size. Sometimes, in cases of severe corruption, the above procedures may not be the problem. In order to completely clear the cache, remove the Cacheltems file in the cache directory and reboot (instead of rebooting, you could try manually stopping and starting AFS using the appropriate rc script and arguments, but that does not always work).

AFS fileserver issues

The following commands can be useful in determining whether something is an AFS fileserver issue:

```
/usr/local/bin/fs checkservers
```

checks on the status of AFS fileservers that your local machine's cache manager has recently contacted.

```
/usr/local/etc/vos examine <volume>
```

reports on location and status of the given volume. Note that you will have to give the volume name (not the path to the volume). For example:

```
/usr/local/etc/vos examine user.bovik
```

You can get a volume name by using the `fs lv <path>` command. For example:

```
fs lv /afs/cs/user/bovik
```

The volume name for a user will always be of the form *user.username*.

```
/usr/local/bin/fs checkvolumes
```

will check the status of alternate locations of volumes for replicated collections. Doing so may fix some types of AFS access problems.