

OpenAFS for Mac OS X

Jean-Matthieu Schaffhauser
schaffhauser.jm@euro.apple.com

April 8, 2003

Contents

1	Overview	3
1.1	What is AFS ?	3
1.2	What is a Cell ?	3
1.3	What are the benefits of using AFS ?	3
1.4	What is this document ?	3
2	Requirements	4
2.1	Obtaining OpenAFS	4
3	Server Installation	5
3.1	Use the source, Luke	5
3.1.1	OpenAFS Security Advisory 2003-001	5
3.1.2	Patch the source, Luke !!	5
3.1.3	Building OpenAFS on Jaguar	5
3.2	Before we start	6
3.3	Starting the AFS Server	7
3.4	Defining Cell Name and Membership for Server Process	7
3.5	Starting the Database Server Process	7
3.6	Initializing Cell Security	8
3.7	Starting the File Server, Volume Server and Salvager	10
3.8	Starting the Server Portion of the Update Server	11
3.9	Installing Client Functionality	11
3.9.1	Defining Cell Membership for Client Processes	12
3.9.2	Creating the Client CellServDB File	12
3.9.3	Configuring the Cache	13
3.9.4	Configuring the Cache Manager	13
3.10	Shutdown and initialization script	14
3.11	Configuring the Top Level of the AFS filesystem	15
4	Client installation	15
5	References	16

1 Overview

1.1 What is AFS ?

AFS is a distributed filesystem that enables co-operating hosts (clients and servers) to efficiently share filesystem resources across both local area and wide area networks. Clients hold a cache for often used objects (files), to get quicker access to them.

AFS is based on a distributed file system originally developed at the Information Technology Center at Carnegie-Mellon University that was called the "Andrew File System". "Andrew" was the name of the research project at CMU - honouring the founders of the University. Once Transarc was formed and AFS became a product, the "Andrew" was dropped to indicate that AFS had gone beyond the Andrew research project and had become a supported, product quality filesystem. However, there were a number of existing cells that rooted their filesystem as /afs. At the time, changing the root of the filesystem was a non-trivial undertaking. So, to save the early AFS sites from having to rename their filesystem, AFS remained as the name and filesystem root.

IBM branched the source of the AFS product, and made a copy of the source available for community development and maintenance. They called the release **OpenAFS**.

1.2 What is a Cell ?

An AFS cell is a collection of servers grouped together administratively and presenting a single, cohesive filesystem. Typically, an AFS cell is a set of hosts that use the same Internet domain name (like for example apple.com). Users log into AFS client workstations which request information and files from the cell's servers on behalf of the users. Users won't know on which server a file which they are accessing, is located. They even won't notice if a server will be located to another room, since every volume can be replicated and moved to another server without user an user noticing. The files are always accessible. Well it's like NFS on steroids :)

1.3 What are the benefits of using AFS ?

The main strengths of AFS are its: caching facility (on client side, typically 100M to 1GB), security features (Kerberos 4 based, access control lists), simplicity of addressing (you just have one filesystem), scalability (add further servers to your cell as needed), communications protocol.

1.4 What is this document ?

This document aims to be a step-by-step tutorial explaining how to properly set up OpenAFS on a Mac OS X. Through the next pages, we'll detail how to get a server ready to distributed files using AFS as well as how a client can be easily configured to access shared resources on a network.

This document is based on OpenAFS original documentation.

2 Requirements

We won't detail here OpenAFS structure. Basically, rights are granted thanks to tickets one can obtain after a valid authentication. Like in a Kerberos scheme, a ticket has a specified lifetime. When time comes, the ticket has to be renewed. As you probably guessed, time is not an option here. To ensure the validity of a ticket, we'll setup both client and server to use the very accurate and practical Network Time Protocol, aka NTP. Nothing's easier on Mac OS X, just open the **System Preferences** dialog, click on **Date & Time** and select the **Network Time** tabview. Check the *Use a network time server* box and select a NTP server in the corresponding combo list as show in Figure 1.

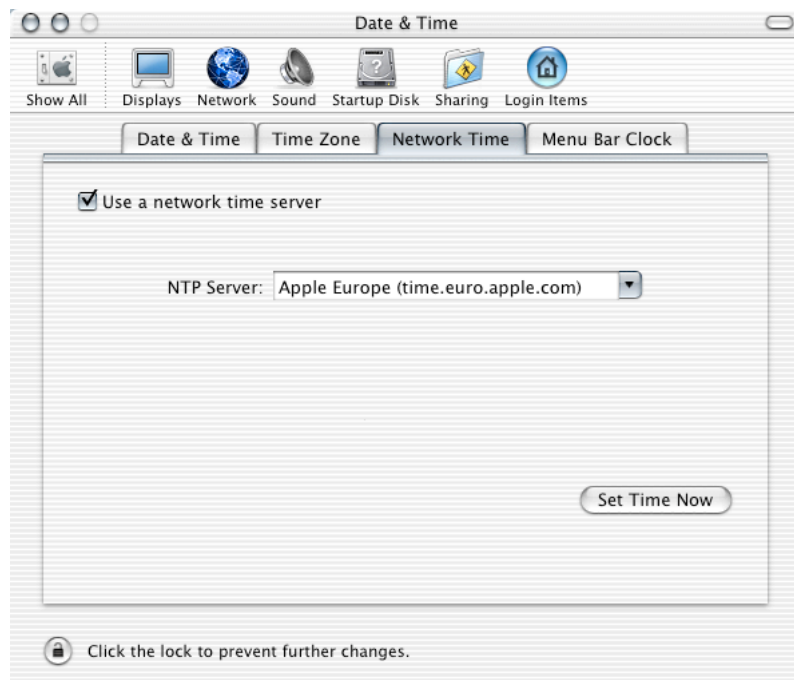


Figure 1: Configuring Network Time Protocol on Mac OS X

2.1 Obtaining OpenAFS

OpenAFS is freely available from <http://www.openafs.org>. In this tutorial, we'll use the binary package for Mac OS X available from OpenAFS-1.2.8 release page. Just browse through [openafs.org](http://www.openafs.org) pages to obtain **OpenAFS.pkg**. Double-click on the file freshly downloaded and follow the installation procedure as usual. Note that you'll use this same package whether you want to setup an AFS server or an AFS client host.

OpenAFS is now deployed on your system. It is time to configure your first AFS server.

3 Server Installation

3.1 Use the source, Luke

If you don't want to use the package for Mac OS X provided by OpenAFS.org, you can always build it yourself. Here's what you've got to. First, download `openafs-1.2.8` source code and store it, for example, in `/usr/src`. Otherwise, go on to Section 3.2.

3.1.1 OpenAFS Security Advisory 2003-001

The OpenAFS Project has just released a security advisory detailing a cryptographic weakness in Kerberos v4 which allows an attacker to compromise OpenAFS servers.

A cryptographic weakness in version 4 of the Kerberos protocol allows an attacker to use a chosen-plaintext attack to impersonate any principal in a realm. OpenAFS `kaserver` implements version 4 of the Kerberos protocol, and therefore is vulnerable. An attacker that knows a shared cross-realm key between any remote realm and the local realm can impersonate any principal in the local realm to AFS database servers and file servers in the local cell, and other services in the local realm. An attacker that can create arbitrary principal names in a realm can also impersonate any principal in that realm.

If your realm has no shared keys, and does not allow users to create arbitrary principal names, you are not exposed to this vulnerability.

There are no known publicly-available exploits for this vulnerability at this time.

3.1.2 Patch the source, Luke !!

Here's how you'll patch OpenAFS. You'll first `untar openafs-1.2.8-src.tar.gz` and then apply the patch released by the OpenAFS Project.

```
# cd /usr/src && tar -zxvf openafs-1.2.8-src.tar.gz
# cd openafs-1.2.8
# patch -p 1 < ../kaserver-disable-krb4-crossrealm-20030317.delta
```

3.1.3 Building OpenAFS on Jaguar

Here are details on how you should build OpenAFS on Mac OS X. Using `configure` in the top level directory, `configure` for your AFS system type, providing the necessary flags:

```
#!/configure --with-afs-sysname=ppc_darwin_60 --enable-transarc-paths
```

There are two modes for directory path handling: "Transarc mode" and "default".

- In Transarc mode, we retain compatibility with Transarc/IBM AFS tools by putting client configuration files in `/usr/vice/etc`, and server files in `/usr/afs` under the traditional directory layout.
- In default mode, files are located in standardized locations, usually under `$(prefix)`.

Client programs, libraries, and related files always go in standard directories under `$(prefix)`. This rule covers things that would go into `$(bindir)`, `$(includedir)`, `$(libdir)`, `$(mandir)`, and `$(sbindir)`.

Other files get located in the following places:

Directory	Transarc mode	Default mode
viceetcdir	/usr/vice/etc	\$(sysconfdir)/openafs
afssrvdir	/usr/afs/bin (servers)	\$(libexecdir)/openafs
afsconfdir	/usr/afs/etc	\$(sysconfdir)/openafs/server
afslocaldir	/usr/afs/local	\$(localstatedir)/openafs
afsdmdir	/usr/afs/db	\$(localstatedir)/openafs/db
afslogdir	/usr/afs/logs	\$(localstatedir)/openafs/logs
afsbosconfig	\$(afslocaldir)/BosConfig	\$(afsconfdir)/BosConfig
afsbosserver	\$(afsbindir)/bosserver	\$(sbindir)/bosserver

Once your software is properly configured it is time to compile it. Issue the following command and go grab a cup of coffee.

```
# make
```

Next, run

```
# make dest
```

This command will create a Transarc-style dest tree in `SYS_NAME/dest` where `SYS_NAME` is the AFS sysname of the system you built for. Finally, copy this directory to the folder you'd like OpenAFS to be installed in.

```
# cp -R dest /Library/OpenAFS/Tools
```

You're done ! You've successfully build and install OpenAFS on your system. Let's move on to the real stuff.

3.2 Before we start ...

This section is all about restoring OpenAFS standard hierarchy. I realised while trying to make it work that it could help a lot to issue the following commands :

```
# ln -s /Library/OpenAFS/Tools/root.server/usr/afs/ /usr/afs
# ln -s /Library/OpenAFS/Tools/root.client/usr/vice /usr/vice
# cp -pR /var/db/openafs/etc/* /usr/afs/etc
# rm -rf /var/db/openafs/etc && ln -s /usr/afs/etc /var/db/openafs/etc
```

One more thing and we'll be totally ready. I recommend you add the following paths to your shell `$PATH` variable. Using `bash`, issue the following command :

```
# export PATH=$PATH:/Library/OpenAFS/Tools/bin:/usr/afs/bin
```

If you prefer `tcsh`, or `csh`, here's what you have to do

```
# setenv PATH $PATH:/Library/OpenAFS/Tools/bin:/usr/afs/bin
```

Add this to your `/.bashrc` or `/.cshrc` for extra fun :)

3.3 Starting the AFS Server

Make sure no CellServDB nor ThisCell file exists. Remove them if they do:

```
# rm -f /usr/afs/etc/{ThisCell,CellServDB}
```

Next you will run the `bosservice` command to initialize the *Basic OverSeer (BOS)Server*, which monitors and controls other AFS server processes on its server machine. Think of it as `init` for the system. Include the `-noauth` flag to disable authorization checking, since you haven't added the admin user yet.

Warning : Disabling authorization checking gravely compromises cell security. You must complete all subsequent steps in one uninterrupted pass and must not leave the machine unattended until you restart the BOS Server with authorization checking enabled. Well this is what the AFS documentation says :)

```
# bosservice -noauth &
```

Verify that the BOS Server created `/usr/afs/etc/CellServDB` and `/usr/afs/etc/ThisCell`

```
# ls -l /usr/afs/etc/
lrwxr-xr-x  1 root  wheel   23 Mar 25 17:35 CellServDB
lrwxr-xr-x  1 root  wheel   21 Mar 25 17:35 ThisCell
```

3.4 Defining Cell Name and Membership for Server Process

Now assign your cell's name. There are some restrictions on the name format. Two of the most important restrictions are that the name cannot include uppercase letters or more than 64 characters. Remember that your cell name will show up under `/afs`, so you might want to choose a short one.

Note : In the following and every instruction in this guide, for the `server name` argument substitute the full-qualified hostname (such as `jimmy.lab.euro.apple.com`) of the machine you are installing. For the `cell name` argument substitute your cell's complete name (such as `afslab`)

Run the `bos setcellname` command to set the cell name:

```
# bos setcellname <server name> <cell name> -noauth
```

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

```
# bos listhosts <server name> -noauth
Cell name is <cell name>
  Host 1 is <server name>
```

3.5 Starting the Database Server Process

Next use the `bos create` command to create entries for the four database server processes in the `/usr/afs/local/BosConfig` file. The four processes run on database server machines only.

1. **kaserver**
The Authentication Server maintains the Authentication Database. This can be replaced by a Kerberos 5 daemon.
2. **buserver**
The Backup Server maintains the Backup Database
3. **ptserver**
The Protection Server maintains the Protection Database
4. **vlserver**
The Volume Location Server maintains the Volume Location Database (VLDB).

```
# bos create <server name> kaserver simple /usr/afs/bin/kaserver \  
-cell <cell name> -noauth  
# bos create <server name> buserver simple /usr/afs/bin/buserver \  
-cell <cell name> -noauth  
# bos create <server name> ptserver simple /usr/afs/bin/ptserver \  
-cell <cell name> -noauth  
# bos create <server name> vlserver simple /usr/afs/bin/vlserver \  
-cell <cell name> -noauth
```

Verify all server are running :

```
# bos status <server name> -noauth  
Instance kaserver, currently running normally.  
Instance buserver, currently running normally.  
Instance ptserver, currently running normally.  
Instance vlserver, currently running normally.
```

3.6 Initializing Cell Security

Now we'll initialize the cell's security mechanisms. We'll begin by creating the following two initial entries in the Authentication Database: The main administrative account, called **admin** by convention and an entry for the AFS server processes, called **afs**. No user logs in under the identity **afs**, but the *Authentication Server's Ticket Granting Service* (TGS) module uses the account to encrypt the server tickets that it grants to AFS clients. This sounds pretty much like Kerberos .

Enter kas interactive mode

```
# kas -cell <cell name> -noauth  
ka> create afs  
initial_password: afs_passwd  
Verifying, please re-enter initial_password:  
ka> create admin  
initial_password: admin_passwd  
Verifying, please re-enter initial_password:  
ka> examine afs
```


User data for afs

```
key (0) cksum is 2632011835, last cpw: Tue Mar 25 18:06:37 2003
password will never expire.
An unlimited number of unsuccessful authentications is permitted.
entry never expires. Max ticket lifetime 100.00 hours.
last mod on Tue Mar 25 18:06:37 2003 by <none>
permit password reuse
ka> setfields admin -flags admin
ka> examine admin
```

User data for admin (ADMIN)

```
key (0) cksum is 2800900319, last cpw: Tue Mar 25 18:06:50 2003
password will never expire.
An unlimited number of unsuccessful authentications is permitted.
entry never expires. Max ticket lifetime 25.00 hours.
last mod on Tue Mar 25 18:07:30 2003 by <none>
permit password reuse
```

Run the `bos adduser` command, to add the admin user to the `/usr/afs/etc/UserList`.

```
# bos adduser <server name> admin -cell <cell name> -noauth
```

Issue the `bos addkey` command to define the AFS Server encryption key in `/usr/afs/etc/KeyFile`
 Note: If asked for the input key, give the password you entered when creating the `afs` entry with `kas`.

```
# bos addkey <server name> -kvno 0 -cell <cell name> -noauth
input key: afs_passwd
Retype input key: afs_passwd
```

Issue the `bos listkeys` command to verify that the checksum for the new key in the `KeyFile` file is the same as the checksum for the key in the Authentication Database's `afs` entry.

```
# bos listkeys <server name> -cell <cell name> -noauth
key 0 has cksum checksum
```

You can safely ignore any error messages indicating that `bos` failed to get tickets or that authentication failed.

If the keys are different, issue the following commands, making sure that the `afs_passwd` string is the same in each case. The checksum strings reported by the `kas examine` and `bos listkeys` commands must match; if they do not, repeat these instructions until they do, using the `-kvno` argument to increment the key version number each time.

```
# kas -cell <cell name> -noauth
ka> setpassword afs -kvno 1
new_password: afs_passwd
Verifying, please re-enter initial_password: afs_passwd
ka> examine afs
User data for afs
key (1) cksum is checksum ...
```

```
ka> quit
```

```
# bos addkey <server name> -kvno 1 -cell <cell name> -noauth
Input key: afs_passwd
Retype input key: afs_passwd
```

```
# bos listkeys <server name> -cell <cell name> -noauth
key 1 has cksum checksum
```

Issue the `pts createuser` command to create a *Protection Database* entry for the `admin` user
 Note: By default, the Protection Server assigns AFS UID 1 to the `admin` user, because it is the first user entry you are creating. If the local password file (`/etc/passwd` or equivalent) already has an entry for `admin` that assigns a different UID use the `-id` argument to create matching UID's

```
# pts createuser -name admin [-id 42] -cell <cell name> -noauth
```

Issue the `pts adduser` command to make the `admin` user a member of the `system:administrators` group, and the `pts membership` command to verify the new membership.

```
# pts adduser admin system:administrators -cell <cell name> -noauth
# pts membership admin -cell <cell name> -noauth
Groups admin (id: 42) is a member of:
    system:administrators
```

Restart all AFS Server processes

```
# bos restart <server name> -all -cell <cell name> -noauth
```

3.7 Starting the File Server, Volume Server and Salvager

Start the `fs` process, which consists of the File Server, Volume Server and Salvager (`filesaver`, `volserver` and `salvager` processes).

```
# bos create <server name> fs fs /usr/afs/bin/filesaver \
  /usr/afs/bin/volserver \
  /usr/afs/bin/salvager \
  -cell <cell name> -noauth
```

Verify all processes are running

```
# bos status <server name> -long -noauth
Instance kaserver, (type is simple) currently running normally.
  Process last started at Tue Mar 25 18:20:48 2003 (5 proc starts)
  Last exit at Tue Mar 25 18:20:48 2003
  Last error exit at Tue Mar 25 18:06:30 2003, by exiting with code 255
  Command 1 is '/usr/afs/bin/kaserver'
```

```
Instance buserver, (type is simple) currently running normally.
  Process last started at Tue Mar 25 18:20:48 2003 (3 proc starts)
```

```
Last exit at Tue Mar 25 18:20:48 2003
Command 1 is '/usr/afs/bin/buserver'
```

```
Instance ptserver, (type is simple) currently running normally.
Process last started at Tue Mar 25 18:20:48 2003 (3 proc starts)
Last exit at Tue Mar 25 18:20:48 2003
Command 1 is '/usr/afs/bin/ptserver'
```

```
Instance vlserver, (type is simple) currently running normally.
Process last started at Tue Mar 25 18:20:48 2003 (3 proc starts)
Last exit at Tue Mar 25 18:20:48 2003
Command 1 is '/usr/afs/bin/vlserver'
```

```
Instance fs, (type is fs) currently running normally.
Auxiliary status is: file server running.
Process last started at Tue Mar 25 18:23:51 2003 (2 proc starts)
Command 1 is '/usr/afs/bin/fileserver'
Command 2 is '/usr/afs/bin/volserver'
Command 3 is '/usr/afs/bin/salvager'
```

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

- If you are installing the first AFS Server ever in the cell create the first AFS volume, root.afs.
Note : For the partition name argument, substitute the name of one of the machine's AFS Server partitions. By convention these partitions are named /vicepx, where x is in the range of a-z.

```
# vos create <server name> /vicepa root.afs -cell <cell name> -noauth
```

- If there are existing AFS file server machines and volumes in the cell issue the vos sncvldb and vos syncserv commands to synchronize the VLDB (Volume Location Database) with the actual state of volumes on the local machine. This will copy all necessary data to your new server.

```
# /usr/afs/bin/vos sncvldb <server name> -cell <cell name> -verbose -noauth
# /usr/afs/bin/vos syncserv <server name> -cell <cell name> -verbose -noauth
```

3.8 Starting the Server Portion of the Update Server

```
# bos create <server name> upserver simple \
  "/usr/afs/bin/upserver -crypt /usr/afs/etc \
  -clear /usr/afs/bin" -cell <cell name> -noauth
```

3.9 Installing Client Functionality

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

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

3.9.1 Defining Cell Membership for Client Processes

Every AFS client machine has a copy of the `/usr/vice/etc/ThisCell` file on its local disk to define the machine's cell membership for the AFS client programs that run on it. The `ThisCell` file you created in the `/usr/afs/etc` directory is used only by server processes. Among other functions, the `ThisCell` file on a client machine determines the following :

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

Change to the `/usr/vice/etc` directory and remove the symbolic link created when you started the BOS server.

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

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

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

3.9.2 Creating the Client CellServDB File

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

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

Remove the symbolic link created when you started the BOS server

```
# rm CellServDB
# cp /usr/afs/etc/CellServDB ./
```

3.9.3 Configuring the Cache

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

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

3.9.3.1 Configuring a Disk or Memory Cache

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

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

Create the `cacheinfo` file to define the configuration parameters discussed previously. The following instruction shows the standard mount location, `/afs` or `/Network/afs` on Mac OS X, and the standard cache location, `/usr/vice/cache`.

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

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

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

3.9.4 Configuring the Cache Manager

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

See the `/Library/StartupItems/OpenAFS/OpenAFS` file to configure the Cache Manager.

Add the `-nosetime` flag, because this is a file server machine that is also a client. The flag prevents the machine from picking a file server machine in the cell as its source for the correct time, which client machines normally do. File server machines instead use NTPD (as controlled by the `runntp` process) or another protocol to synchronize their clocks.

Add the `-memcache` flag if the machine is to use a memory cache.

3.10 Shutdown and initialization script

The AFS client is now ready, the Cache manager is properly configure. In this section, we'll first shutdown the AFS server then we'll restart it using our Startup item.

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

```
# bos shutdown <server name> -wait -noauth
```

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

```
# ps -aux | grep bossserver
# kill -15 bossserver_PID
```

Then, reboot your server or just issue :

```
# /Library/StartupItems/OpenAFS/OpenAFS
Starting AFS Server processes
Starting afsd
afsd: All AFS daemons started.
```

Note : check that your device is mounted in `/vicepa` before executing the script.

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

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

As a basic test of correct AFS functioning, issue the `klog` command to authenticate as the admin user. Provide the password (`admin_passwd`) you defined before.

```
# klog admin
Password: admin_passwd
```

Issue the `tokens` command to verify that the `klog` command worked correctly

```
# tokens
```

Tokens held by the Cache Manager:

```
User's (AFS ID 42) tokens for afs@<cell name> [Expires Apr  8 16:44]
--End of list--
```

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

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

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

```
# cd / && fs checkvolumes
All volumeID/name mappings checked.
```

3.11 Configuring the Top Level of the AFS filesystem

First you need to set some acl's, so that any user can lookup `/Network/afs`.

```
# fs setacl /Network/afs system:anyuser rl
```

Issue the `vos create` command to create the `root.cell` volume. Then issue the `fs mkmount` command to mount it as a subdirectory of the `/Network/afs` directory, where it serves as the root of your cell's local AFS filesystem. Finally, issue the `fs setacl` command to create an ACL entry for the `system:anyuser` group (or `system:authuser` group). For the partition name argument, substitute the name of one of the machine's AFS server partitions (such as `/vicepa`). For the cellname argument, substitute your cell's fully-qualified Internet domain name (such as `lab.euro.apple.com`).

```
# vos create <server name> <partition> root.cell
Volume 536870915 created on partition <partition> of <server name>
# fs mkmount /Network/afs/<cellname> root.cell
# fs setacl /Network/afs/<cellname> system:anyuser rl
```

Optionaly you can create a link to shortened cell name :

```
# cd /Network/afs
# ln -s lab.euro.apple.com <cell name>
```

Issue the `fs mkmount` command to create a read/write mount point for the `root.cell` volume. By convention, the name of a read/write mount point begins with a period, both to distinguish it from the regular mount point and to make it visible only when the `-a` flag is used on the `ls` command.

```
# fs mkmount /Network/afs/.lab.euro.apple.com root.cell -rw
```

You now have a working AFS filesystem. If you logged into Mac OS X as `root`, open a Finder window and go to `/Network/afs`. Try to drop a file and see what happens.

4 Client installation

Setting up an AFS client is much easier than you could imagine. For this task, I definitely recommend you download the package provided by the OpenAFS Project. Install it, then modify two files located in the `/var/db/openafs/etc` folder. As you probably already guessed, these files are known as `CellServDB` and `ThisCell`. Make them look as the ones residing on your server and reboot or execute OpenAFS startup script. Voila ! Open a `Terminal` and issue the `klog` command :

```
# klog admin
Password :
```

You will get a valid ticket from the AFS server and will be able to access the AFS cell within seconds.

5 References

Your AFS server is far from being optimized or secure at this point. Moreover AFS has tons of a features we didn't discust here. Go browse the documentation to learn more about it ;-)

- OpenAFS Documentation - <http://www.openafs.org/doc/index.htm>
- OpenAFS Download - <http://www.openafs.org/release/latest.html>