--

AudioFile System, Version 2
Release 2
Release Notes and Installation Instructions

ABSTRACT

The AudioFile (AF) System is a network-transparent audio server and client library originally developed for supporting LoFi audio hardware at Digital Equipment Corporation's Cambridge Research Laboratory. The AudioFile distribution contains source code for audio servers, clients, and client library that support several audio devices and platforms.

1. Introduction

The AudioFile System (AF) is a device-independent network-transparent audio server. With AudioFile, multiple audio applications can run simultaneously, sharing access to the actual audio hardware.

Network transparency means that application programs can run on machines scattered throughout the network. Because AF permits applications to be device-independent, applications need not be rewritten to work with new audio hardware. AudioFile does for sound what the X Window System does for text and graphics.

Development of AudioFile began in 1989 at Digital Equipment Corporation's Cambridge Research Laboratory, but it builds on ideas from earlier work. We originally envisioned AudioFile as a server to support all the capabilities of the CRL LoFi (LoFi became the product known as DECaudio) audio hardware running on the DECstation 5000 workstation, but we soon began adding support for a variety of other platforms and audio hardware.

This distribution of AudioFile includes device drivers for several audio devices, server code for a number of platforms, a programming API and library, out-of-the-box core applications, and a number of "contributed" applications. The key difference between the core and contributed applications is that the many of the contrib clients also depend on the Tcl/Tk graphics toolkits distributed by the University of California. (The AudioFile distribution does not include Tcl/Tk, but we tell you where to get it.)

With AudioFile, it is easy to incorporate audio into applications. Simple "play" and "record" applications are included, or you can write your own using the AudioFile API and client library. AudioFile allows applications to generate and process audio in real-time but it also permits more leisurely applications. AudioFile is quite resistant to the vaguaries of scheduling and operating systems that can make handling audio difficult. We have successfully used AudioFile for both trivial applications (audio BIFF) and complex applications (real-time teleconferencing, speech synthesis and recognition.) AudioFile may not be appropriate for all purposes, but we have found it to be a versatile and effective tool.

We believe that AudioFile is highly portable, and that it should be straightforward to add additional support for other systems equipped with audio hardware.

AudioFile is distributed in source form, with a copyright allowing unrestricted use for any purpose except sale (see the Copyright notice included with the kit.)

We would like to encourage other organizations to add support to AudioFile for additional platforms, operating systems, and devices, and to contribute additional applications.

1.1. AudioFile Server

This distribution of AudioFile includes server support for Digital RISC systems running Ultrix, Digital Alpha AXP systems running OSF/1, and Sun Microsystems SPARCstations running SunOS. The servers support audio hardware ranging from the built-in CODEC audio on SPARCstations and Personal DECstations to 48 KHz stereo audio using the DECaudio TURBOchannel module. In addition, server support is present for telephone access devices such as DECaudio.

1.2. AudioFile API

The distribution includes C language bindings to the protocol that make it easy to write distributed audio applications.

1.3. AudioFile Clients

This distribution includes a number of core applications for recording, playback, telephone control, device control, and access control.

1.4. AudioFile "Contrib" Clients

The distribution includes a number of contributed applications including an audio file browser, an FFT spectrogram display, and multicast network audio service.

2. Installation Overview

The remainder of this document contains information and instructions for use in building and installing this release in your local environment. The information in this document assumes that you have already created a local audio hierarchy by following the steps in *README-V2*.

The software in this release is divided into two distributions: one for the *core* server and library software (located in the directory tree *./AF*) and one for device specific software such as device drivers and test programs (located in the directory tree *./devices*.) You will need roughly 6 Megabytes to hold the software kit and an additional 8 Megabytes for building the release.

3. Platform Specific Information

This section provides information about the devices supported by AudioFile and the platforms on which the audio servers and client applications have been built and tested.

3.1. Supported Devices

As mentioned before, AF supports several audio devices on a variety of systems. This table contains a list of the five audio hardware systems that AF has been modified to support.

	Ultrix/RISC	OSF/1 Alpha	SunOS	Server
LoFi (a.k.a. DECaudio)	x	x	-	Alofi
DEC 3000 Audio	n/a	x	-	Aaxp
Personal DECstation Audio	x	n/a	-	Amaxine
J-Video	x	x	-	Ajv
SparcStation	-	-	x	Asparc

It should be possible to add support to AF for SGI Indigo, NeXT, and other Unix workstations with audio hardware.

The software in the *./devices/lofi* directory tree include *setld* device driver kits and library sources that permit programs to interface directly with the LoFi hardware rather than through the AudioFile server. This hierarchy also contains a LoFi diagnostic test program, *hwddt*.

Information on installing and configuring LoFi, Personal DECstation, and Alpha AXP workstation audio device drivers is included below.

4. Building and Installing the Release

Steps to build this release include customizing the build rules according to directory conventions at your local site followed by a few *make* build commands. This section contains the directions for the steps to modify the build rules then build and install this release.

The AudioFile kit has been successfully built and tested on Ultrix/RISC 4.2, 4.2a, and 4.3 systems as well as DEC OSF/1 Alpha V1.2 and SunOS 4.1.1. Please consult the *Errata* section (below) for a description of known build issues with this release.

Almost all *Makefiles* in the core software hierarchy are generated automatically by a utility called *imake*. The program combines machine-independent descriptions (called *Imakefiles*) of targets to be built with machine-dependent sets of parameters. Initial versions of all of the *Makefiles* are included for those sites that cannot use *imake*. For your own applications, the *amkmf* script in *AF/util/scripts* can be used to build a *Makefile* from an *Imakefile* once the core has been built and installed.

This release makes extensive use of a utility called *imake* to generate machine-specific *Makefiles* from machine-independent *Imakefiles*. Another utility, called *makedepend*, is used to generate *Makefile* dependencies for C language files. Sample *Makefiles* are provided, although you are strongly urged to use *imake* and *makedepend* so that your software works across releases.

4.1. Configuring the Release

> *As you follow these instructions, replace each occurrence of <audio_root> with the pathname to the root of the newly created hierarchy and replace each occurrence of <build_root> with the pathname to the root of the corresponding architecture dependent symbolic link tree. If you do not create architecture dependent symbolic link trees, then <audio_root> and <build_root> are interchangeable.*

Site configuration files are located in two places: *./AF/config/site.def* and *./Makefile.rules*. There are two sub-trees in this kit, *./AF* and *./devices*. The build environment in the devices sub-tree does not use the imake utility. As a result, builds in this tree are informed of site configuration rules through the file *./Makefile.rules* and not *./AF/config/site.def* although they both convey the same information generally. (This duplication of information is a known problem.)

The definitions in *./Makefile.rules.example* should be used as a template to create *./Makefile.rules*. The parameter definitions at the top of this file should be customized for your site. The list of these parameters is included below along with a description of its use.

```
% cd <audio_root>
% cp Makefile.rules.example Makefile.rules
```

The default parameters used by *./Makefile.rules* are:

BINDIR	This is the installation directory for binary programs. The template file uses the command, */bin/machine* to create an architecturally specific suffix (typically *alpha* or *mips*.)
MANDIR	This is used to specify the location where manual pages are to be installed. The man pages will be installed in the *man{1-8}* subdirectories of MANDIR.
USRLIBDIR	This is the installation directory for libraries. The template file uses the command, */bin/machine* to create an architecturally specific suffix (typically *alpha* or *mips*.)
INCDIR	Header files will be installed in this location.
LODDIR	DSP56001 *lodfiles* will be installed in this location. Typically, the audio servers for LoFi and J-Video will look here for the image necessary for loading onto the hardware during initialization.
DOCDIR	Miscellaneous PostScript and ASCII text documents are placed in this directory for easy access.
SCRIPTDIR	This is a repository for script files for the LoFi diagnostic program, *hwddt*.

The configuration files for *imake* are located in the directory *./AF/config*. *Makefiles* are created from a template file named *Imake.tmpl*, a machine-specific *.cf* file, and a site-specific *site.def* file. With only a few exceptions, configuration parameters are *cpp* symbols that may be defined on a per-server basis or for all servers in a given site. The template file should *not* be modified.

Site-specific configurations should be described in the file *site.def* using the following syntax:

```
#ifndef BuildParameter
#define BuildParameter site-specific-value
#endif
```

The default parameters used by *./AF/config/site.def* are:

BinDir	Should be the same as BINDIR above.
ManDirectoryRoot	Should be the same as MANDIR above.
UsrLibDir	Should be the same as USRLIBDIR above.
IncRoot	Should be the same as INCDIR above.

As shipped, the configuration files will cause each server supported on a given operating system to be built. If you wish to limit the build to a smaller set of servers, you need to customize either *./AF/config/ultrix.cf.* or *./AF/config/osf.cf* and undefine the unwanted servers.

We ship the configuration file for Ultrix/RISC, *ultrix.cf*, with the DECnet library specified as an extra library. If you do not have DECnet installed, simply comment out this line in the configuration file. It is only necessary to include this library as an extra library, in general, if your X libraries were built with DECnet support included.

```
#define ExtraLibraries -ldnet
```

4.2. SunOS Specific Information

For properly installing and using the SPARC portion of the AudioFile release you must be sure to have installed the supplied audio driver and environment that comes with SunOS 4.1.1 or later. Follow the instructions there for getting that low level support in place. Once this is done you can then follow the directions supplied with AF in both the README-V2 document and this document with the following additional comments:

1. Be sure to edit the Makefile.ini file in the config directory to reflect the BOOTSTRAPCFLAGS appropriate for SPARC. It is currently shipped as commented out.

2. Also be sure that DECnet is not turned on in the .cf files you end up using as this isn't supported for SPARC.

3. You will need to eliminate uses of */bin/machine* in the file *./Makefile.rules*.

4.3. Installation Summary

In general, you will only need to perform build steps from either the *<build_root>*, *<build_root>/devices*, or *<build_root>/AF* directories. The distribution is set-up such that builds will descend through the necessary sub-trees in order to build the predefined targets. There are several *Makefile* targets that are generally implemented. These targets are,

clean	Building this target causes all temporary and derived files to be removed.
depend	Will build the dependency rules using the makedepend utility. This target is generally implemented in the *./AF* area only.
all	*make all* will build the defined targets in the current hierarchy.
install	*make install* will install the defined targets in the current hierarchy.
install.man	*make install.man* will install the defined manual pages in the current hierarchy.

For example, a *make clean* invoked from the *<build_root>* will perform the clean target action in the current directory and all subdirectories defined by the existing Makefiles.

4.4. Detailed Build Steps

To install this release of the AudioFile System, you will need to:

1. Finish reading these Release Notes.

2. If you plan to compile the release on more than one machine and have a distributed file system, you may wish to use the script *./AF/util/scripts/lndir.sh* to create symbolic link trees on each of the target machines. This allows all of the platforms on which you wish to run AF to share a single set of sources. In either case, the phrase *build tree* will be used to refer to the directory tree in which you are compiling (to distinguish it from the *source tree* which contains the actual files).

3. Using *./Makefile.rules.example* as a template, create and customize the parameters in *./Makefile.rules* according to the preferences for your site. This step is described in more detail in the section *Configuring the Release* above.

4. Check the *imake* configuration parameters in *./AF/config/imakemdep.h* and *./AF/config/Imake.tmpl* as well as the site specific definitions in *site.def*. These are the default definitions for *BinDir*, *UsrLibDir*, *ManDirectory*, and *IncRoot*. These are also the most likely definitions that will require changes for a new installation.

   ```
   BinDir
   UsrLibDir
   ManDirectoryRoot
   IncRoot
   ```

   This step is described in more detail in the section *Configuring the Release* above.

5. The first time you build and install the AudioFile release, you will need to create some directories used for the installation steps. The *dirs* target can be used to create these directories automatically.

   ```
   # cd <build_root>
   # make dirs
   ```

   You may have to perform this step as root depending upon whether you have write access to all target parent directories.

6. Make sure you are starting from scratch.

   ```
   % cd <build_root>
   % make clean
   ```

Steps 7a through 7b are necessary if you are building AF servers for LoFi or J-Video audio devices. The configuration files shipped in the AF kit will cause these servers to be built by default on all but SunOS systems.

7a. Build the targets in the *./devices* hierarchy.

   ```
   % cd <build_root>/devices
   % make all
   ```

7b. Install device specific header files and libraries needed to build several of the server targets.

```
# cd <build_root>/devices
# make install >& make.install &
```

You may have to perform this step as root depending upon whether you have write access to all target directories.

8. After building and installing in the *./devices* hierarchy, you are ready to build the core distribution. Look at the *.cf* file (in *./AF/config*) for your system. There may be a line in it that sets a *make* variable named *BootstrapCFlags*. If you don't find such a variable, you can use the following commands to start the build:

```
% cd <build_root>/AF
% make World >& make.world &
```

If you do find the variable, you should append that definition to the command line, using the *make* variable BOOTSTRAPCFLAGS. This is used by *imake* to set particular *cpp* symbols for all compiles.

This step will rebuild all of the *Makefiles* and execute a *make -k all* to compile everything in the core distribution. A World build will take about 10 minutes depending upon system type and file system location.

9. When the *make* is done, check the log file for any problems. There should be no serious errors. See the *Errata* section (below) for a partial list of warning messages that are expected and safe to ignore.

10. If you are satisfied that everything has built correctly and you already have existing audio device driver support installed, you may test the various programs (server, atest, atime, aevents, aplay etc.) to verify the operation of the server and device driver. See *Testing the Release* (below) for further information.

11. If you have previously installed an AF release, you might want to make backup copies of your old AF header files, binaries, libraries, etc.

12. Install the AF targets by,

```
# cd <build_root>/AF
# make install >& make.install
```

You will either have to do this as root, or have write access to the appropriate directories.

13. If you would like to install the manual pages, type the following at the top of the build tree:

```
# cd <build_root>
# make install.man
```

If you are installing AF for the first time, you may also need to do some of the steps listed below.

14. Make sure that all AF users have the installed binary directory, *BINDIR* and *BinDir*, in their search paths. Also, you may wish to append the installed man page directory to your MANPATH environment variable.

15. Install device drivers as necessary.

16. Give it a try! See the section on *Testing the Release* (below.)

Release 2 of Version 2 of the AudioFile System should now be ready to use.

5. Testing the Release

Start the appropriate AudioFile server according to the one of the following manual pages: *Alofi.1, Aaxp.1, Amaxine.1, Ajv.1, Asparc.1*. Next, try executing some of the AudioFile clients.

aevents Logs all supported events to stdout. (For example, if you have a LoFi, start aevents and call the phone number for the phone line that is connected to the module.)

ahost Allows you to control access to your the AudioFile server by other hosts.

ahs Hookswitch control client for LoFi.

alsatoms This client lists the interned atoms.

apass An example client that communicates with two AF servers and manages the time jitter in the audio devices. This client reads samples from the input audio server and writes them to the output server with a specified delay between input and output.

aphone A simple phone dialing application for LoFi.

aplay A play sound file client. There are a few example sound clips in *<audio_root>/sound_files*.

aprop Displays properties in an AudioFile server.

arecord Records audio samples from an audio device.

aset Sets and queries various audio devices settings in a server.

6. Contrib

We include some additional AudioFile clients in the *./AF/contrib/clients* hierarchy. Clients in this hierarchy may depend upon some external piece of software. For example, some clients use the Tcl/Tk X toolkit developed by John Ousterhout at Berkeley.

The Tcl/Tk distribution is available via Internet anonymous FTP from:

sprite.berkeley.edu:/tcl

The V3.1 distribution of Tcl/Tk has been tested on Ultrix/RISC and Alpha/OSF. We recommend that you retrieve and install this software to gain access to several other example AudioFile client applications.

The following are CRL provided client applications that are located in the *./AF/contrib/clients* hierarchy. These clients are not built by default.

abrowse A Tk based sound file directory browser application.

adial A Tk based phone dialer for LoFI.

afft Real-time FFT displayer. Based on Tk.

afxctl Controls the inputs and outputs and gain of the audio devices in the server. Also logs AF events.

xpow Displays the power level of the recorded audio on the specified audio device.

atest Sends NoOp protocol requests to the server and logs that fact to stdout.

atime Requests an audio device time and logs the time to stdout.

axset A Tk based audio device control panel. Similar functions as supported by afxctl.

radio Contains a program to multi-cast an audio source and the companion program to listen to the multi-cast source.

xplay A Motif based audio sound file browser.

6.1. Building Contrib

In order to build these clients you will need to type the following commands,

```
% cd <build_root>/AF/contrib/clients
% make Makefiles clean depend all
```

and as root,

```
# cd <build_root>/AF/contrib/clients
# make install install.man
```

7. Device Driver Information

We provide device driver kits for LoFi, Personal DECstation base board audio, and DEC OSF/1 Alpha AXP workstation base board audio devices. These kits are in *setld* format.

Information on how to install the device driver for LoFi (a.k.a. DECaudio) can be found in:

Ultrix/RISC

<audio_root>/devices/lofi/driver/mips.src/lofi.4

OSF/1 Alpha

<audio_root>/devices/lofi/driver/alpha.src/lofi.7
<audio_root>/devices/lofi/driver/alpha.src/README

The audio server for the Personal DECstation audio device is dependent upon the */dev/bba* device driver. Information on how to install the device driver for the Personal DECstation audio can be found in:

Ultrix/RISC
<audio_root>/devices/maxine/driver/mips.src/bba.4

The audio server for the Alpha AXP workstations are dependent upon the bba audio device driver. Information on how to install the device driver for the base board audio can be found in:

OSF/1 Alpha <audio_root>/devices/axp/driver/README

The *bba* driver for OSF/1 Alpha workstations should only be installed on V1.2 (the first official external release) or internal version X1.2 (BL11) only.

8. DSP Port Devices

The LoFi audio device has an external *DSP* port compatible with the DSP port on NeXT computers. We have done some minimal testing of our LoFi server with the following DSP port devices:

Ariel Pro-Port (Model 656)

Ariel Digital Microphone (DM-N)

Applied Speech Technologies AST A/D16

9. Errata for this Release

Protocol Change **Note to users of previous releases of AF. We have fixed some elements of the protocol that were known to be broken. It was necessary to do this prior to this more general release of AF. This protocol change affects the *AFRecordSamples* and *AFPlaySamples* interfaces. The audio context attributes have been augmented in a upwards compatible fashion. Please modify your existing AF client applications according to the new interfaces as documented in the appropriate man pages.**

OSF/1 If you are building on OSF/1 Alpha BL10 you will need to comment out the lines containing "#include <c_asm.h>" in If you are building on OSF/1 Alpha BL11 (X1.2), you need to make sure the "#include <c_asm.h>" lines exist in the files mentioned above -- however you will see warning messages during the build containing the string, *"asm(mb) ... not a legal asm string"*. You may safely ignore these warning messages.

Device No. Be aware that some client programs default to audio device 1 and that AudioFile servers for some audio devices with a single CODEC only present audio device 0. As a result, you may need to specify *-device 0* or *-d 0* to some AF clients.

Multiple Dev. The server does not yet support more than one physical device (e.g., LoFi) in the same machine. Much of the structure to support this is there, but some work must still be done. It is possible to run multiple servers on a single host, however, allowing clients to access multiple LoFi devices on a given machine.

Aaxp The CODEC device gain control (as supported by afxctl) is not implemented in the Aaxp server in this release. We expect this to be done shortly, however.

Interop.	We have not fully tested interoperability between Sparc and non-Sparc systems. If you are able to exercise interoperability between these systems and find problems, please let us know.
setld	If you need to install a device driver from this kit, be sure to invoke the *setld* command from the source tree and not the corresponding symbolic link tree. *setld* will not succeed if you invoke it from the link tree.
DSP56K	We do not provide the DSP tools in this distribution. If you wish to do development of DSP56001 code on LoFi, contact Motorola or any other manufacturer of code development software for the DSP56001 processor. The source files for the DSP56001 used by the LoFi and J-Video server are included. The audio servers expect to read the *lod* file format images.

10. Future Work

Right now, the various audio channels supported by a server are assigned integer device numbers in approximately random order. There should be a more symbolic way to talk about "the local loudspeaker" or "the telephone".

Right now, the AudioFile clients which deal with disk files only know about uninterpreted byte streams. There is no support for the various popular sound file formats which automatically specify their datatypes. This should be fixed.

We would like to add capability to server to play and record compressed datatypes. There is a possibility that the current play and record protocol interfaces are not adequate for really complex compressed formats which might include variable rate codes and unusual blocksizes.

Right now we have pushed all worries about sample rate conversion onto the clients. That is probably the right place for it, but we could possibly provide some client library support for it.

Audio devices have a bitfield of formats they support. The representation should be an ordered list rather than a bitfield so that a device preference for one format over another can be communicated.

It should be possible to reconfigure the DSP port interface on DECaudio dynamically, rather than only at server initialization time.

We realize that additional device support would be good: The Silicon Graphics Indigo, the NeXT, and other machines could easily support AudioFile, but we haven't done the necessary work.

Right now we have separate servers for each audio device on a particular machine. A single server should be able to support all configured devices. A single server should be able to support multiple devices of the same kind as well. This is not a design issue, just implementation.

On Alpha/OSF we should be able to dynamically link to the POSIX real-time library and use it to good advantage.

We hope to have a CRL technical report describing AudioFile in more detail available in the near future.

11. Acknowledgements

If you retrieved and installed the companion AF kit, *AF2R2-other.tar.Z*, you will have a few sample Hi-Fi sound files in *<audio_root>/sound_files/xan**. The audio examples are excerpts from the piece *Xanadu* by Joseph T. Kung. It was realized using Barry Vercoe's Csound during a computer music course run by Prof. Vercoe at the MIT Media Lab's Music and Cognition Group in December 1988. The companion kit contains 16, 44.1, and 48 KHz versions of *Xanadu* for use with the LoFi device's Hi-Fi output or DSP port device.

Ricky and Larry Palmer ported the server to a SparcStation and provided the source code changes back to us for inclusion in this distribution.

12. Mailing Lists

We have established a mailing list, *af@crl.dec.com*, for general discussions about AudioFile. To be added to or removed from this mailing list, please send mail to,

    af-request@crl.dec.com

You may report problems with the AudioFile software by sending mail to *af-bugs@crl.dec.com*. Please be specific as possible when describing the problem and include source code for a small test case if possible.

13. Summary

We hope you enjoy this AudioFile release.