AF Release Notes and Installation Instructions

Please read this thoroughly before installing AF for the first time.

This is the table of contents for the release notes and installation instructions for AF3R1.

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 AF distribution contains source code for audio servers, clients, and client library that support several audio devices and platforms.

Introduction

The AF System 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. AF does for sound what the X Window System does for text and graphics.

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

This distribution of AF 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 Tcl/Tk FTP site is located at ftp.cs.berkeley.edu:ucb/tcl.

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

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

AF is distributed in source form, with a copyright allowing unrestricted use similar to the terms in the X Window System copyright (see the Copyright notice included with the kit.)

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

Further information on AF is available in our USENIX paper and in a CRL Technical Report.

Servers

This distribution of AF includes server support for Digital Alpha AXP systems running OSF/1, Digital RISC systems running Ultrix, SGI Indigo systems running IRX 4.0.5, and Sun Microsystems SPARCstations running SunOS 4.1.3 or Solaris. 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 or the SPARCstation DBRI interface. In addition, server support is present for telephone access devices such as DECaudio.

This lists all known audio hardware systems that AF has been modified to support.

OSF/1 for Alpha

- *Aaxp* manages the base−board audio hardware on Alpha workstations.
- *Amsb* manages the Microsoft Sound Board audio hardware on an Alpha PC.
- *Aj300* manages the J300 Sound and Motion TURBOchannel audio hardware.
- *Alofi* manages the LoFi/DECaudio TURBOchannel audio hardware.
- *Ajv* manages the J−Video prototype TURBOchannel audio hardware.

Ultrix/RISC

- *Alofi* manages the LoFi/DECaudio TURBOchannel audio hardware.
- *Amaxine* manages the base−board audio hardware on Personal DECstations.
- *Ajv* manages the J−Video prototype TURBOchannel audio hardware.

SGI Irix

- *Asgi* manages the audio hardware on Indigo.

SunOS and Solaris

- *Asparc* manages the base−board audio hardware on Sparcstations.
- *Asparc10* manages the audio hardware on a Sparcstation 10.

It should be possible to add support to AF for other Unix workstations with audio hardware.

API

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

Clients

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

"Contrib" Clients

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

What is new in this Release?

- The core distribution uses a less restrictive copyright similar to the terms in the MIT X Window System copyright.
- **To build this release of AF, you must use a compiler that supports function prototypes**. AF3R1 no longer includes the `NeedFunctionPrototypes` macro to conditionally remove prototypes during compilation.
- Includes all known bug fixes (thanks to all of you who reported problems and/or provided fixes) including several that were not provided as patches to AF2R2.
- New protocol revision.
- Uses our official AF TCP port number.
- Includes a few new clients, including tkPlay, a Tk based simple GUI to aplay which is a good candidate for your Mosaic showaudio command.
- Includes automatic datatype conversion via an encoding type attribute on a per AC basis. Supports most simple datatypes as well as some compressed formats (IMA, GSM, etc.) See the aplay source as an example of a client using this feature.
- Includes a sampling rate and sample format conversion client and supporting library code. See aconvert in the contrib hierarchy.
- Includes a few higher level interfaces in AFlib supporting file I/O using Sox (Tektronix).
- Hypertext documentation suitable for viewing with Mosaic.
- Server support for the Microsoft Sound Board on the Alpha PC.
- Server support for the J300 Sound and Motion board on Alpha workstations.
- Server support for SGI Indigo courtesy of Guido van Rossum, CWI.
- Server extension code and Sun Solaris port courtesy of Wes Whitnah, Tektronix.
- This kit defaults to installing into the /usr/local hierarchy. Should enable most sites to simply unpack the kit and perform the *make World* step.
- Sparc 10 DDA contributed by Yechezkal−Shimon Gutfreund, GTE Laboratories.
- No doubt I have left things off this list...

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 unpacking the source kit and following the steps in *README−V3*.

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 drivers (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.

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 AF kit has been successfully built and tested on Ultrix/RISC 4.2, 4.2a, and 4.3 systems, DEC OSF/1 Alpha V1.2, V1.3, and 2.0 systems, as well as IRIX 4.0.5 and SunOS 4.1.3 systems. 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.

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. 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.

Configuring the AF 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.**

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
: This is the installation directory for binary programs.

ManDirectoryRoot
: 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.

IncRoot
: Header files will be installed in this location.

The default locations in site.def for AF3 have been changed and now use /usr/local/ which tends to be the most common installation location. We have included site.def.crl if you wish to use the same default installation target directories that were used in AF2R2.

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 the appropriate platform configuration file,

- *./AF/config/ultrix.cf*
- *./AF/config/osf.cf*
- *./AF/config/sun.cf*
- *./AF/config/sgi.cf*

and undefine the unwanted servers.

Building for Debugging

If you wish to build the release with debugging options turned on, you will need to modify the appropriate *.cf* file for your system. You might choose to define the following,

#define ServerCDebugFlags −g

#define DefaultCDebugFlags −g

Alternatively, you can define *CDEBUGFLAGS* in any individual Imakefile to enable debugging for specific pieces of the AF distribution.

Ultrix/RISC Specific Information

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
```

If you plan to use Ultrix 4.3a or later, you should include *−oldc* as a switch for *DefaultCCOptions* definition in the *ultrix.cf* file. AF does not build properly when using the newest Ultrix/RISC compiler shipped with 4.3a and later.

SunOS and Solaris Specific Information

For properly installing and using the SPARC portion of the AF release you must be sure to have installed the supplied audio driver and environment that comes with SunOS 4.1.1 or later, and with Solaris. 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−V3* document and this document with the following additional comments:

1. Install */usr/demo/SOUND* supplied with SunOS 4.1.1 or later, and with Solaris 2.3 .

2. You will need to use a compiler that supports function prototypes, such as the SunPro compiler or gcc. GNU’s gcc version 2.4.5 or later is required for SunOS, and 2.5.8 or later is required for Solaris.

You can get a gcc kit from gatekeeper.dec.com, but consider other GNU mirror sites that might be "closer". Original sources can be found at prep.ai.mit.edu:/pub/gnu.

3. Edit *sun.cf* . If you are using gcc then be sure the *HasGcc* line is set to *YES*, otherwise set it to *NO*. If set to *NO* the configuration file will assume you are using the SunPro compiler. To use the gcc compiler the line should look like this:

   ```
   #define HasGcc YES
   ```

   To compile a server for the Sparc−10 DBRI interface add the line:

   ```
   #define Asparc10Server Asparc10
   ```

   immediately after this line:

   ```
   #define AsparcServer Asparc
   ```

   This will cause both Sparc servers to be built. Asparc will run on Sparc−10's but the server will be limited to the Sparc−2's CODEC capabilities.

   See the ./AF/server/dda/sparc10/README file for more information on the Sparc−10 server.

4. Edit *Makefile.ini* . Comment out the first *BOOTSTRAPCFLAGS* line by placing a '#' as the first character of the line.

   For SunOS, remove the '#' from the second *BOOTSTRAPCFLAGS* line and delete everything past the '=' sign, so that the line looks like this:

   ```
   BOOTSTRAPCFLAGS =
   ```

   For Solaris, remove the '#' from the second *BOOTSTRAPCFLAGS* line and delete everything past the '=' sign, then add '−DSVR4', so that the line looks like this:

   ```
   BOOTSTRAPCFLAGS = −DSVR4
   ```

   Edit the *LOCC* line to use the correct compiler. Most sites can leave this line alone, but Solaris users will have to modify it. For gcc users the line would look like this:

   ```
   LOCC = COMP_HOST_ROOT=/ COMP_TARGET_ROOT=/ gcc −O
   ```

5. Edit *site.def* . Verify that the paths and defaults are appropriate for your site.

Installation Summary

There are several *Makefile* targets that are generally implemented. These targets are,

Makefile
: This causes the Makefile in the current directory to be rebuilt from its Imakefile source.

Makefiles
: This causes all Makefile targets in the hierarchy to be rebuilt from their Imakefile sources.

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* will perform the clean target action in the current directory and all subdirectories defined by the existing Makefiles.

Detailed Build Steps

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

1. Finish reading these Release Notes, then return to step 2.

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. Install device drivers as necessary. Pointers to the device specific driver setld kits are contained in the release notes (below.)

4. Check *site.def* and modify according to your local site preferences. This file contain 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.)

    You might also wish to check the *imake* configuration parameters in *./AF/config/imakemdep.h* and *./AF/config/Imake.tmpl* In general, you will not need to modify these two files.

5. Extension code is included in this version of AF. A sample extension can be configured by adding the following definition to your site.def or system configuration (*.cf) file.

    ```
    #define ExtensionDefines -DAFTIME
    ```

6. Look at the platform configuration file for your system. There may be a line in it that sets a *make* variable named *BootstrapCFlags*. *Currently, the configuration files osf.cf, ultrix.cf, sun.cf, and sgi.cf do not set this variable.* 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 make 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.

7. When the *make* is done, check the log file for any problems. There should be no errors.

8. 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.

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

10. 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.

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

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

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

Release 3 of Version 1 of the AF System should now be ready to use.

Testing the Release

Start the appropriate AF server in *<build_root>/AF/server/* according to the one of the following manual pages:

- *Alofi.1*,
- *Aaxp.1*,
- *Amaxine.1*,
- *Aj300.1*,
- *Ajv.1*,
- *Asparc.1*,
- *Asparc10.1*,
- *Asgi.1*,
- *Amsb.1*

Next, try executing some of the AF 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 AF 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 AF server.

arecord
: Records audio samples from an audio device.

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

Contrib

We include some additional AF 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 *ftp.cs.berkeley.edu:/ucb/tcl*

The 7.2/3.5 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 AF 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.

aconvert
: A sample rate and sample format conversion utility. Also, see the libaconvert.a library in the contrib hierarchy.

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.

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.

tkPlay
: A Tk based GUI to aplay. Preferred usage is as the "showaudio" equivalent for browsing the WWW.

xplay
: A Motif based audio sound file browser.

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

Building Contrib

The simplest way to build the contrib tree is to type the following commands,

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

and as root,

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

This sequence will build the library and clients in the contrib hierarchy before installing all appropiate targets.

If you do not have some of the external dependencies installed, you may have to be more selective about how you build the contrib clients.

Device Driver Information

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

Alpha Base−Board Audio Device (/dev/streams/bba)

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 in this AF kit is for OSF/1 Alpha workstations running V1.2 (the first official external release) only. If you are running V1.3 or later, you merely need to add the appropriate bba configuration line to you systems config file. The bba driver is already included in the base system on V1.3. **Do not install the driver from the AF kit if your are running OSF/1 V1.3 or later.**

Alpha (Jensen) Microsoft Sound Card (/dev/msb)

The audio server for the Microsoft Sound Card (Alpha PCs) are dependent upon the msb audio device driver. Information on how to install the device driver for the msb audio can be found in:

OSF/1 Alpha
: <audio_root>/devices/axp150/msb/README.msb

If you have already installed the msb driver shipped with the Beta test version of AF3R1 then you must deinstall the field test driver and install the version shipped with this kit. Otherwise you may notice system hangs.

LoFi (DECaudio) TURBOchannel Audio Device (/dev/lofi)

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/lofi.4.man

OSF/1 Alpha
: <audio_root>/devices/lofi/driver/README.osf

: <audio_root>/devices/lofi/driver/lofi.7.man

Personal DECstation Base−Board Audio Device (/dev/bba)

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

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

Errata for this Release

Protocol Change
: Now at AF Version 3 Release 1.

Multiple Devices

A single 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, thus allowing clients to access multiple LoFi devices on a given machine.

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 (if it exists). *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.

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".

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

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.

Acknowledgements

If you retrieved and installed the companion AF kit, *AF2R2-other.tar*, 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*.

Guido van Rossum contributed device dependent code for SGI Indigo.

Wes Whitnah contributed extension code, AFsox library, and *several* bug fixes.

Mailing Lists

We have established a mailing list, *af@crl.dec.com*, for general discussions about AF. 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 AF 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. Also, state your current patch revision (as found in *./patchlevel.h*).

Product Information

You can obtain information on Digital’s products through the Digital Information W3 server.

Summary

We hope you enjoy this AF release.

---

af−bugs@crl.dec.com