Kent Lindquist White Paper

Number 2001-004

AEIC Antelope upgrade from 4.3u to 4.4u

Kent Lindquist

December 3, 2001

AEIC Alaska Earthquake Information Center

Geophysical Institute, UAF

Upgrading Antelope from 4.3u to 4.4u

The upgrade from Antelope 4.3u to 4.4u at the AEIC is a multi-step procedure. The goal is to bring the more powerful new Antelope release to bear on AEIC operational tasks, without at any time disrupting lab operations. This requires a sequence of carefully considered decisions, often a necessary logical ordering of steps. Because of changes throughout each year to the AEIC software base, plus changes in Antelope features, this transition is unique each time it is done. The purpose of this white paper is to track one specific upgrade to give a picture of the types of details involved.

We will start with an Antelope pre-release dated October 9, 2001. This is being installed to bring us to the cutting edge[1]. For the most part we will be able to perform the installation without interfering with lab operations. The new Antelope will install to /opt/antelope/4.4u, thus will exist in parallel to the /opt/antelope/4.3u software being used by the lab. We will also replace /opt/antelope/rtdemo_sc and /opt/antelope/dbdemo, however there are no operational tasks that rely on those directories. The tcl/tk version in Antelope has been upgraded to tcltk8.3, so the new /opt/antelope/tcltk8.3 directory will peacefully coexist with /opt/antelope/tcl7.4tk4.0 and /opt/antelope/tcltk8.0. The one difficulty is /opt/antelope/perl, which was Perl version 5.005 in the Antelope 4.3u distribution and will be Perl 5.6 in the Antelope 4.4 distribution. Since the /opt/antelope/perl/bin/perl is linked to /usr/bin/perl, we will have to make the Perl switch after hours and keep the disruption short. The Antelope 4.4u Perl 5.6 is designed to be completely backwards compatible with Antelope 4.3u and Perl 5.5 [D. Quinlan, *pers. comm.* 2001], thus this should be a simple replacement. As a precautionary measure (the full value of which will be seen below), I moved /opt/antelope/perl to /opt/antelope/perl_4.3u rather than allowing the Antelope install script to overwrite /opt/antelope/perl. Hopefully by performing this move, the inodes for the directory and executables remain the same, allowing any open Perl processes to continue running. There still may have been a small [10-20 minute] disruption in Perl service. Given that strategy, I ran the install script for the Antelope 4.4u pre-release.

The currently running Perl in Antelope 4.3u has been modified. These modifications need to be propagated into the Antelope 4.4u release. Specifically, the dbrecenteqs implementation which is generating AEIC near-real-time earthquake plots on the web uses the PerlMagick objects of the freeware ImageMagick image-handling utilities [the PerlMagick objects allow dbrecenteqs to

1. Normally we prefer that AEIC operations not be on the cutting edge. Thus, even though the 4.4u pre-release is being installed, we will delay switching the lab to it. 'Cutting edge' is often used in appealing terms. However, the prototype nature of these pre-releases virtually guarantees that the user will get cut by at least some part of the new distribution [putting a point on the analogy]. Cutting edge is only worth it if the incremental gains over the last stable release outweigh the pain of getting cut. In this case, our need to get AEIC catalog generation back online requires that we finish getting regionalized travel-time models implemented for Alaska. That, in turn, is currently blocked by version skew issues between Antelope 4.3u, the Antelope contrib area, and the Antelope development source-tree from which the 4.4u pre-release was extracted. Thus in this case, installing cutting-edge technology allows us to continue towards a primary AEIC goal without having to wait months between steps.

modify gif and jpg images by painting earthquakes on them]. I previously built this Perl module into our installed Antelope 4.3u distribution[1]. To repeat this, we need to go to

```
/opt/free/src/IMAGEMAGICK/ImageMagick-5.3.0
```

and re-run the installation:

```
./configure --enable-lzw --enable-shared --prefix=/opt/free --with-perl=/opt/antelope/perl/bin/perl CC=cc CFLAGS="-I/opt/sfw/include -I/opt/free/include -xCC" LDFLAGS="-R/opt/sfw/lib -L/opt/sfw/lib -R/opt/free/lib -L/opt/free/lib" CPPFLAGS="-I/opt/sfw/include -I/opt/free/include -xCC" --without-magick-plus-plus

make

make install
```

After installing the new distribution, I made sure that the PerlMagick objects, dbrecenteqs, and the AEIC alarm-response scripts still work properly.

Since the Antelope license manager changes with each release, I had to request a new license from BRTT. This was shipped, so I have created a file license.pf and put it in /opt/antelope/4.4u/data/pf.

Now we need to update local source-code. First, in /usr/local/aeic/4.3u/src I ran

```
nordic% cvs rtag last_43u src
```

to freeze AEIC software development under Antelope 4.3u. Now check out for development under Antelope 4.4u:

```
nordic% cd /usr/local/aeic/
nordic% mkdir 4.4u
nordic% cd 4.4u
nordic% setenv CVSROOT /usr/local/aeic/cvs
nordic% cvs checkout src
```

Despite the apparent continuity of this document, a month has passed between that last step (Oct. 22, 2001) and the next (Dec. 2, 2001), thus I will update the checkout:

```
nordic% cd /usr/local/aeic/4.4u/src/
nordic% cvs update -d -P .
```

1. I asked Dan Quinlan of BRTT about the possibility of building this Perl module into the distributed Antelope Perl. However, that is not realistic at this time.

Next I will modify my own user environment, which I keep separate from the lab environment for just such purposes, to run Antelope 4.4u instead of 4.3u. This amounts to changing my setup files:

```
setenv ANTELOPE /opt/antelope/4.4u
setenv SCHEMA_DIR $ANTELOPE/data/schemas:/usr/local/aeic/$ANTELOPE:t/data/schemas
```

Note that data/schemas2 has finally changed back to data/schemas.

First we will handle iceworm recompilation. This needs to be done three times, once for each compiled copy of iceworm[1]. I will begin with the development system, hoping to find and resolve any inconsistencies without bringing down lab acquisition and processing. The point of handling iceworm compilation before /usr/local/aeic compilation is that the latter presents more difficulties in creating a safe testbed.

The immediate discovery is that Antelopemake has changed drastically, with a $(DEST)/$(SUBDIR) mechanism that may substantially supercede the $(PACKAGE) strategy I had used previously (with antelopemake local-mods) to handle the /opt/iceworm and /usr/local/aeic source trees. Paralleling the Antelopemake mechanism, we will allow parallel source-code trees to be managed by an expansion of the Antelope makefile mechanism, by defining one new environment variable per tree. For Iceworm, we will have ICEWORMMAKE.

Move /opt/iceworm/include/packagemake to /opt/iceworm/include/icewormmake.

Move the ‘packagemake’ functions of the locally modified /opt/antelope/4.3u/include/antelopemake into the new icewormmake[2]:

include $(ANTELOPEMAKE)
DEST=/opt/iceworm
CFLAGS, COFLAGS, FFLAGS, and LDFLAGS additions
change from LDFLAGS specification to LDPATH/LDRUN for 4.4u Antelope

Within each makefile in the /opt/iceworm/src tree, change

```
PACKAGE=/opt/iceworm
include $(ANTELOPEMAKE)
```

to

1. Each of the operational, backup, and development Iceworm systems get their own copy of the iceworm code, to make the machines independent of each other and robust against network problems.
2. It is critical that DEST occur after the include $(ANTELOPEMAKE), otherwise the macro is overridden by the default entry in ANTELOPEMAKE, opposite of what is intended. There is an alternative to the definition of an ICEWORMMAKE environment variable: DEST additions could be automatically sought out and handled by antelopemake, requiring simply that every makefile in the source-code tree have a correct DEST specification. However, it is likely that one of those would get left out or reversed in order, thus formalizing this with a separate environment variable and master makefile is more robust.

```
    include $(ICEWORMMAKE)
```

Also we add

```
setenv ICEWORMMAKE /opt/iceworm/include/icewormmake
```

I had to move /opt/iceworm/src/* to /opt/iceworm/src/4.4u/* to avoid cross-directory installation complaints of the new deposit(1) command. I sent feedback to Dan Quinlan about whether the interaction between user-specifiable DEST and deposit(1) installation checking needs to be improved. At the moment this change to 4.4u/src is not so bad, since it makes explicit the dependence on the source tree of the 4.4u Antelope distribution.

The new antelope moves all static libraries to $(DEST)/static. /opt/iceworm/static cannot be created due to the way automount is currently setup. We could change that, however it is probably more easy and appropriate just to make libunr.a dynamic-only, since static libraries are being phased out anyway. Change the libunr makefile to reflect that.

The next small problem that needs to be resolved is apparently a beta-test problem. Nothing will compile without libbanner, which is apparently a static-library only. All the static libraries were omitted from the beta-release CD. After fixing this[1], we can execute

```
nordic% cd /opt/iceworm/src/4.4u/
nordic% make Include
nordic% make install
```

The next step is to update all tcl and wish-based scripts. We find these with the command

```
nordic% cd /opt/iceworm/bin/
nordic% file *
```

The only file to fix is logwatch. This required removal of redundant startup lines from logwatch.xwish; inclusion of “package require Tclx” to support the int command; and switching from the Unix ‘echo’ to the tcl ‘puts’ command. This program has a number of other problems best addressed by erasing it.

After all these changes, we do

```
nordic% pwd
/opt/iceworm/src/4.4u
nordic% cvs commit .
```

1. My initial fix has been superseded by Dan Quinlan’s advice to omit libbanner entirely from Antelopemake. Unfortunately that still didn’t work, leaving us with the temporary fix until the official 4.4u release comes out.

We cannot repeat these changes on the ice and earlybird iceworm distributions until the Antelope distributions (and perl) are upgraded there:

```
    earlybird% certainly_remove 4.2u/
    # rmdir 4.2u
    # mv perl/ perl_4.3u
    earlybird% cp -r /net/nordic/opt/antelope/{4.4u,perl,tcltk8.3} .
```

I also had to clean up a couple soft-links after this, including:

```
    earlybird% pwd
    /opt/antelope/tcltk8.3/lib
    earlybird% ln -s iwidgets3.0.1 iwidgets
```

also

```
    earlybird% certainly_remove schemas2
    earlybird% ln -s schemas schemas2
```

We repeat these steps on ice, with the one small improvement

```
    # cd /opt
    # chown kent:bdogs antelope
```

In both of these cases, we need to make sure /usr/bin/perl is the correct version. Earlybird was OK because of a soft-link, however ice had the soft-link moved so I fixed it.

Moving back to /usr/local/aeic, we create the supporting directories

```
nordic% cd /usr/local/aeic/4.4u/
nordic% mkdir bin data include lib man
```

We create a file

```
    nordic% cat /usr/local/aeic/4.4u/include/aeicmake
    include $(ANTELOPEMAKE)

    DEST=/usr/local/aeic/4.4u

    CFLAGS += -I$(DEST)/include
    COFLAGS += -I$(DEST)/include
    FFLAGS += -I$(DEST)/include
    LDPATH += -L$(DEST)/lib
    LDRUN += -R$(DEST)/lib

    nordic%
```

and add to my environment

```
setenv AEICMAKE /usr/local/aeic/4.4u/include/aeicmake
```

Next we go through all the makefiles in this source-code tree with

```
nordic% vi ‘find . -name Makefile -print‘
```
and eliminate all lines in makefiles that say

```
PACKAGE=/usr/local/aeic/RELEASE
```

Next we change all occurrences of ANTELOPEMAKE to AEICMAKE:

```
nordic% perl -p -i -e ’s/ANTELOPEMAKE/AEICMAKE/’ ‘find . -name Makefile -print‘
```

Finally we have to account for the special cases:

```
vi ./data/tables/genloc/db/Makefile ./data/tables/genloc/tt1dcvl/Makefile ./data/tables/genloc/
ttlvz/Makefile
```

to enter the modification

```
# suspend DEST assignment to put files directly in $ANTELOPE
include $(ANTELOPEMAKE)
```

Similarly, in src/lib/libuser

```
# install libuser directly in $ANTELOPE
include $(ANTELOPEMAKE)
```

Finally, we need to change strategies on the Makefile-embedded autoload path assignment for fktools and wormwatch, which use

```
wormwatch.sh : wormwatch.sh.ap
      perl -p -e "s@auto_path\s+DIR@auto_path $(PACKAGE_PATH)@g" < worm-
watch.sh.ap > $@
```

PACKAGE_PATH will now always be identical to DEST.

After a small step to add a makefile to the new aeic_rtsys_blueprint so compilation can continue, we do a

```
nordic% cd /usr/local/aeic/4.4u/src
nordic% make Include
```

Unfortunately, a further make install is blocked due to dependencies on the Antelope contrib distribution, which must be compiled first. (By all rights that should have been handled before copying Antelope to earlybird and ice.) The issue here is that changes were made to the contrib distribution after the production of the beta release, that affect code in /usr/local/aeic. Check-out and compile the contrib distribution:

```
nordic% cd /opt/antelope/4.4u/
nordic% mkdir contrib
nordic% cd contrib
nordic% setenv CVSROOT aesn.geology.indiana.edu:/opt/antelope/cvs
nordic% cvs checkout -d src contrib
nordic% cd src
nordic% make Include
nordic% make install
```

This turned up a number of problems in the CVS repository. I cleaned up the ones that were my fault.

Repeat the compilation on earlybird and ice:

```
nordic% rsh earlybird
earlybird% cd /opt/antelope/4.4u/
earlybird% cp -r /net/nordic/opt/antelope/4.4u/contrib .
earlybird% cd contrib/src/
earlybird% make Include
earlybird% make install
```

Again, this could have been handled differently by completing the contrib assembly before transferring to the operational machines.

Returning to /usr/local/aeic, we are now able to remake the distribution without version skew problems against the contrib code. A number of changes were necessary to aeic_dbap to bring it up to date with the new version of Tcl/Tk. I also fixed a number of other problems in this source code tree, which only a fresh CVS checkout (such as the one performed) was able to turn up.

Now we need to upgrade all TCL/Tk programs to the new version of the interpreters, as well as the new dynamic loading of interpreter extensions as adopted by BRTT. Once again, /usr/bin/file on /usr/local/aeic/4.4u/bin shows the scripts that need to be modified and checked. Note that I had to test as many functions as each script as possible, since they are possible of hiding until run-time execution. These scripts are:

```
aeic_release_distributor:      executable shell script
analysis_control:              executable shell script
autopick_change:               executable shell script
calldown_notification_tool: executable shell script
circuit_check:                 executable shell script
```

discno_change:	executable shell script
felt_report_tool:	executable shell script
fktools:	executable shell script
gui_dbicepick:	executable shell script
icetools:	executable shell script
local_cdrom_wfdisc:	executable shell script
mapset:	executable shell script
neic2db:	executable shell script
pdecode2css:	executable shell script
smartpick:	executable shell script
timecorr_change:	executable shell script
tkedit_ml:	executable shell script
tkshow_message:	executable shell script
wormwatch:	executable shell script

Issues:
hard-wires in aeic_release.pf were changed from 4.3u to 4.4u
"package require Tclx" was inserted in shells that needed it
The unix 'echo' must be replaced by 'puts' in TCL scripts

local_cdrom_wfdisc did not work due to the mysterious disappearance of FileBox(6) from the distribution. BRTT was contacted about this. The quick response is that due to incr tcl/tk compatibility issues with tcltk8.3, this function should be replaced with tk_getOpenFile. I did this after learning about the new command.

The tcl dbfind command appears to be broken. I have sent a bug demonstration to BRTT and made four atrocious hacks, labelled with "HACK", as workarounds in wormwatch. Dan Quinlan's reply demonstrates that I had misunderstood dbfind. The tcl version starts searching 'after' the specified 'first' record. To make sure that dbfind searches over all records including the first, I specified start record as -1 and removed the hacks.

In wormwatch extract.tcl,

```
regsub {[^/][-a-zA-Z0-9_]*$} $State(dest_db) {{&}} DD
```

replaces

```
regsub {[^/][a-zA-Z0-9-_]*$} $State(dest_db) {{&}} DD
```

so the last "-" is not misinterpreted by the new TCL as part of a botched range.

I had to modify dbchecker_tool so instead of 'exec dbwish' it does

```
# This comment extends to the next line for tcl \
exec $ANTELOPE/bin/wish -f $0 $*
```

```
    package require Datascope
    package require Tclx
```

This is a consequence of dbchecker_tool not yet being incorporated into the lab software distribution (intentional, so we can engineer in the functionality properly).

Now that these changes have been made, we perform a cvs commit from /usr/local/aeic/4.4u/src, then cvs update the copies on earlybird and ice and recompile them.

The next step is to upgrade the rtexec.pf file on nordic and make sure the system runs. This involves replacing all 4.3u with 4.4u occurrences; removing the ELOG_SIGNALS, ELOG_DELIVER, and ELOG_TAG environment variables, which are better handled with elog.pf; and changing the Failure_retry_period from one hour to one day.

Next we need to replace /etc/init.d/antelope with /opt/antelope/4.4u/data/system/S99_antelope, which has a few minor changes (adjusting path and usernames, of course).

In orbassoc, nsta_thresh moved into each grid specification in the parameter file. Also, the ttgrid needs to be generated, which for now we will do with

```
    nordic% cd /iwrun/dev/run/pf
    nordic% ttgrid -pf ttgrid -time now /Seis/databases/stations/master_stations > new_ttgrid
    nordic% mv new_ttgrid /iwrun/dev/run/ttgrid
```

Similarly, upgrade rtexec.pf on fk. This reveals that fk has its own copy of Antelope, thus we need to repeat the distribution copying at the expense of some slight loss of data due to this oversight. I also had to import and compile /usr/local/aeic to fk. Replace /etc/init.d/antelope with the new version.

Next, we need to change the lab user environment over to the new Antelope. Do this by editing /usr/tools/setup/setenv with the above-mentioned environment changes. Export this new file to ice and earlybird. Change ppicker's private setup files on ice to contain the new paths and environment variables.

The next step is to recompile /opt/iceworm from ice. This uses the same source-code tree but installs the products in local bin and lib directories, thanks to the automount organization. With the iceworm distribution recompiled, we can make the same set of changes to the rtexec.pf file and restart the Antelope system. Finally, we of course update the /etc/init.d/antelope and give it execute permission.

Finally, update earlybird: replace and edit /etc/init.d/antelope. Copy the new /usr/tools/setup/setenv to earlybird.

```
    earlybird:ppicker% cd /opt/iceworm/src/4.4u
    earlybird:ppicker% make Include
    earlybird:ppicker% make install
```

Edit rtexec.pf in /iwrun/op/run with the above mentioned changes and make sure the whole system restarts.

Finally, we perform all the necessary changes to upgrade marvin, including copying the code, checking perl, modifying the /etc/init.d/antelope, and changing rtexec.pf.

The megathrust and ugle systems are both down and will remain so until needed.

For the Matlab Antelope toolbox, we run install_matlab_antelope_links which updates a couple of soft-links.

Epilogue

A review of this experience will produce several opportunities to improve next time. For example, a more systematic listing of the independent distributions of each software tree might help ensure thorough upgrades to all systems with minimal omission, backtracking, and downtime. A slightly smoother sequence of events will be possible next time by reviewing the steps taken here and planning ahead for all of them (after accounting for differences between 4.4u and the next version).