Kent Lindquist White Paper

Number 2001-003

Organization of Segmented Data at AEIC

Kent Lindquist

November 5, 2001

Introduction

We need to get our AEIC catalogs up to date. This includes finishing the development of catalog generation procedures for our new infrastructure; organizing old segmented data so missed catalogs can be completed; and establishing procedures and software for the ongoing segmentation of data for analyzed earthquakes. The goal is to have all the catalogs done, then have each new catalog quality-controlled and issued a few days to weeks after the end of the month. Also, we want data for segmented events easily available to researchers. Natasha Ratchkovsky and I began a discussion of the quality-control and catalog-generation issues on 10/4/01, which I will summarize from my notes. A great deal of work has been half-done already, both towards creation of this qc procedure and making segmented data available, so for maximum speed of progress the best route is to complete the development in a way consistent with the engineering vision that has been established.

Organizing and handling segmented data is a project in itself. This white paper summarizes my work so far, getting the segmented data organized both for catalog preparation and for research. This is a running commentary of tasks taken, intentionally left in a serial presentation. Reorganization would hide a lot of the actual reasoning that went into the steps.

The net progress due to the work in this paper is as follows:

- Alaskan AH format is now a native Antelope datatype
- There is a new converter, *aah2db*, to make CSS databases from Alaskan AH files
- Numerous improvements have been made to the suite of schemas, utilities, and databases that convert old station names to new station/component names (correct handling of wfdisc tables, in addition to correct and improved handling of arrival tables)
- All segmented data for analyzed AEIC earthquakes from 6/1/1988 to 11/14/1999 is online; organized; accessible to automated search utilities via aeic_rtsys.pf database-name templates; preserved with the AH files in original form; and expressed with a database layer (wfdisc table) that reflects new station/chan names compatible with current analysis software and site databases.
- An automatic procedure is in place, effective Oct. 21, 2001, which segments data for analyzed events into day-volumes when an analyst checks each day in.
- Segmented data have been retroactively extracted from continuous data (both online continuous data and continuous data reread from Exabyte tapes) from Sept. 1, 2001 to Oct. 20, 2001. This step will also serve as an example for the extraction of missing segmented data from continuous tapes.

Unaddressed issues at the time of this report:

- Day volumes for segmented data must still be concatenated by hand into month volumes. This was left as a manual step to avoid synchronization clashes for data segmentation. We can automate this later however I have deferred that layer of complexity until the basic operation is in place.

- Day volumes for parametric analyzed data must still be concatenated by hand into month volumes.
- If the data segmentation goes awry, the failure must be spotted by the QC person and recovery must be done manually. The data extraction script is saved in the $HOME/process subdirectory of the analyst to make this slightly smoother.
- We need to extract segmented data for events detected between 11/15/1999 and 8/31/2001. This will require purchase of an Exabyte stacker or equivalent technology, plus the dedication of at least 100 GB of disk space (for off-loaded continuous data) to the ongoing task. Some small amount of scripting will also be necessary to achieve this recovery.
- If there are problems with the 6/1/88-11/14/99 dataset, these will have to be recognized by the QC person and addressed by hand, with great care not to disrupt the organization[1] that has been achieved so far.
- Ultimately, the implementation of XDR (Lamont) AH as a native Antelope datatype must also be completed. The full solution would also involve moving *ah2db* to contrib, rewriting, and combining it with *aah2db* so the user does not have to be bothered with details of AH type.
- Myriad tasks remain such as QC checkout mechanism, database transitions from QC'd to AEIC catalog; db2catalog rewrite; etc. that do not directly affect segmented data.

Plans for re-establishing AEIC catalog generation

This section transcribes notes from a meeting with Natasha Ratchkovsky, with whom I will be working to get AEIC catalogs back online.

Things we need to do: check for travel-time and magnitude residuals.

What exists now? Dan's dbchecker and aeic_dbclean, the latter of which needs to expand.

We need weekly report mechanisms integrated.
We need monthly report mechanisms integrated.

We need to update the tracking database. Catalogs 1991 to May 1995 are done. There is a gap from June 1995 to August 1998, though thanks to Mitch Robinson the wavefom data are online. June 1995 is halfway done. Sept. 1998 to June 1999 is done. July 1999 onwards are not done.

The waveform data through November 1999 are online but need organization and reformatting.

The picks from June 1999 to present are online and almost ready for review, though they may need some cleaning.

The picks from June 1995 to August 1998 are online in CSS format and probably ready.

---

1. "Organization" is synonymous with "Support for automation." Disorganization destroys the ability to automate tasks, thus forcing humans to do the work of the computers (either directly through data manipulation by hand, or indirectly via the cost of software development to handle needless complexity).

We need a segmenting mechanism for current waveform data.

We need to load data online for December 1999 through the present.

We need travel-time model issues decided.

We need Dan M.'s catalog generator db2catalog working again. That was a temporary lash-up which needs to be rewritten.

What is the status of /Seis/catalogs/aeic, for example does the 1995 McNamara work exist there in up-to-date form?

Proposed division of labor:
Kent: data organization and software
Natasha: database review and catalog generation

We will conduct simultaneous efforts to
1) Keep up to date with current catalogs
2) Fill in old catalogs

We will start with a guinea pig database month: July 1995. Use this to identify all the things we want the Quality-Control procedure to accomplish. This gives us an initial dialogue:

- Kent delivers database for July 1995 to Natasha
- Natasha reviews it with dbloc2 and finds everything that's wrong, making a written list (sorted into things that are automatically detectable, for warnings and perhaps automatic cleanup; and things detectable by humans only)
- Kent performs code fixes and re-establishes db2catalog
- Natasha runs db2catalog and creates July 1995 catalog report

While Natasha does her parts on July 1995, that gives Kent time to prepare data for August 1995 and September 2001, and also to work on database organization.

Organization and Conversion of Old Segmented Data

Mitch has loaded segmented data files from June, 1988 through November, 1999 onto /Seis/seg sorted by year and then by month. These data are in Alaskan AH format, i.e. the old Lamont AH format that does not use XDR data representation. Mitch is pretty sure this AH format is always big-endian, which seems to be supported by a comment in ah2xdr source code about the old ah format being 'Sun AH'.

These data have wfdisc tables which were created with an old program, lean_ah2css. The resulting wfdisc files have a number of problems. First, not all files uncompressed correctly, leaving some corrupted rows. Second, that program faked the channel names and left station names

untouched, thus the waveforms are out of synch with our site database and pick storage. If we are going to handle a decade of segmented waveform data, it is time to do this job right.

First, the statrans1.0 database along with the translator tools tabulate_missed_stations, capitalize_station, and translate_stachan_names were moved from /usr/local/aeic and submitted to the Antelope Users Group contrib area. Other community members have mentioned an interest in these utilities, so it's time to put them in the right place.[1]

Now is the time to put the Alaskan AH format into Antelope and CSS3.0 as an official datatype. Investigation begins at the addwf(5) man page. There are several notes about this particular addition. Dan Quinlan already added datatype "aa" to the CSS3.0 description of datatype and to trdefaults.pf, with the name 'AAH' internal to the trace library (both of these abbreviations mean "Alaskan AH"). Datatype 'aa' is not yet in the CSS3.0 datatype Range, however. Also, the trdefaults stock parameter file refers to a header function wfhdrAAH which actually already has a placeholder function built into the Antelope libtr. Incidentally Antelope support of XDR AH is as yet incomplete. After discussion with Quinlan, for now I will develop wfinAAH, wfoutAAH, and wfhdrAAH in an AEIC *libuser*, then submit these to BRTT when done. Since the existing prototype wfhdrAAH is hard-wired into libtr, I will have to do this development under a different datatype name. Otherwise my tests will never be able to see the developing wfhdrAAH, even if renamed in trdefaults as wfhdrAAHtmp (I tested this and found out the hard way after extensive coding). There is a further difficulty with defining nonexistent new datatypes which brings me to a stopping point without help from Dan Quinlan. The problem is described here:

> I can't finish my development under datatype 'aa' because the prototype wfhdrAAH is already built into libtr. I can't sidestep that name because even if I put trwfhdrAAHtmp in trdefaults.pf, trwfhdr.c isn't listening.
> I also don't know how to perform the unappealing temporary hack to tell things like trexcerpt to use the wfhdrAAH in libuser instead of the one in libtr.
>
> I can't finish my development under datatype 'ao' because I need a different tr code for it, e.g. trAAHO. Even if I put that row in trdefaults.pf, trexcerpt tells me
>
> Can't interpret 'trAAHO' in trdefaults.pf
>
> This is because, even though trinit.c carefully used Waveform_types from the trdefaults.pf &Tbl, the trace library went on to check each entry against the pre-approved datatypes it already knows about.
> I don't know how to perform the unappealing temporary hack to tell libtr to use my Trxlat table instead of the one compiled into the library.

---

1. A large part of the strategy for succeeding in this large software job with only a few people, is to take a bulldog mentality towards eliminating mediocrity in the data organization and the toolkits. Doing things right the first time introduces an efficiency that is critical for success. Also, this simultaneously creates a long-term investment since the organization of the ground floors governs the quality and height of the rest of the structure. In a similar vein, submitting these tools to the community triggered the addition of a schemas table to the Antelope reference guide in the process of adding notes about the new tools.

Most importantly I don't know what the 'right way' would be.

In addition I see that tr.h has
    #define trAAH 21
which suggests I'm going to have trouble down the line using trAAHO without a corresponding preprocessor macro. I'm unsure what the further consequences are here.

The solution to this puzzle [thanks much to Dan Quinlan who explained it] is that if the waveform-type code is an integer instead of a predefined macro, that integer gets used automatically. The only caveat is that one has to choose an integer that isn't already assigned to a predefined code in tr.h. Thus I've reassigned the datacode for type 'aa' to 55 (instead of 'trAAH') and I can now develop all the component functions and convert databases as necessary, with forward compatibility once these changes are officially incorporated. I believe this integer should be dynamically assignable [i.e. if I use an unrecognized alphanumeric datacode, the trace-library chooses an integer for it]; that feedback has been submitted to BRTT.

In principle, the AAH write routines could scan the data and decide if they would fit in short-integers instead of floats, without clipping or truncation. I am not going to implement this decision-making because it would prevent dynamic updating of AH files if they are being continually written, for example by orb2db. Repeated calls to wfhdrAAH and wfoutAAH would not be able to change the datatype of the previously written data points. Thus, for the fully general case, I will implement Alaskan AH file output as float only.

I omitted a couple things in writing the current version of wfAAH.c. The routines do not look up calibration curves and event information from the database, in order to fill these in. Also, Alaskan AH files are currently listed as not appendable.

In the course of building wfinAAH I needed *ah2asc* [so I could dump my test file to see what was wrong with my parsing attempts], which I found as a soft-link in /usr/tools/bin linked to /usr/local/we/src/ah/ah2asc/ah2asc. There is already an ak_ahheader.h in Antelope contrib via the iceworm libds library. Thus, I moved the rusted[1] source code for ah2asc into my tree of maintained source-code /usr/local/aeic. In the process, I found that the installed ak_ahheader.h had some mis-

1. "rusted" means unable to recompile without hacking, thus likely to break without warning on the next update of any particular software component or operating system feature. It is necessary to move code pieces like this into maintained source-code trees to keep them working in the midst of library, operating-system, and software-package upgrades. I have heard people propose at times that in order to avoid the problem that "software rusts", we simply freeze all source code, software packages, and operating systems. That's not usually realistic. Improvements in hardware (larger disks and faster chips) drive changes in operating systems. Most people want to buy bigger drives and faster computers when they come on the market. Changes in operating systems and response to increasingly complex user needs and demands drives software package updates. Thus we must acquiesce to being in a dynamic environment. Coming up with strategies to keep all of our technology working in such a dynamic environment is a specialty in itself, which needs to be delegated to professionals familiar with those strategies. The topic of how to create and maintain a dynamic source-code tree while providing continued functionality to users is beyond the scope of this footnote.

takes, some omissions from the original non-XDR Lamont format, and some poorly chosen names. Thus, I fixed all these[1], which also of course meant that I had to propagate the fixes through /usr/local/aeic, $ANTELOPE/contrib, and /opt/iceworm so the source trees are all consistent.

The next step is to write a converter for Alaskan AH files to the database. There is already an *ah2db*, written by Geoff Abers for Lamont-style (modern) XDR-based AH files. This source code is part of Antelope itself, though I have Dan Quinlan's permission to move it to the contrib area. The correct way to do this job is to write a new *ah2db* which automatically detects whether the AH file is Alaskan (non-XDR) or modern Lamont format, and handle it appropriately. There are too many small details to resolve all at once, however, so as a stepping stone I will write *aah2db.* For now aah2db will just handle creation of wfdisc rows (other details will include site information, calibration information, and event info). The new aah does, however, handle multiple-block AH files (more than one header/data pair). Also, importantly for our current segmented data, *aah2db* will correctly handle files that are compressed.

We are beginning to see some repeated source-code, which may need to be organized into a library[2], i.e. *libaah*. The components are:

- ak_ahheader.h
  currently installed from $ANTELOPE/contrib/src/lib/iceworm/wormds
- n2h_ak_ahhead
  currently in aah2db.c
- h2n_ak_ahhead
  currently in wfAAH.c
- get_null_ak_ahhead
  currently in $ANTELOPE/contrib/src/lib/iceworm/wormds and in wfAAH.c of *libuser*
- aah_datatype_to_size
  currently in wfAAH.c and aah2db.c
- aah_abstime_to_epoch
  currently in aah2db.c

Currently, a *libaah* might make sense for contrib, however that would mean wfAAH of the trace library would depend on *libaah*, requiring adding *libaah* to the TRLIBS macro of Antelopemake.

---

1. These types of changes are examples of what it means to maintain a source-code tree: Continual small changes to improve the quality. As systems grow more complex, one needs to constantly improve the organization so developer and user time may be spent on the complexities of the science and technology problems themselves, rather than on complexities caused by bad engineering. This strategy is the only way to make forward progress in an environment where one of the dominant limiting terms is developer time.
2. Another standard strategy to keep a source-code tree clean and efficient. The goal is to have the division between functions and groups of functions (libraries) well thought out. The software should express current thinking on the way to represent the real-world problem. As one obtains a more and more thorough understanding of the real-world problems, the organization of the software tree should keep step so developers can write programs that 'do the right thing'. Lack of correct structure means time is spent drowning in disorganization instead of improving functionality.

Now start reorganizing the existing segmented data. I moved all the wfdiscs Mitch created to /Seis/seg/old_wfdiscs. Those are not done using the native aa data format. Also they have a number of filename-translation problems and problems resulting from attempts to read compressed files.
Mitch has /Seis/seg/19YY/MM/wf/YYMMDD etc. Change to /Seis/seg/wf/19YY/MM/YYMMDD etc. The goal here is to remove the hard-wiring of wfdisc structure to month and year volumes. 'wf' is an artefact of the division into separate wfdisc volumes. This removal of 'wf' is accomplished from /Seis/seg with

```
nordic# foreach y ( ???? )
foreach? cd /Seis/seg/wf/1988/*/*/f/$y
foreach?  foreach m (??)
foreach?  cd /Seis/seg/wf/$y/$m
foreach?  mv wf/* .
foreach? rmdir wf
foreach? end
foreach? end
```

There are a number of incorrectly nested waveform directories, i.e. waveform directories for single events which themselves contain waveform directories for other events. These are uncovered with

```
nordic% pwd
/Seis/seg/wf
nordic%  foreach y ( ???? )
foreach? echo $y
foreach? cd /Seis/seg/wf/$y
foreach? foreach m ( ?? )
foreach? echo $y $m
foreach? cd /Seis/seg/wf/$y/$m
foreach? ls -1 -d */* | egrep '/$'
foreach? end
foreach? end
```

which shows the following problem cases:

```
881122041106/881122041106/
881122093906/881122093906/
881122184716/881122184716/
881217090220/881217090220/
881217205550/881217205550/
881230003935/881230003935/
890101185252/890101185252/
890102020332/890102020332/
890102051222/890102051222/
890102113622/890102113622/
```

890102162031/890102162031/
890102182151/890102182151/
890102182301/890102182301/
890102200541/890102200541/
890102203051/890102203051/
890102221621/890102221621/
890102221841/890102221841/
890105043210/890105043210/
890105044620/890105044620/
890105104730/890105104730/
890118162557/890118162557/
890204081742/890204081742/
890823060354/89082300/
890824022500/890824022434/
891213225259/891213225259/
900317130716/900317175757/
910306192403/#066048/
910306195843/#066560/
910306205343/#067072/
910307210201/#072704/
910307210201/910307210201/
910407072728/910407072727/
910508053412/910508053411/
911010140508/911010140509/
911125231710/911125231711/
911224134833/911224134832/
920211025010/920211025011/
920211025011/920211025010/
920307100637/920307100636/
920317115832/920317115833/
920428233211/920410234500/
920428233211/920423003228/
920428233211/920423045618/
920428233211/920423154418/
920522174135/920522174134/
920611233104/920611233105/
920621061937/920621012258/
920807192230/920807192229/
921021133004/921021133544/
921220080351/921220080350/
930531181854/930531181904/
930613233758/930613112109/
930613233758/930613112539/
930613233758/930613113929/
930613233758/930613124859/
930613233758/930613145729/

930613233758/930613180858/
930628234916/930628112356/
930712233947/930710124211/
930712233947/930710142630/
930811070529/930811070530/
930821200350/930821200244/
930915235645/930908180505/
930915235645/930908181845/
930915235645/930908184005/
930915235645/930908185035/
930915235645/930908192335/
930915235645/930908193155/
930915235645/930908193415/
930915235645/930908193535/
930915235645/930908193735/
930915235645/930908194105/
930915235645/930908222415/
930915235645/930908222935/
930915235645/930908230025/
930915235645/930909001835/
930915235645/930909010413/
931027200706/931027200707/
931027200708/931027200707/
931211044338/931211044016/
940129000117/940129000116/
940214133606/940214133607/
940330024644/940330024645/
940412024220/940412024219/
940412084439/junk/
940523065746/940523065747/
940709133842/940709133841/
940721193940/940721193950/
940728020400/940728020400/
940831192501/940831192502/
940907125024/test/
940922225551/940922095531/
940922225551/940922112231/
940922225551/940922113101/
940922225551/940922113102/
940922225551/940922135241/
941122025730/941120210230/
941122025730/941121011830/
941122025730/941121022930/
941122025730/941121042030/
941122025730/941121193030/
950106221510/950106221511/

950106221511/950106221511/
950211054023/rede/
950214205414/results/
950214205414/tmp/
950720001139/950720001019/
951106114230/9511061140/
951113014423/951113014422/
951117215037/951117215038/
960102090129/960102090119/
960117130726/960117130725/
960201125309/960201125410/
960205112406/960205103456/
960229173403/960229173403/
960229173404/960229173403/
960311061306/tmp/
960401133703/960401133603/
960403172508/960403172630/
960425090939/960425090909/
960525132428/960525132358/
960902074406/960902074226/
960902074406/960902074406/
961129084918/961129084818/
970120163506/970120163406/
970120181526/970120181426/
970209214225/970209214115/
970209214225/970209214155/
971125221931/971125221932/
980717093630/980717023636/
980727084527/980727084525/
990125041944/990125041943/
990212032723/990212032722/

This may be only a partial subset, since I got ‘argument list too long’ from ls. We will have to double-check the cleanup via a method based on find(1) or with tighter nesting on the foreach, after this first round. There are several different failure mechanisms at work here.Thus, I see no other way to clean them all up than to go through them by hand, making sure not to erase or incorrectly move any important data. In some cases, the data are the same in the main and sub directories, although with some files not compressed in the main directory and some files missing from the main directory. These problems probably resulted from some compression script. Fixing these usually requires recompressing all files and doing a recursive diff between the parent and sub directories, then erasing the duplicate entries, moving missed entries to the parent, and removing the nested subdirectory. For contiguous occurrences where the parent and sub-directory names are the same, we can help get through the comparisons with

```
nordic# foreach f (890102051222 890102113622 890102162031 890102182151
890102182301 890102200541 890102203051 890102221621 890102221841 890105043210
890105044620 890105104730 890118162557 890204081742)
foreach? cd /Seis/seg/wf/1989/01/$f
foreach? compress $f/*
foreach? diff $f .
foreach? end
```

...then remove by hand the matching directories. Some problems with conflicting files I have postponed:

```
mv 890118162557/ ../890118162557_B
```

The next type of problem is that there are subdirectories of some events which appear to be for a completely different event, or for a second, time-shifted segmentation of the same event. This often results from the once-common procedure of "cloning an event", i.e. copying the event to a different directory with the origin time shifted by one second, in order to locate a second event that occurred in the same trigger[1]. Despite the possibility of leaving overlapping data, rebuilding these data files with no overlap is a large undertaking which is also unnecessary since the Antelope tools will survive the duplicate wfdisc entries. I am going to postpone that for another cleanup phase in the distant future. All such directories will be moved up to be parallel with the main waveforms. Also, because of the nested structure, many of these missed the compression stage, which I will fix by hand. Of course, it is true that for overlapping segmentations we want to subordinate the second copy somehow. We can't do that by messing up the directory structure, however, which destroys the conversion process. A few of these duplicate segmentations have completely cryptic names for the duplicate copies. Fix this by switching to the common 'origin time + one second' format in the rest of the waveforms:

```
mv 910306192403/#066048/ 910306192404
mv 910306195843/#066560 910306195844
```

I will spare the whole list. There are other of these directories whose subdirectory structure is more convoluted. Worse yet, there are numerous instances where the off-by-one directory name occurs both in the correct location and as a subdirectory of the original source of the clone, with conflicting pickfiles. A nice example:

```
920211025010/920211025011/
920211025011/920211025010/
```

After cleaning these all up, it is necessary to re-survey by a different method to find other nesting problems. Try the heavy-handed approach

```
nordic# find . -type d -print > /home/kent/work/aah/dirs
```

and use vi search expressions to remove acceptable patterns. This turns up another 41 problems

---

1. This type of strategy shows the weakness of event-based recording, since it is hard to separate the waveform storage from the storage of hypocentral parameters. The result in this case is duplicate copies of the same waveforms. In addition to wasting storage space, that makes correcting waveform problems much more involved if there is more than one copy of some of the waveforms. One of the principal payoffs of switching to a properly normalized relational database is to avoid this type of issue, still allowing event-based sorting when necessary but also keeping the waveform storage clean.

./1988/12/881217090220/881217090220
./1989/01/890118162557_B
./1995/08/950828120456/950828133426
./1995/08/950828194650/950828194620
./1995/10/951005084631/951005084632
./1995/10/951006052141/tmp
./1995/10/951011160645/951011160644
./1995/10/951018031517/951018031430
./1995/05/950524111035/950524111034
./1996/10/961005002122/961005002121
./1996/10/961006122752/961006122712
./1996/10/961022222244/961022222243
./1996/08/960819024108/960819024238
./1996/08/960819024108/960819024238/960819024108
./1996/08/960819024238/960819024108
./1997/09/970911060739/970911060738
./1997/09/970926081830/970926081821i.
./1997/10/971003064747/971003064747
./1997/10/971003064747/971003064747/971003064747
./1997/10/971003064747/971003064747/971003064747/971003064747
./1997/10/971003064748/971003064747
./1997/10/971003064748/971003064747/971003064747
./1997/10/971003064748/971003064747/971003064747/971003064747
./1997/07/970707062651/970707062716
./1997/07/970707062651/970707062716/970707062651
./1997/07/970707062716/970707062651
./1997/07/970726095216/970726095216
./1997/06/970609091123/970609090000
./1997/06/970609091123/AH1
./1997/06/970609091123/AH1/970528031654
./1997/06/970609091123/AH1/970528041743
./1998/03/980304204000/980304204001
./1998/03/980321205826/980321205825
./1998/11/981106105003/981106105004
./1998/09/980925134451/980925134450
./1998/04/980418105807/980418105806
./1998/05/980510060001/980510060000
./1998/05/980512225730/980512225731
./1998/05/980518183100/980518183000
./1998/05/980518183100/980518183100
./1999/07/990702053319/990702053319w.

which we will once again fix by hand.

There appear to be a number of compressed pickfiles in these directories also, which will need to be sidestepped with something like

nordic% ls wf/1988/06/*/* | egrep -v '[89][89][^/]*.Z'

For the wfdisc tables, mkdir /Seis/seg/YYYY. Now we convert the ah files with a nested foreach

```
nordic% cd /Seis/seg/wf
nordic% foreach y ( ???? )
foreach? cd /Seis/seg/wf/$y
foreach? foreach m (??)
foreach? cd /Seis/seg/wf/$y/$m
foreach? foreach e ( `ls -1` )
foreach? cd /Seis/seg/wf/$y/$m/$e
foreach? echo $y $m $e |&  tee -a /home/kent/work/aah/conversion_note
foreach? aah2db -v `ls * | egrep -v '[89][0-9][^/]*'` /Seis/seg/$y/$y\_$m |& tee -a /home/kent/
work/aah/conversion_note
foreach? end
foreach? end
foreach? end
```

This actually took multiple tries due to read permissions, directory listing permissions, etc. It also inspired several robustness fixes to *aah2db.* The timescale for running all of these conversions appears to be over a week on a Sun Blade 1000. The multiple attempts resulted in

```
conversion_note   (1988/06 to 1989/08)
conversion_note2   (1989/08 to 1989/12)
conversion_note3   (1990/01 to 1994/07)
conversion_note4   (1994/07 to 1994/12)
```

There are too many errors in these due to file conversion problems: bad datatype values, corrupt compression, bad time fields, good header/data blocks followed by bad header/data blocks in the same file, empty directories that result in attempts to read and convert the non-existent file '*'. Erase all wfdiscs and start over completely, after hardening *aah2db.*

We'll start by using 1995_07 as a test case. Go through the full procedure to prepare a dataset for Natasha's catalog-production work to proceed

```
nordic% cd /Seis/seg/1995
nordic% capitalize_station 1995_07
```

This turns up another type of problem file: AH files with a small amount of garbage at the end of a valid header/data block: enough garbage to trigger continued file processing, but not enough to completely replace the header structure and trigger error detection.

Fixing and rerunning the conversion and the capitalize_station script on 1995_07, the next step is

```
nordic% tabulate_missed_stations /Seis/databases/station_name_translations/
station_name_translations /Seis/seg/1995/1995_07
```

This creates a list and a database (/home/kent/work/aah/dbmissed) of stations in /Seis/seg/1995/1995_07 which do not appear in the station_name_translations database. For 1995_07, the missing stations are CP2T, CRPT, GOEE, IRGE, IRGH, IRH, IRIG, and XXX. These omissions must be resolved before running the translation script, since the translation script may not be run twice (otherwise RDTN->RDT_SHN, then on the second step RDT->RDT_SHZ since the translator looks only at the station names, mislabelling the channel, with similar problems on many other stations). GOEE, IRGE, IRGH, IRH, and IRIG are all timecode, thus get the channel name "TIME" with the station name unchanged. I considered erasing the XXX station, since it strongly appears to be temporary and meaningless. However, since there is at least one good signal there I will be conservative and, reluctantly, add XXX SHZ to the translation database, matching the station-code in the AH file. As for CP2T and CRPT, there are no traces of these stations in Steve Estes's station list, in the site database, in stations.dat, or in the knowledge of Guy Tytgat and Scott Stihler. The AH files say "SHZ" and there are no picks for these in /Seis/catalogs/aeic/1995_07/aeic_1995_07, so I will just put them in the translation database as "SHZ". Rerunning *tabulate_missed_stations* shows we are ready for the next step ("0 missed stations"). Finally, translate the station-channel names:

```
nordic% translate_stachan_names /Seis/databases/station_name_translations/
station_name_translations /Seis/seg/1995/1995_07
```

To get Natasha started, I copied /Seis/catalogs/aeic/1995_07/aeic_1995_07 to /Seis/space/1995_07/qcwork_1995_07, with dbdescriptor reference to the /Seis/seg/1995/1995_07 wfdisc table. There are still some segmentation-fault problems occurring with the data in dbpick.

In the meantime, begin wfdisc creation for the rest of the segmented waveforms. This was done with a slightly different method, to accommodate launch from a PPP connection. We made the script

```
nordic% cat go
cd /Seis/seg/wf
foreach y (1988 1989 1990 1991)
cd /Seis/seg/wf/$y
foreach m (??)
cd /Seis/seg/wf/$y/$m
foreach e ( `ls -1`)
echo $y $m $e >>! /home/kent/work/aah/conversion_note2
cd /Seis/seg/wf/$y/$m/$e
aah2db -v `ls -1 * | egrep -v '[89][0-9]*'` /Seis/seg/$y/$y\_$m >>& /home/kent/work/aah/
conversion_note2
end
end
end
nordic%
```

and launched this with

```
nordic% nohup source go &
```

after which the PPP connection was terminated. The complete set of waveforms will be translated in several stages by changing the directories cited in this script. Output files are conversion_note{2,3,4,5}.

A couple restarts were required, one in the middle of a directory which was accomplished by replacing

```
foreach e ( `ls -1`)
```

with

```
foreach e (`ls -1 | sed -e '1,/990921150116/d'`)
```

for Sept. 1999, which completed through the specified event, after which the disk filled up and no further conversions would launch.

After completing all the wfdisc creation, I scanned the conversion_note files by hand for unexpected errors. Aside from a few bad data files, everything looks reasonable on overview.

The next step is to run the station-name conversion process on these tables. First:

```
nordic% cd /Seis/seg
nordic% foreach d ( `ls -1 ????/*.wfdisc | sed -e 's/.wfdisc//g'` )
foreach? capitalize_station $d
foreach? end
```

Second:

```
nordic% tabulate_missed_stations
/Seis/databases/station_name_translations/station_name_translations `ls -1 /Seis/seg/????/
*.wfdisc | sed -e 's/.wfdisc//g'` |& tee missed_notes
```

This reveals 201 stations that are not in the translation database. These must all be resolved before the translation of old station names to new station names can begin. After several days of attempts to identify correct component names [interviews with John Power, Guy Tytgat, etc.], it is clear we need better tools. I modified *tabulate_missed_stations* to provide a more informative report on the missed stations, including time of first occurrence and component names that were taken out of the AH files. Analyzing this output, some of the 'stations' are actually labelling errors confined to a few months, for example some of the arrays and COLA get shifted: OLAA is clearly COLA etc. Fix this directly in the waveform database since they are not station-name-translations in the strict sense.

```
nordic% cd /Seis/seg/1997
nordic% dbset 1997_07.wfdisc sta 'sta == "OLAA"' COLA
nordic% dbset 1997_07.wfdisc sta 'sta == "L011"' IL01
nordic% dbset 1997_07.wfdisc sta 'sta == "L022"' IL02
nordic% dbset 1997_07.wfdisc sta 'sta == "L033"' IL03
nordic% dbset 1997_07.wfdisc sta 'sta == "L044"' IL04
```

```
nordic% dbset 1997_07.wfdisc sta 'sta == "L055"' IL05
nordic% dbset 1997_07.wfdisc sta 'sta == "L066"' IL06
nordic% dbset 1997_07.wfdisc sta 'sta == "L077"' IL07
nordic% dbset 1997_07.wfdisc sta 'sta == "L100"' IL10
nordic% dbset 1997_07.wfdisc sta 'sta == "L111"' IL11
nordic% dbset 1997_07.wfdisc sta 'sta == "L122"' IL12
nordic% dbset 1997_07.wfdisc sta 'sta == "L133"' IL13
nordic% dbset 1997_07.wfdisc sta 'sta == "L144"' IL14
nordic% dbset 1997_07.wfdisc sta 'sta == "L155"' IL15
nordic% dbset 1997_07.wfdisc sta 'sta == "L166"' IL16
nordic% dbset 1997_07.wfdisc sta 'sta == "L177"' IL17
nordic% dbset 1997_07.wfdisc sta 'sta == "LBBB"' ILBB
nordic% dbset 1997_07.wfdisc sta 'sta == "LL11"' ALL1
nordic% dbset 1997_07.wfdisc sta 'sta == "LL22"' ALL2
nordic% dbset 1997_07.wfdisc sta 'sta == "LL33"' ALL3
nordic% dbset 1997_07.wfdisc sta 'sta == "LL44"' ALL4
nordic% dbset 1997_07.wfdisc sta 'sta == "LL55"' ALL5
nordic% dbset 1997_07.wfdisc sta 'sta == "LL66"' ALL6
nordic% dbset 1997_07.wfdisc sta 'sta == "LL77"' ALL7
nordic% dbset 1997_07.wfdisc sta 'sta == "XBBB"' FXBB
nordic% dbset 1997_07.wfdisc sta 'sta == "X011"' FX01
nordic% dbset 1997_07.wfdisc sta 'sta == "M011"' BM01
nordic% dbset 1997_07.wfdisc sta 'sta == "M022"' BM02
nordic% dbset 1997_07.wfdisc sta 'sta == "M033"' BM03
nordic% dbset 1997_07.wfdisc sta 'sta == "M044"' BM04
nordic% dbset 1997_07.wfdisc sta 'sta == "M055"' BM05
nordic% dbset 1997_07.wfdisc sta 'sta == "C011"' BC01
nordic% dbset 1997_07.wfdisc sta 'sta == "C022"' BC02
nordic% dbset 1997_07.wfdisc sta 'sta == "C033"' BC03
nordic% dbset 1997_07.wfdisc sta 'sta == "C044"' BC04
nordic% dbset 1997_07.wfdisc sta 'sta == "C055"' BC05
nordic% dbset 1997_07.wfdisc sta 'sta == "T011"' TT01
nordic% dbset 1997_07.wfdisc sta 'sta == "LDYY"' BLDY
nordic% dbset 1997_07.wfdisc sta 'sta == "LHAA"' BLHA
nordic% dbset 1997_07.wfdisc sta 'sta == "NTCC"' CNTC
nordic% dbset 1997_07.wfdisc sta 'sta == "RLKK"' BRLK
nordic% dbset 1997_07.wfdisc sta 'sta == "N7AA"' PN7A
nordic% dbset 1997_07.wfdisc sta 'sta == "S1AA"' PS1A
nordic% dbset 1997_07.wfdisc sta 'sta == "S4AA"' PS4A
nordic% dbset 1997_07.wfdisc sta 'sta == "SOMM"' MSOM
nordic% dbset 1997_07.wfdisc sta 'sta == "TBLL"' MTBL
nordic% dbset 1997_07.wfdisc sta 'sta == "NATT"' MNAT
nordic% dbset 1997_07.wfdisc sta 'sta == "RR33"' DRR3
nordic% dbset 1997_07.wfdisc sta 'sta == "NCKK"' ANCK
nordic% dbset 1997_07.wfdisc sta 'sta == "CIRR"' MCIR
nordic% dbset 1997_07.wfdisc sta 'sta == "GLSS"' MGLS
```

nordic% dbset 1997_07.wfdisc sta 'sta == "GODD"' MGOD

These changes are corroborated by the filenames, which appear healthy.

Next we need to change the behavior of the *translate_stachan_names* script: a blank channel name in the translation database will now indicate that whatever pre-existing channel name is in the database should be used. This is necessary for multi-component stations that exist alongside old-style stations that need to be translated. Also, we will add a parameter file for channel aliases, such as "spz" for "SHZ", "shp" for "BDF", "shn" for "SHN" etc. These translations will be applied when chan is blank in the translations database.

This leaves just a few mysteries. ZHIN, ZRAG, ZSGA, and ZWAX appear only in August 1989. I believe they are temporary names for the stations HIN, RAG, SGA, and WAX, and will translate them as such. After talking with John Power, Guy Tytgat, and Steve McNutt, I have made HOMP, GOUP, STRP, DFRP, and CPAP into BDF channels, chopping the 'P' off the station name. Stations 'spb' and 'sps' show up in /usr/hypoe/stations.dat, so I will assume SPBZ and SPSZ are the BHZ and SHZ components of SPB and SPS. The remaining stations with '#' at the end I found explained in a January 29, 1996 email from John Lahr. These changes along with all others were transferred from the modified dbmissed.statrans onto /Seis/databases/station_name_translations/station_name_translations.statrans. In order to make use of the changes, I also had to upgrade the script *translate_stachan_names*(1). New features include support for the blank-channel-name convention; a 'verbose' switch; a 'report plans without executing' switch; an optional parameter file to specify blanket translations of specific channel names, e.g. "spz" -> "SHZ"; and the ability to loop over multiple databases. This allows us to pre-approve the intended translations:

nordic% translate_stachan_names -nv /Seis/databases/station_name_translations/station_name_translations 'ls -1 /Seis/seg/????/*.wfdisc | sed -e 's/.wfdisc//'' >& /home/kent/work/segdata/trans_plans

This reveals 1394 intended translations, which need to be reviewed. There are a great many errors here, often due to three-component stations being entirely assigned to SHZ. This is due to earlier pickfile conversion, where the pick was assumed to be on the vertical component during the translation. I will go back in the translation database and set the vertical-component stations to have newchan set to NULL, in accordance with the new convention for the translation database. This may mean that future translations of databases made from pickfiles may have difficulty, since the pickfiles will not have even the flaky component information that we are getting out of the AH files. The solution at that point will be to add a 'defaultchan' field to the statrans1.0 schema, filling in the default (assumed) channel for multi-component stations when no channel information at all is available. To support this, *translate_stachan_names*(1) will have to use the defaultchan entry when no other information is available about component name. Some examples of what I'm about to fix:

```
Planning AKSE_she      => AKS_SHE
Planning AKSE_shz      => AKS_SHE
Planning AKSN_shn      => AKS_SHN
Planning AKSN_shz      => AKS_SHN
```

```
Planning AKS_SHE       => AKS_SHZ
Planning AKS_SHN       => AKS_SHZ
Planning AKS_SHZ       => AKS_SHZ
Planning AKS_she       => AKS_SHZ
Planning AKS_shn       => AKS_SHZ
Planning AKS_shz       => AKS_SHZ
```

Note that the incorrect assignment of channel names for some of the AKSN and AKSE stations (a difficulty that is endemic to the whole set of AH files) makes this conversion less simple to automate than it looks.

On reconsideration, what I have done to achieve correct translation of segmented-data wfdisc tables has broken the translation of arrival tables from pickfiles. Thus we have gone backwards, which will be a problem. Natasha will very likely use Xpick to review old catalogs, given our current problems with location residuals; thus the final AEIC catalog will have to be regenerated from pickfiles. Therefore I have to fix the whole procedure so it works again for pickfile conversion as well as AH file conversion. First, I made a new schema statrans1.1 which adds the 'defaultchan' field to the statrans field. For completeness, I modified the output dbmissed database of *tabulate_missed_stations*(1) to use the new schema. I used *dbconvert*(1) to transfer /Seis/database/station_name_translations/station_name_translations to the new schema. Then I expanded *translate_stachan_names*(1) to assign the defaultchan if all other options are exhausted (newchan is null; existing channel name not in chantrans array; existing channel name not preassigned to something non-null). Finally, I added default channels to all the entries (138 of them) in the station_name_translations database that had null values for newchan. This was done based on knowledge of the waveforms that exist and recognition of individual stations. Following these fixes, rerun the *translate_stachan_names -n* and review intended changes. This immediately reveals another translator bug. Fix and repeat. Fix errors in translation database and review intended translations; iterate repeatedly until correct.

The final state of the channel-name-translation parameter file is:

```
nordic% cat translate_stachan_names.pf
chantrans &Arr{
        bhe BHE
        bhn BHN
        bhz BHZ
        ppp BDF
        she SHE
        shl SLZ
        shn SHN
        shp BDF
        SHP BDF
        sht TIME
        shz SHZ
        spz SHZ
        tcc TIME
```

```
}
nordic%
```

Of course, the bulk of the translation information lies not in these aliases but in the station-specific instructions in the station_name_translations database. Now that all of the intended translations are correct, we can apply the corrections:

```
nordic% translate_stachan_names -v /Seis/databases/station_name_translations/
station_name_translations ‘ls -1 /Seis/seg/????/*.wfdisc | sed -e ’s/.wfdisc//’‘ > & /home/kent/
work/segdata/trans_notes
```

To make these databases available to automatic find utilities, I have added the following line to aeic_rtsys.pf:

```
segmented_waveform_database /Seis/seg/%Y/%Y_%m
```

This line is a template to be used with the *str2epoch*(3) family of commands in Antelope. It means that all you need to know is the epoch time of interest and this template will return to you the name of the correct database in which to look for segmented data.[1]

Establishing Routine Segmentation of Data

Now that previous segmented waveforms have been restored and made available, the next step is to establish routine segmentation of event data, immediately after the analysis is done.

The subsetting expression for data segmentation has two parts:
1. Choosing the correct length of waveform segment based on the hypocentral location
2. Choosing which stations to include, based on distance from event and magnitude

For the distance from the event, we have decided to use the following criteria:

- start 20 seconds before the IASP91 predicted P-phase arrival
- Choose the longer of:
- parrival + 2*(S-P time)
- arrival of phase with apparent-velocity 2.8 km/sec
- Add 30 seconds padding to this to find the end-time for the segment

These conditions allow the capture of Lg (assuming an Lg window around 3.5 to 3.0 km/sec), and also make sure that at least some data are saved for stations close to the earthquake.

To help find a magnitude cutoff, the following Matlab commands are helpful:

1. This strategy is used for other classes of databases at AEIC as well.

```
db=dbopen('/Seis/processing/analyzed/2001_10/analyzed_2001_10_20','r');
db=dbprocess(db,{'dbopen origin';'dbsubset ml != NULL';'dbjoin assoc';'dbsort orid
ml';'dbgroup orid ml'});
maxrange=dbeval(db,'max(assoc.delta)');
ml=dbgetv(db,'ml');
plot(ml,maxrange,'*')
dbclose(db)
```

Using these, Natasha and Trilby report:

> Based on statistics from 1993-2001 earthquake data we get the following distance criteria:
>
> | Ml | distance (deg.) |
> |---|---|
> | <=0.0 | <=1 |
> | 0.1-0.5 | <=2 |
> | 0.6-1.2 | <=3 |
> | 1.3-1.6 | <=5 |
> | 1.7-2.1 | <=7 |
> | 2.2-2.5 | <=8 |
> | 2.6-3.0 | <=9 |
> | 3.1-3.9 | <=10 |
> | >=4.0 | whole network |

To avoid a horrendous subsetting expression, let's attempt to simplify this:

Ml == NULL	save entire network
Ml >= 4.0	save entire network
2.0 <= Ml < 4.0	save delta <= 10 degrees
0.5 <= Ml < 2.0	save delta <= 7 degrees
0.5 <= Ml	save delta <= 3 degrees

The first is a safety catch in case a solution escapes magnitude estimation during analysis, or if the magnitude is listed as Mb or MS. The latter cases usually occur for larger events anyway, so it will make sense to save the whole network. This simplified stair-step function makes 82 MB of SEED volumes for 10/19/2001. We have about 86 GB left on /Seis/seg, or 1000 days at that rate.

We could fit our current needs at this rate, however there do appear to be a lot of long, unused data segments for distant stations. Therefore, we will acquiesce to the more complex subsetting function based on Natasha's stairstep.

The resulting subsetting command is

```
trexcerpt -a -v -w '/Seis/seg/wf/%Y/%m/%d/%{sta}.%{chan}.%Y:%j:%H:%M:%S' -j 'ml ==
NULL || ml >= 4 || (ml>=3.0 && distance(origin.lat,origin.lon,site.lat,site.lon)<=10) ||
(ml>=2.5 && distance(origin.lat,origin.lon,site.lat,site.lon)<=9) || (ml>=2.1 && distance(ori-
gin.lat,origin.lon,site.lat,site.lon)<=8) || (ml>=1.6 && distance(origin.lat, origin.lon,
site.lat,site.lon)<=7) || (ml>=1.2 && distance(origin.lat,origin.lon,site.lat,site.lon)<=5) ||
(ml>=0.5 && distance(origin.lat,origin.lon,site.lat,site.lon)<=3) || (ml>=0.1 && distance(ori-
gin.lat,origin.lon,site.lat,site.lon)<=2) || distance(origin.lat,origin.lon,site.lat,site.lon)<=1' -m
event /home/trilby/process/2001_10_19/process_2001_10_19 try 'parrival()-20' 'parrival()
+max(2*(sarrival()-parrival()),phase_arrival(2.8/111.195)-parrival())+30'
```

This is based on a fall-through cascade for the range-vs-magnitude stairstep.

We need to apply this to the currently finished days of analysis that still have data online. In frustration I hand-constructed two files

```
nordic% head -1 list09 list10
==> list09 <==
/Seis/processing/analyzed/2001_09/{analyzed_2001_09_04}:/iwrun/op/db/archive/
{archive_2001_09_04}:/Seis/databases/stations/{master_stations}

==> list10 <==
/Seis/processing/analyzed/2001_10/{analyzed_2001_10_01}:/iwrun/op/db/archive/
{archive_2001_10_01}:/Seis/databases/stations/{master_stations}
nordic%
```

and launched the perl script

```
nordic% cat go
#!/usr/bin/perl

    open( L, "list09" );
    foreach $f ( <L> ) {
        open(D,">mydb" );
        print D "#\nschema css3.0\ndbpath $f";
        close(D);
        system( "cat mydb" );
        system( "trexcerpt -a -v -w '/Seis/seg/wf/%Y/%m/%d/
    %{sta}.%{chan}.%Y:%j:%H:%M:%S' -j 'ml == NULL || ml >= 4 || (ml>=3.0 && dis-
    tance(origin.lat,origin.lon,site.lat,site.lon)<=10) || (ml>=2.5 && distance(origin.lat,ori-
    gin.lon,site.lat,site.lon)<=9) || (ml>=2.1 &&
    distance(origin.lat,origin.lon,site.lat,site.lon)<=8) || (ml>=1.6 && distance(origin.lat,ori-
    gin.lon,site.lat,site.lon)<=7) || (ml>=1.2 && distance(origin.lat,ori-
    gin.lon,site.lat,site.lon)<=5) || (ml>=0.5 &&
    distance(origin.lat,origin.lon,site.lat,site.lon)<=3) || (ml>=0.1 && distance(origin.lat,ori-
    gin.lon,site.lat,site.lon)<=2) || distance(origin.lat,origin.lon,site.lat,site.lon)<=1' -m event
```

```
mydb /Seis/seg/2001/2001_09 'parrival()-20' 'parrival()+max(2*(sarrival()-par-
rival()),phase_arrival(2.8/111.195)-parrival())+30' >> notes 2>&1" );
}
open( L, "list10" );
foreach $f ( <L> ) {
        open(D,">mydb" );
        print D "#\nschema css3.0\ndbpath $f";
        close(D);
        system( "cat mydb" );
        system( "trexcerpt -a -v -w '/Seis/seg/wf/%Y/%m/%d/
%{sta}.%{chan}.%Y:%j:%H:%M:%S' -j 'ml == NULL || ml >= 4 || (ml>=3.0 && dis-
tance(origin.lat,origin.lon,site.lat,site.lon)<=10) || (ml>=2.5 && distance(origin.lat,ori-
gin.lon,site.lat,site.lon)<=9) || (ml>=2.1 &&
distance(origin.lat,origin.lon,site.lat,site.lon)<=8) || (ml>=1.6 && distance(origin.lat,ori-
gin.lon,site.lat,site.lon)<=7) || (ml>=1.2 && distance(origin.lat,ori-
gin.lon,site.lat,site.lon)<=5) || (ml>=0.5 &&
distance(origin.lat,origin.lon,site.lat,site.lon)<=3) || (ml>=0.1 && distance(origin.lat,ori-
gin.lon,site.lat,site.lon)<=2) || distance(origin.lat,origin.lon,site.lat,site.lon)<=1' -m event
mydb /Seis/seg/2001/2001_10 'parrival()-20' 'parrival()+max(2*(sarrival()-par-
rival()),phase_arrival(2.8/111.195)-parrival())+30' >> notes 2>&1" );
}
nordic%
```

Yes, there's probably a nicer way to do it. The 'hard way' is faster here, and there is nothing to be gained by writing a generalized wrapper. This run took about 18 hours on my Sun Blade 1000.

The next step is to incorporate this data segmentation into the routine database-checkin procedure. In other words, we need to launch the trexcerpt command from within the *analysis_control* tk script. Some considerations: ideally I would like each day's events to go into the appropriate month volumes, as above. However, analyst checkin is asynchronous, thus the month volumes will very likely get out of order at some point. That could be solved with an after-the-fact time sorting of the wfdisc table. Also, though, the segmentation takes over 30 minutes, so we would have to set up NFS-based database locking for these waveform databases. That requires the additional small complexity of writing descriptors for each wfdisc table, as well as running a *dbids* server to support NFS locking. The NFS locking will probably slow down the segmentation a bit; also, records may also be not only out of order in day blocks, but intermingled at the station-by-station level. Finally, if and when there are neighboring days that have overlapping waveform segments, the 'append' option in our trexcerpt command (which was necessary during testing due to occasional wfdisc problems) has the potential to make a thorough mess if two users are trying to write to the same wfdisc file. No database synchronization anywhere will solve that. Since we are already storing analyzed parametric data in day volumes, acquiescing to having the coalescence into month volumes be at a different layer, there is not much sacrifice in doing the same with the wfdiscs for segmented data. The build of day-volumes into month-volumes can be done at a later stage. In order to take advantage of analyst switching between operational and backup waveform databases, the segmentation will proceed from the database directly in the /home/ANALYST-NAME/process directory structure. This will introduce the requirement that the analyst not touch

their processed database for an hour or so after processing is finished. Because the data segmentation takes a significant amount of time, in order to allow the analyst to check out another database we will have to launch the *trexcerpt* in the background (to return control back to *analysis_control* immediately) and with *nohup* (to allow the analyst to log out if they desire, also to protect the procedure against accidental log-out etc.). This approach introduces the requirement that the machine not die or be rebooted for an hour or so after check-in, in order for the segmentation to finish. Finally, we need to worry about tracking the results of the segmentation and correcting for errors. That first-order problem is important, but will be deferred until the zeroth-order functionality itself is implemented.[1]

The test code which performs this data segmentation from tcl is here:

```
nordic% cat trex.tcl
set dbin mydb
#set dbout /Seis/seg/2001/2001_09
set dbout moo
set notes testnotes
set wf_pattern "/Seis/seg/wf/%Y/%m/%d/%{sta}.%{chan}.%Y:%j:%H:%M:%S"
set distance_pattern ""; append distance_pattern \
  "ml == NULL || " \
  "ml >= 4 || " \
  "(ml>=3.0 && distance(origin.lat,origin.lon,site.lat,site.lon)<=10) || "
  "(ml>=2.5 && distance(origin.lat,origin.lon,site.lat,site.lon)<=9) || " \
  "(ml>=2.1 && distance(origin.lat,origin.lon,site.lat,site.lon)<=8) || " \
  "(ml>=1.6 && distance(origin.lat,origin.lon,site.lat,site.lon)<=7) || " \
  "(ml>=1.2 && distance(origin.lat,origin.lon,site.lat,site.lon)<=5) || " \
  "(ml>=0.5 && distance(origin.lat,origin.lon,site.lat,site.lon)<=3) || " \
  "(ml>=0.1 && distance(origin.lat,origin.lon,site.lat,site.lon)<=2) || " \
  "distance(origin.lat,origin.lon,site.lat,site.lon)<=1"

set pre_p_pad_sec 20
set post_pad_sec 30
set vel_cutoff_kmps 2.8
set segment_start "parrival()-$pre_p_pad_sec"
set segment_end "parrival()+max(2*(sarrival()-parrival()),phase_arrival($veloff_kmps/
111.195)-parrival())+$post_pad_sec"

set tmpscript /tmp/ac_[pid]_[lindex [array get env USER] 1]
set scriptfd [open $tmpscript w]
puts $scriptfd "trexcerpt -a -vv -w '$wf_pattern' -j '$distance_pattern' -m t $dbin $dbout
'$segment_start' '$segment_end'"
close $scriptfd
set result [catch "exec nohup sh $tmpscript >& $notes &" trexcerpt_return]
#exec /bin/rm $tmpscript
```

1. Complexity must be managed in layers.

nordic%

I built these ideas into the *analysis_control*, of course with more reasonable pathnames plus reliance on aeic_rtsys etc. I had to sidestep the fact that aeic_rtsys refers, correctly, to month volumes but at this intermediate stage we have to excerpt to day volumes. Also, analysis_control has DBLOCKS set in its environment to 1, in order to coordinate interaction with the tracking database. Unfortunately this locking also blocks addition of segmented data, thus we have to restructure the locking mechanism for the tracking database in order to proceed (change to dbdescriptor-specified locks, using "1" mechanism instead of nfs to avoid having to set up a dbidserver for this). Note that since all *analysis_control* operation is currently launched from nordic, all data segmentation will be done from nordic. Ultimately, we will want to switch the aeic_analysis database to nfs locking.

In the Tcl (actually dbwish8) code above, I wrote the complex *trexcerpt* expression to a temporary file to avoid dealing with all the interpreter-escape issues going through tcl, catch, exec, and nohup. As a first step towards recovery after error, I copied the script file to the users' processing directory so it can be re-run if necessary [another advantage of segmenting waveform data into day-volumes as an intermediate step]. Finally, I directed the notes to trexcerpt_notes in the users' process directory.

Recovery of Missing Data Segments from Continuous Data

The final part of segmented data handling is extraction of data segments for events from the continuous data tapes. This will be an extensive project, since we will have to recover waveforms from mid-November 1999 through Sept. 3, 2001. We will probably want to purchase a tape stacker for the unloading of continuous data, unload 10 or so days at a time, and perform the data extraction. The wfdisc tables themselves are still on-line in /iwrun/op/db/archive.

As an exercise, we will re-extract the waveforms for Sept. 1-3, 2001. That will create a complete database for September 2001, allowing quality-control to begin on it. The continuous waveform data for this time period are on Exabyte tape labelled 2001-244, 245, and 246. There is still room in /iwrun/op/archive_wf for these days, so Mitch unloaded them there. This means we can use the wfdisc tables in /iwrun/op/db/archive in their present form. In the future, we will have to unload days of continuous data into a temporary area; copy the appropriate wfdisc tables from /iwrun/op/db/archive; and re-set the 'dir' field in those wfdisc tables to point to the waveforms.

For now [I don't recommend this approach for routine use], I went to Trilby and Ed's process directories, checked the descriptor files to make sure they were pointing to the 'op' databases, copied and correctly edited one of the trexcerpt_scripts, launched a dbwish8 shell, and cut-and-pasted the correct command from *analysis_control*:

```
nordic% cat /home/trilby/process/2001_09_02/trexcerpt_script
trexcerpt -a -vv -w '/Seis/seg/wf/%Y/%m/%d/%{sta}.%{chan}.%Y:%j:%H:%M:%S' -j 'ml
== NULL || ml >= 4 || (ml>=3.0 && distance(origin.lat,origin.lon,site.lat,site.lon)<=10) ||
(ml>=2.5 && distance(origin.lat,origin.lon,site.lat,site.lon)<=9) || (ml>=2.1 && distance( ori-
```

```
gin.lat, origin.lon,site.lat,site.lon)<=8) || (ml>=1.6 && distance(origin.lat, origin.lon,
site.lat,site.lon)<=7) || (ml>=1.2 && distance(origin.lat,origin.lon,site.lat,site.lon)<=5) ||
(ml>=0.5 && distance(origin.lat,origin.lon,site.lat,site.lon)<=3) || (ml>=0.1 && distance(ori-
gin.lat,origin.lon,site.lat,site.lon)<=2) || distance(origin.lat,origin.lon,site.lat,site.lon)<=1' -m
event /home/trilby/process/2001_09_02/process_2001_09_02 /Seis/seg/2001/2001_09_02
'parrival()-20' 'parrival()+max(2*(sarrival()-parrival()),phase_arrival(2.8/111.195)-par-
rival())+30'
nordic%

nordic% dbwish8
% set result [catch "exec nohup sh trexcerpt_script >& trexcerpt_notes &" trexcerpt_return]
```

After these finish, we need to recombine the September databases into one month volume:

```
nordic% cd /Seis/seg/2001
nordic% mv 2001_09.wfdisc 2001_09tmp.wfdisc
nordic% mv 2001_09.lastid 2001_09tmp.lastid
nordic% aeic_dbconcat_event -x 2001_09_01 2001_09_02 2001_09_03 2001_09tmp
2001_09
nordic% rm 2001_09_01* 2001_09_02* 2001_09_03* 2001_09tmp*
```

Similarly we need to combine the parametric data into month volumes, a process which will eventually be automated:

```
nordic% cd /Seis/processing/analyzed
nordic% foreach d ( 2000* 2001_0* )
foreach? echo $d
foreach? cd /Seis/processing/analyzed/$d
foreach? aeic_dbconcat_event 'ls -1 *.origin | sed -e 's/.origin//'' analyzed_$d |& tee /home/
kent/work/segdata/concat_notes
foreach? end
```

The only problem is a profusion of wfmeas entries for arrivals that no longer exist. The aeic_dbconcat_event relabelled them to have arid -1. These will be cleaned up at the QC stage. Out of paranoia, rather than removing the day volumes I moved them[1]:

```
nordic% cd /Seis/processing/analyzed/
nordic% mkdir /Seis/space/catalog_staging/analyzed_bak
nordic% mv 200[01]_??/analyzed_????_??_??.* /Seis/space/catalog_staging/analyzed_bak
```

Finally, we will create a database to use as a base for QC work for September, 2001. At the moment this is done by hand, in /Seis/space/2001_09. The database is qcwork_2001_09.

Similarly, by November 2, 2001 the analysis is finished for all of October. Even though we still have to do the assembly by hand, we can immediately make a QC database for October, which I

have done in /Seis/space/2001_10/qcwork_2001_10[1]. This puts us on track to keep up with quality-control work as the analysis is finished.

1. Note that I moved them very far away from /Seis/processing/analyzed. In order for an automated system to work well, there must be an active effort to keep everything as clean, neat and orderly as possible. Things that are recognizable by humans (such as names for backup copies and temporary directories that you come up with on the fly), but different from some established pattern in a directory, wreak havoc with any attempts to use the computer to automate tasks. In other words, lack of organization scuttles one's ability to use the computer for its strengths. If there's any doubt about this, see the preceding weeks of effort to clean up idiosyncrasies in the old segmented data directories. Sometimes people will pollute clean, organized directories for other reasons, for example when there's lots of free disk space remaining in the clean directory, or when copying to a different directory would involve a time-consuming transfer of data across the network. In the long run, however, it is a far better deal to trade that disk space and trade that copying time in order to save weeks of developer time writing unnecessarily complex code and cleaning up horrendous messes.
1. This burst of efficiency, immediately joining the October 2001 day volumes into a month volume, produced an unexpected glitch in the generation of the last weekly report for October. Previously, that software had assumed the existence of day volumes. This will have to be fixed when the weekly-report software is revisited.

Appendix A: Summary of translations applied to segmented waveforms

Table 1:

ACHE_she => ACH_SHE	ACHE_shz => ACH_SHE	ACHN_shn => ACH_SHN	ACHN_shz => ACH_SHN	ACH_SHE => ACH_SHE
ACH_SHN => ACH_SHN	ACH_SHZ => ACH_SHZ	ACH_she => ACH_SHE	ACH_shn => ACH_SHN	ACH_shz => ACH_SHZ
ADAG_SHZ => ADAG_SHZ	ADBE_bhe => ADK_BHE	ADBN_bhn => ADK_BHN	ADB_bhz => ADK_BHZ	ADK-_shz => ADK_SLZ
ADK-_slz => ADK_SLZ	ADK_BHE => ADK_BHE	ADK_BHN => ADK_BHN	ADK_BHZ => ADK_BHZ	ADK_SHZ => ADK_SHZ
ADK_shz => ADK_SHZ	ADK_spz => ADK_SHZ	AHB_SHZ => AHB_SHZ	AHB_shz => AHB_SHZ	AIB_BHE => AIB_BHE
AIB_BHN => AIB_BHN	AIB_BHZ => AIB_BHZ	AJAX_SHZ => AJAX_SHZ	AK1_shz => AK1_SHZ	AK2_SHZ => AK2_SHZ
AK2_shz => AK2_SHZ	AK3_shz => AK3_SHZ	AK4_SHZ => AK4_SHZ	AK4_ppp => AK4_BDF	AK4_shz => AK4_SHZ
AK4_slz => AK4_slz	AK5_SHZ => AK5_SHZ	AK5_ppp => AK5_BDF	AK5_shz => AK5_SHZ	AKM_shp => AKM_BDF
AKM_shz => AKM_SHZ	AKSE_she => AKS_SHE	AKSE_shz => AKS_SHE	AKSN_shn => AKS_SHN	AKSN_shz => AKS_SHN
AKS_SHE => AKS_SHE	AKS_SHN => AKS_SHN	AKS_SHZ => AKS_SHZ	AKS_she => AKS_SHE	AKS_shn => AKS_SHN
AKS_shz => AKS_SHZ	AKT1_shz => AKT1_SHZ	AKT2_shz => AKT2_SHZ	AKT3_shz => AKT3_SHZ	AKT4_shz => AKT4_SHZ
AKT5_shz => AKT5_SHZ	AKTE_bhe => AKT_BHE	AKTE_she => AKT_BHE	AKTE_shz => AKT_BHE	AKTM_shz => AKTM_SHZ
AKTN_bhn => AKT_BHN	AKTN_shn => AKT_BHN	AKTN_shz => AKT_BHN	AKTZ_shz => AKTZ_SHZ	AKT_BHE => AKT_BHE
AKT_BHN => AKT_BHN	AKT_BHZ => AKT_BHZ	AKT_bhe => AKT_BHE	AKT_bhn => AKT_BHN	AKT_bhz => AKT_BHZ
AKT_she => AKT_SHE	AKT_shn => AKT_SHN	AKT_shz => AKT_SHZ	AKV_SHZ => AKV_SHZ	AKV_shz => AKV_SHZ
AL1E_lpe => AL1_LHE	AL1N_lpn => AL1_LHN	AL1_lpz => AL1_LHZ	AL2E_lpe => AL2_LHE	AL2N_lpn => AL2_LHN
AL2_lpz => AL2_LHZ	AL3E_lpe => AL3_LHE	AL3N_lpn => AL3_LHN	AL3_lpz => AL3_LHZ	AL4E_lpe => AL4_LHE
AL4N_lpn => AL4_LHN	AL4_lpz => AL4_LHZ	AL5E_lpe => AL5_LHE	AL5N_lpn => AL5_LHN	AL5_lpz => AL5_LHZ
AL6E_lpe => AL6_LHE	AL6N_lpn => AL6_LHN	AL6_lpz => AL6_LHZ	AL7E_lpe => AL7_LHE	AL7N_lpn => AL7_LHN
AL7_lpz => AL7_LHZ	ALL1_LHE => ALL1_LHE	ALL1_LHN => ALL1_LHN	ALL1_LHZ => ALL1_LHZ	ALL2_LHE => ALL2_LHE
ALL2_LHN => ALL2_LHN	ALL2_LHZ => ALL2_LHZ	ALL3_LHE => ALL3_LHE	ALL3_LHN => ALL3_LHN	ALL3_LHZ => ALL3_LHZ
ALL4_LHE => ALL4_LHE	ALL4_LHN => ALL4_LHN	ALL4_LHZ => ALL4_LHZ	ALL5_LHE => ALL5_LHE	ALL5_LHN => ALL5_LHN
ALL5_LHZ => ALL5_LHZ	ALL6_LHE => ALL6_LHE	ALL6_LHN => ALL6_LHN	ALL6_LHZ => ALL6_LHZ	ALL7_LHE => ALL7_LHE
ALL7_LHN => ALL7_LHN	ALL7_LHZ => ALL7_LHZ	ANCK_SHZ => ANCK_SHZ	ANCK_shz => ANCK_SHZ	ANIA_SHZ => ANIA_SHZ
ANIA_shz => ANIA_SHZ	ANM_SHZ => ANM_SHZ	ANNE_SHZ => ANNE_SHZ	ANNE_shz => ANNE_SHZ	ANNW_SHZ => ANNW_SHZ
ANNW_shz => ANNW_SHZ	ANON_SHE => ANON_SHE	ANON_SHN => ANON_SHN	ANON_SHZ => ANON_SHZ	ANPB_SHZ => ANPB_SHZ
ANPB_shz => ANPB_SHZ	ANPK_SHZ => ANPK_SHZ	ANPK_shz => ANPK_SHZ	ANSL_SHE => ANSL_SHE	ANSL_SHN => ANSL_SHN
ANSL_SHZ => ANSL_SHZ	ANSL_she => ANSL_SHE	ANSL_shn => ANSL_SHN	ANSL_shz => ANSL_SHZ	APEX_BDF => APEX_BDF
ASR_EHZ => ASR_EHZ	ATTU_BHE => ATTU_BHE	ATTU_BHN => ATTU_BHN	ATTU_BHZ => ATTU_BHZ	AUBE_bhe => AUB_BHE
AUBN_bhn => AUB_BHN	AUB_bhe => AUB_BHE	AUB_bhn => AUB_BHN	AUB_bhz => AUB_BHZ	AUC_SHZ => AUC_SHZ
AUC_shz => AUC_SHZ	AUD-_shz => AUD_SLZ	AUD-_slz => AUD_SLZ	AUD-_tcc => AUD_SLZ	AUD_SHZ => AUD_SHZ
AUD_SLZ => AUD_SHZ	AUD_shz => AUD_SHZ	AUD_tcc => AUD_SHZ	AUEP_ppp => AUE_BDF	AUEP_shz => AUE_BDF
AUE_BDF => AUE_BDF	AUE_SHP => AUE_BDF	AUE_SHZ => AUE_SHZ	AUE_shp => AUE_BDF	AUE_shz => AUE_SHZ
AUE_spz => AUE_SHZ	AUH_SHZ => AUH_SHZ	AUH_shz => AUH_SHZ	AUH_spz => AUH_SHZ	AUIE_she => AUI_SHE
AUIE_shz => AUI_SHE	AUIE_spz => AUI_SHE	AUIN_shn => AUI_SHN	AUIN_shz => AUI_SHN	AUIN_spz => AUI_SHN
AUI_SHE => AUI_SHE	AUI_SHN => AUI_SHN	AUI_SHZ => AUI_SHZ	AUI_she => AUI_SHE	AUI_shn => AUI_SHN

Table 1:

AUI_shz => AUI_SHZ	AUI_spz => AUI_SHZ	AUL_BHE => AUL_BHE	AUL_BHN => AUL_BHN	AUL_BHZ => AUL_BHZ
AUL_SHZ => AUL_SHZ	AUL_shz => AUL_SHZ	AUL_spz => AUL_SHZ	AUP_SHZ => AUP_SHZ	AUP_shz => AUP_SHZ
AUP_spz => AUP_SHZ	AUR_SHZ => AUR_SHZ	AUR_shz => AUR_SHZ	AUS_SHZ => AUS_SHZ	AUS_shz => AUS_SHZ
AUW_SHZ => AUW_SHZ	AUW_shz => AUW_SHZ	AUW_spz => AUW_SHZ	AVN_SHZ => AVN_SHZ	AVN_shz => AVN_SHZ
AVN_spz => AVN_SHZ	BALA_shz => BALA_SHZ	BALL_BDF => BALL_BDF	BALL_spz => BALL_SHZ	BAL_SHZ => BAL_SHZ
BAL_shz => BAL_SHZ	BAL_spz => BAL_SHZ	BBB_BHE => BBB_BHE	BBB_BHN => BBB_BHN	BBB_BHZ => BBB_BHZ
BC01_SHZ => BC01_SHZ	BC02_SHZ => BC02_SHZ	BC03_SHZ => BC03_SHZ	BC04_SHZ => BC04_SHZ	BC05_SHZ => BC05_SHZ
BC3_BHZ => BC3_BHZ	BC3_spz => BC3_BHZ	BCP_SHZ => BCP_SHZ	BCP_shz => BCP_SHZ	BGL_SHZ => BGL_SHZ
BGL_shz => BGL_SHZ	BGL_spz => BGL_SHZ	BGM_SHZ => BGM_SHZ	BGM_shz => BGM_SHZ	BGM_spz => BGM_SHZ
BGR_SHZ => BGR_SHZ	BGR_shz => BGR_SHZ	BI0_shz => BI0_SHZ	BILL_BHE => BILL_BHE	BILL_BHN => BILL_BHN
BILL_BHZ => BILL_BHZ	BKG#_shz => BKG_SHZ	BKG_SHZ => BKG_SHZ	BKG_shz => BKG_SHZ	BLDY_SHZ => BLDY_SHZ
BLDY_shz => BLDY_SHZ	BLGA_SHE => BLGA_SHE	BLGA_SHN => BLGA_SHN	BLGA_SHZ => BLGA_SHZ	BLHA_SHZ => BLHA_SHZ
BLHA_shz => BLHA_SHZ	BLH_spz => BLH_SHZ	BM01_SHZ => BM01_SHZ	BM02_SHZ => BM02_SHZ	BM03_SHZ => BM03_SHZ
BM04_SHZ => BM04_SHZ	BM05_SHZ => BM05_SHZ	BM3_SHZ => BM3_SHZ	BM3_spz => BM3_SHZ	BMR_BHE => BMR_BHE
BMR_BHN => BMR_BHN	BMR_BHZ => BMR_BHZ	BMR_HHE => BMR_HHE	BMR_HHN => BMR_HHN	BMR_HHZ => BMR_HHZ
BNAB_EHZ => BNAB_EHZ	BNB_EHZ => BNB_EHZ	BRLE_shz => BRLK_SHE	BRLK_SHZ => BRLK_SHZ	BRLK_shz => BRLK_SHZ
BRLK_spz => BRLK_SHZ	BRLN_shz => BRLK_SHN	BRPK_SHZ => BRPK_SHZ	BRPK_shz => BRPK_SHZ	BRW_SHZ => BRW_SHZ
BRW_shz => BRW_SHZ	BRW_spz => BRW_SHZ	BWN_SHZ => BWN_SHZ	BWN_shn => BWN_SHZ	BWN_shz => BWN_SHZ
BWN_spz => BWN_SHZ	CAHL_SHZ => CAHL_SHZ	CAHL_shz => CAHL_SHZ	CBYE_bhe => CBY_BHE	CBYN_bhn => CBY_BHN
CBYZ_bhz => CBY_BHZ	CBY_bhz => CBY_BHZ	CCB_SHZ => CCB_SHZ	CCB_shz => CCB_SHZ	CCB_spz => CCB_SHZ
CDD_SHZ => CDD_SHZ	CDD_shz => CDD_SHZ	CDD_spz => CDD_SHZ	CEBE_bhe => CEB_BHE	CEBN_bhn => CEB_BHN
CEBZ_bhz => CEB_BHZ	CEB_BHE => CEB_BHE	CEB_BHN => CEB_BHN	CEB_BHZ => CEB_BHZ	CEB_bhz => CEB_BHZ
CFI_SHZ => CFI_SHZ	CFI_shz => CFI_SHZ	CFPZ_shz => CFP_SHZ	CFP_SHZ => CFP_SHZ	CFP_shz => CFP_SHZ
CGL_SHZ => CGL_SHZ	CGL_shz => CGL_SHZ	CGL_spz => CGL_SHZ	CHNH_ => CHN_SHH	CHNH_shh => CHN_SHH
CHNH_shz => CHN_SHH	CHN_ => CHN_SHZ	CHN_shz => CHN_SHZ	CHSA_BHZ => CHSA_BHZ	CHXT_shz => CHXT_SHZ
CHX_SHZ => CHX_SHZ	CHX_shz => CHX_SHZ	CIGO_BDF => CIGO_BDF	CKL_SHZ => CKL_SHZ	CKL_shz => CKL_SHZ
CKL_spz => CKL_SHZ	CKN#_shz => CKN_SHZ	CKN_SHZ => CKN_SHZ	CKN_shz => CKN_SHZ	CKT#_shz => CKT_SHZ
CKT_SHZ => CKT_SHZ	CKT_shz => CKT_SHZ	CNP_SHZ => CNP_SHZ	CNP_shz => CNP_SHZ	CNP_spz => CNP_SHZ
CNTC_SHZ => CNTC_SHZ	CNTC_shz => CNTC_SHZ	COLA_BH1 => COLA_BH1	COLA_BH2 => COLA_BH2	COLA_BHZ => COLA_BHZ
COLA_HLE => COLA_HLE	COLA_HLN => COLA_HLN	COLA_HLZ => COLA_HLZ	COLA_LH1 => COLA_LH1	COLA_LH2 => COLA_LH2
COLA_LHZ => COLA_LHZ	COLA_LLE => COLA_LLE	COLA_LLN => COLA_LLN	COLA_LLZ => COLA_LLZ	COLA_SHE => COLA_SHE
COLA_SHN => COLA_SHN	COLA_SHZ => COLA_SHZ	COLA_UHZ => COLA_UHZ	COLA_VH1 => COLA_VH1	COLA_VH2 => COLA_VH2
COLA_VHZ => COLA_VHZ	COLA_VK1 => COLA_VK1	COL_BHE => COL_BHE	COL_BHN => COL_BHN	COL_BHZ => COL_BHZ
CP2T_SHZ => CP2T_SHZ	CP2T_shz => CP2T_SHZ	CP2_SHZ => CP2_SHZ	CP2_shz => CP2_SHZ	CPAE_shz => CPA_SHE
CPAN_shz => CPA_SHN	CPAP_shz => CPA_BDF	CPA_shz => CPA_SHZ	CPK-_shz => CPK_SLZ	CPKE_shz => CPK_SHE
CPKN_shz => CPK_SHN	CPK_shz => CPK_SHZ	CRBE_bhe => CRB_BHE	CRBE_she => CRB_BHE	CRBN_bhn => CRB_BHN
CRBN_shn => CRB_BHN	CRB_BHE => CRB_BHZ	CRB_BHN => CRB_BHZ	CRB_BHZ => CRB_BHZ	CRB_bhz => CRB_BHZ
CRB_shz => CRB_BHZ	CRPE_she => CRP_SHE	CRPE_shn => CRP_SHE	CRPE_shz => CRP_SHE	CRPN_she => CRP_SHN
CRPN_shn => CRP_SHN	CRPN_shz => CRP_SHN	CRPP_BDF => CRP_BDF	CRPP_ppp => CRP_BDF	CRPP_shz => CRP_BDF

Table 1:

CRPT_SHZ => CRPT_SHZ	CRPT_shz => CRPT_SHZ	CRP_SHE => CRP_SHE	CRP_SHN => CRP_SHN	CRP_SHZ => CRP_SHZ
CRP_she => CRP_SHE	CRP_shn => CRP_SHN	CRP_shz => CRP_SHZ	CRP_spz => CRP_SHZ	CRQ_SHZ => CRQ_SHZ
CRQ_shz => CRQ_SHZ	CRQ_spz => CRQ_SHZ	CTG_SHZ => CTG_SHZ	CTG_shz => CTG_SHZ	CTG_spz => CTG_SHZ
CUT_SHZ => CUT_SHZ	CUT_shz => CUT_SHZ	CUT_spz => CUT_SHZ	CVA_SHZ => CVA_SHZ	CVA_shn => CVA_SHZ
CVA_shz => CVA_SHZ	CVA_zhs => CVA_SHZ	CYK_SHZ => CYK_SHZ	CYK_shz => CYK_SHZ	DAWY_BHE => DAWY_BHE
DAWY_BHN => DAWY_BHN	DAWY_BHZ => DAWY_BHZ	DDM_SHZ => DDM_SHZ	DDM_shz => DDM_SHZ	DDM_spz => DDM_SHZ
DEER_BDF => DEER_BDF	DFRP_shz => DFR_BDF	DFR_SHZ => DFR_SHZ	DFR_shz => DFR_SHZ	DFR_spz => DFR_SHZ
DHY_SHZ => DHY_SHZ	DHY_shz => DHY_SHZ	DIV_BHE => DIV_BHE	DIV_BHN => DIV_BHN	DIV_BHZ => DIV_BHZ
DIV_HHE => DIV_HHE	DIV_HHN => DIV_HHN	DIV_HHZ => DIV_HHZ	DJE_shz => DJE_SHZ	DLBC_BHE => DLBC_BHE
DLBC_BHN => DLBC_BHN	DLBC_BHZ => DLBC_BHZ	DLG_spz => DLG_SHZ	DMW_shz => DMW_SHZ	DMW_spz => DMW_SHZ
DOL_SHZ => DOL_SHZ	DOL_shz => DOL_SHZ	DOT_SHZ => DOT_SHZ	DOT_shz => DOT_SHZ	DOT_spz => DOT_SHZ
DRE_shz => DRE_SHZ	DRIA_shz => DRIA_SHZ	DRR3_SHZ => DRR3_SHZ	DRR3_shz => DRR3_SHZ	DRR_spz => DRR_SHZ
DT1A_shz => DT1A_SHZ	DT1_SHZ => DT1_SHZ	DT1_shz => DT1_SHZ	DT2_spz => DT2_SHZ	DTN_SHZ => DTN_SHZ
DTN_shz => DTN_SHZ	DTN_spz => DTN_SHZ	EAF#_shz => EAF_SHZ	EAF_SHZ => EAF_SHZ	EAF_shz => EAF_SHZ
EKS2_BHE => EKS2_BHE	EKS2_BHN => EKS2_BHN	EKS2_BHZ => EKS2_BHZ	EKS2_LHE => EKS2_LHE	EKS2_LHN => EKS2_LHN
EKS2_LHZ => EKS2_LHZ	ERM_BHE => ERM_BHE	ERM_BHN => ERM_BHN	ERM_BHZ => ERM_BHZ	ET0_spz => ET0_SHZ
ETKA_SHZ => ETKA_SHZ	FB2_spz => FB2_SHZ	FBA_SHZ => FBA_SHZ	FBA_shz => FBA_SHZ	FBA_spz => FBA_SHZ
FCC_BHE => FCC_BHE	FCC_BHN => FCC_BHN	FCC_BHZ => FCC_BHZ	FIBE_bhe => FIB_BHE	FIBN_bhn => FIBN_BHN
FIB_BHE => FIB_BHE	FIB_BHN => FIB_BHN	FIB_BHZ => FIB_BHZ	FIB_bhe => FIB_BHE	FIB_bhn => FIB_BHN
FIB_bhz => FIB_BHZ	FID_SHZ => FID_SHZ	FID_shz => FID_SHZ	FID_spz => FID_SHZ	FL2_EHZ => FL2_EHZ
FSB_EHZ => FSB_EHZ	FX01_SHZ => FX01_SHZ	FX1_SHZ => FX1_SHZ	FX1_spz => FX1_SHZ	FXBB_LHE => FXBB_LHE
FXBB_LHN => FXBB_LHN	FXBB_LHZ => FXBB_LHZ	FXBB_SHE => FXBB_SHE	FXBB_SHN => FXBB_SHN	FXBB_SHZ => FXBB_SHZ
FYU_SHZ => FYU_SHZ	FYU_shz => FYU_SHZ	FYU_slz => FYU_slz	FYU_spz => FYU_SHZ	GAR_spz => GAR_SHZ
GCSA_BHE => GCSA_BHE	GCSA_BHN => GCSA_BHN	GCSA_BHZ => GCSA_BHZ	GEEE_tcc => GEEE_TIME	GHO_SHZ => GHO_SHZ
GHO_shz => GHO_SHZ	GHO_spz => GHO_SHZ	GIBE_bhe => GIB_BHE	GIBE_she => GIB_BHE	GIBN_bhn => GIB_BHN
GIBN_shn => GIB_BHN	GIB_BHE => GIB_BHE	GIB_BHN => GIB_BHN	GIB_BHZ => GIB_BHZ	GIB_bhe => GIB_BHE
GIB_bhn => GIB_BHN	GIB_bhz => GIB_BHZ	GIB_shz => GIB_SHZ	GI_shz => GI_SHZ	GLBE_she => GLB_SHE
GLBE_shz => GLB_SHE	GLBE_spz => GLB_SHE	GLBN_shn => GLB_SHN	GLBN_shz => GLB_SHN	GLBN_spz => GLB_SHN
GLB_SHE => GLB_SHE	GLB_SHN => GLB_SHN	GLB_SHZ => GLB_SHZ	GLB_she => GLB_SHE	GLB_shn => GLB_SHN
GLB_shz => GLB_SHZ	GLB_spz => GLB_SHZ	GLI_SHZ => GLI_SHZ	GLI_shz => GLI_SHZ	GLI_spz => GLI_SHZ
GLM_SHZ => GLM_SHZ	GLM_shz => GLM_SHZ	GLM_spz => GLM_SHZ	GLN_shz => GLN_SHZ	GLN_spz => GLN_SHZ
GOEE_TIME => GOEE_TIME	GOEE_she => GOEE_TIME	GOEE_shz => GOEE_TIME	GOEE_tcc => GOEE_TIME	GOES_sht => GOES_TIME
GOES_tcc => GOES_TIME	GOU#_shz => GOU_SHZ	GOUE_she => GOU_SHE	GOUE_shz => GOU_SHE	GOUN_shn => GOU_SHN
GOUN_shz => GOU_SHN	GOUP_shz => GOU_BDF	GOU_SHE => GOU_SHE	GOU_SHN => GOU_SHN	GOU_SHZ => GOU_SHZ
GOU_she => GOU_SHE	GOU_shn => GOU_SHN	GOU_shz => GOU_SHZ	GSCK_SHZ => GSCK_SHZ	GSIG_SHZ => GSIG_SHZ
GSMY_SHZ => GSMY_SHZ	GSSP_SHZ => GSSP_SHZ	GSTD_SHE => GSTD_SHE	GSTD_SHN => GSTD_SHN	GSTD_SHZ => GSTD_SHZ
GSTR_SHZ => GSTR_SHZ	GUE#_she => GOU_SHE	GUE#_shz => GOU_SHE	GUN#_shn => GOU_SHN	GUN#_shz => GOU_SHN
HAG_SHZ => HAG_SHZ	HAG_shz => HAG_SHZ	HDA_SHZ => HDA_SHZ	HDA_shz => HDA_SHZ	HDA_spz => HDA_SHZ
HIN_SHZ => HIN_SHZ	HIN_shz => HIN_SHZ	HMT_SHZ => HMT_SHZ	HMT_shz => HMT_SHZ	HOMP_shz => HOM_BDF

Table 1:

HOM_SHZ => HOM_SHZ	HOM_shz => HOM_SHZ	HOM_spz => HOM_SHZ	HQN_shz => HQN_SHZ	HSB_SHZ => HSB_SHZ
HSB_shz => HSB_SHZ	HUR_SHZ => HUR_SHZ	HUR_shz => HUR_SHZ	HUR_spz => HUR_SHZ	HYT_EHZ => HYT_EHZ
I59H1_BDF => I59H1_BDF	I59H2_BDF => I59H2_BDF	I59H3_BDF => I59H3_BDF	I59H4_BDF => I59H4_BDF	IL01_SHZ => IL01_SHZ
IL02_SHZ => IL02_SHZ	IL03_SHZ => IL03_SHZ	IL04_SHZ => IL04_SHZ	IL05_SHZ => IL05_SHZ	IL06_SHZ => IL06_SHZ
IL07_SHZ => IL07_SHZ	IL08_SHZ => IL08_SHZ	IL09_SHZ => IL09_SHZ	IL10_SHZ => IL10_SHZ	IL11_SHZ => IL11_SHZ
IL12_SHZ => IL12_SHZ	IL13_SHZ => IL13_SHZ	IL14_SHZ => IL14_SHZ	IL15_SHZ => IL15_SHZ	IL16_SHZ => IL16_SHZ
IL17_SHZ => IL17_SHZ	IL18_SHZ => IL18_SHZ	IL19_SHZ => IL19_SHZ	IL1_SHZ => IL1_SHZ	IL1_spz => IL1_SHZ
ILBB_LHE => ILBB_LHE	ILBB_LHN => ILBB_LHN	ILBB_LHZ => ILBB_LHZ	ILBB_SHE => ILBB_SHE	ILBB_SHN => ILBB_SHN
ILBB_SHZ => ILBB_SHZ	ILBE_spe => ILB_SHE	ILBN_spn => ILB_SHN	ILB_SHE => ILB_SHE	ILB_SHN => ILB_SHN
ILB_SHZ => ILB_SHZ	ILB_spz => ILB_SHZ	ILI_SHZ => ILI_SHZ	ILI_shz => ILI_SHZ	ILI_spz => ILI_SHZ
ILS_SHZ => ILS_SHZ	ILS_shz => ILS_SHZ	ILW_SHZ => ILW_SHZ	ILW_shz => ILW_SHZ	IM01_SHZ => IM01_SHZ
IM02_SHZ => IM02_SHZ	IM03_SHZ => IM03_SHZ	IM04_SHZ => IM04_SHZ	IM05_SHZ => IM05_SHZ	IM3_SHZ => IM3_SHZ
IM3_spz => IM3_SHZ	IMA-_shz => IMA_SLZ	IMA-_slz => IMA_SLZ	IMA-_spz => IMA_SLZ	IMAI_SHZ => IMAI_SHZ
IMAI_shz => IMAI_SHZ	IMA_SHZ => IMA_SHZ	IMA_SLZ => IMA_SHZ	IMA_shz => IMA_SHZ	IMA_spz => IMA_SHZ
INE_SHZ => INE_SHZ	INE_shn => INE_SHZ	INE_shz => INE_SHZ	INK_BHE => INK_BHE	INK_BHN => INK_BHN
INK_BHZ => INK_BHZ	INW_shz => INW_SHZ	IRGE_ => IRGE_TIME	IRGE_TIME => IRGE_TIME	IRGE_shv => IRGE_TIME
IRGE_shz => IRGE_TIME	IRGE_tcc => IRGE_TIME	IRGH_TIME => IRGH_TIME	IRGH_sht => IRGH_TIME	IRGH_tcc => IRGH_TIME
IRH_TIME => IRH_TIME	IRH_shz => IRH_TIME	IRH_spz => IRH_TIME	IRIG_TIME => IRIG_TIME	IRIG_sht => IRIG_TIME
IRIG_tcc => IRIG_TIME	ISNN_SHZ => ISNN_SHZ	ISNN_shz => ISNN_SHZ	ISTK_SHZ => ISTK_SHZ	ISTK_shz => ISTK_SHZ
IVEE_she => IVE_SHE	IVEE_shz => IVE_SHE	IVEN_shn => IVE_SHN	IVEN_shz => IVE_SHN	IVE_SHE => IVE_SHE
IVE_SHN => IVE_SHN	IVE_SHZ => IVE_SHZ	IVE_she => IVE_SHE	IVE_shn => IVE_SHN	IVE_shz => IVE_SHZ
IVS_SHZ => IVS_SHZ	IVS_shz => IVS_SHZ	IVW_shz => IVW_SHZ	JIS_BHE => JIS_BHE	JIS_BHN => JIS_BHN
JIS_BHZ => JIS_BHZ	JPK_SHZ => JPK_SHZ	JPK_shz => JPK_SHZ	JUN_EHZ => JUN_EHZ	K01_she => K01_SHE
K01_shl => K01_SLZ	K01_shn => K01_SHN	K01_shz => K01_SHZ	K02_she => K02_SHE	K02_shl => K02_SLZ
K02_shn => K02_SHN	K02_shz => K02_SHZ	K03_she => K03_SHE	K03_shn => K03_SHN	K03_shz => K03_SHZ
K04_she => K04_SHE	K04_shn => K04_SHN	K04_shz => K04_SHZ	K05_she => K05_SHE	K05_shl => K05_SLZ
K05_shn => K05_SHN	K05_shz => K05_SHZ	K06_she => K06_SHE	K06_shn => K06_SHN	K06_shz => K06_SHZ
K07_she => K07_SHE	K07_shn => K07_SHN	K07_shz => K07_SHZ	K10_she => K10_SHE	K10_shl => K10_SLZ
K10_shn => K10_SHN	K10_shz => K10_SHZ	K11_she => K11_SHE	K11_shn => K11_SHN	K11_shz => K11_SHZ
K12_she => K12_SHE	K12_shn => K12_SHN	K12_shz => K12_SHZ	K13_she => K13_SHE	K13_shl => K13_SLZ
K13_shn => K13_SHN	K13_shz => K13_SHZ	K14_she => K14_SHE	K14_shn => K14_SHN	K14_shz => K14_SHZ
K15_she => K15_SHE	K15_shl => K15_SLZ	K15_shn => K15_SHN	K15_shz => K15_SHZ	KABR_SHZ => KABR_SHZ
KABR_shz => KABR_SHZ	KAHC_SHZ => KAHC_SHZ	KAHC_shz => KAHC_SHZ	KAHG_SHZ => KAHG_SHZ	KAHG_shz => KAHG_SHZ
KAIC_SHZ => KAIC_SHZ	KAIC_shz => KAIC_SHZ	KAI_SHZ => KAI_SHZ	KAI_shz => KAI_SHZ	KAI_zhs => KAI_SHZ
KAPH_SHE => KAPH_SHE	KAPH_SHN => KAPH_SHN	KAPH_SHZ => KAPH_SHZ	KAPH_she => KAPH_SHE	KAPH_shn => KAPH_SHN
KAPH_shz => KAPH_SHZ	KARR_SHZ => KARR_SHZ	KARR_shz => KARR_SHZ	KAWH_SHZ => KAWH_SHZ	KAWH_shz => KAWH_SHZ
KBK_BHE => KBK_BHE	KBK_BHN => KBK_BHN	KBK_BHZ => KBK_BHZ	KBK_LHE => KBK_LHE	KBK_LHN => KBK_LHN
KBK_LHZ => KBK_LHZ	KBM_SHZ => KBM_SHZ	KBM_shz => KBM_SHZ	KCE_SHZ => KCE_SHZ	KCE_shz => KCE_SHZ
KCE_slz => KCE_SHZ	KCGE_she => KCG_SHE	KCGE_shz => KCG_SHE	KCGN_shn => KCG_SHN	KCGN_shz => KCG_SHN

Table 1:

KCG_SHE => KCG_SHE	KCG_SHN => KCG_SHN	KCG_SHZ => KCG_SHZ	KCG_she => KCG_SHE	KCG_shn => KCG_SHN
KCG_shz => KCG_SHZ	KCVP_SHP => KCVP_BDF	KCVP_shp => KCVP_BDF	KCVP_shz => KCVP_BDF	KCV_shp => KCVP_BDF
KDAK_BHE => KDAK_BHE	KDAK_BHN => KDAK_BHN	KDAK_BHZ => KDAK_BHZ	KDAK_LHE => KDAK_LHE	KDAK_LHN => KDAK_LHN
KDAK_LHZ => KDAK_LHZ	KDC-_shz => KDC_SLZ	KDC-_slz => KDC_SLZ	KDC-_spz => KDC_SLZ	KDC_SHZ => KDC_SHZ
KDC_SLZ => KDC_SHZ	KDC_shz => KDC_SHZ	KDC_spz => KDC_SHZ	KELN_shn => KEL_SHN	KELN_shz => KEL_SHN
KEL_SHN => KEL_SHN	KEL_SHZ => KEL_SHZ	KEL_shz => KEL_SHZ	KICM_SHZ => KICM_SHZ	KIKV_SHZ => KIKV_SHZ
KIMD_SHZ => KIMD_SHZ	KINC_SHZ => KINC_SHZ	KIRH_SHZ => KIRH_SHZ	KIWB_SHZ => KIWB_SHZ	KJL_SHZ => KJL_SHZ
KJL_shz => KJL_SHZ	KKJ_BHE => KKJ_BHE	KKJ_BHN => KKJ_BHN	KKJ_BHZ => KKJ_BHZ	KLU_SHZ => KLU_SHZ
KLU_shz => KLU_SHZ	KLU_spz => KLU_SHZ	KNI_shz => KNI_SHZ	KNI_spz => KNI_SHZ	KNK_SHZ => KNK_SHZ
KNK_shz => KNK_SHZ	KNK_spz => KNK_SHZ	KODE_bhe => KOD_BHE	KODN_bhn => KOD_BHN	KOD_BHE => KOD_BHE
KOD_BHN => KOD_BHN	KOD_BHZ => KOD_BHZ	KOD_bhz => KOD_BHZ	KTH_SHZ => KTH_SHZ	KTH_shz => KTH_SHZ
KTH_spz => KTH_SHZ	KVT_SHZ => KVT_SHZ	KVT_shz => KVT_SHZ	KZA_BHE => KZA_BHE	KZA_BHN => KZA_BHN
KZA_BHZ => KZA_BHZ	KZA_LHE => KZA_LHE	KZA_LHN => KZA_LHN	KZA_LHZ => KZA_LHZ	LTI_SHZ => LTI_SHZ
LTI_shz => LTI_SHZ	LTI_spz => LTI_SHZ	LVA_SHZ => LVA_SHZ	LVA_shz => LVA_SHZ	LVY_spz => LVY_SHZ
MA2_BHE => MA2_BHE	MA2_BHN => MA2_BHN	MA2_BHZ => MA2_BHZ	MBC_BHE => MBC_BHE	MBC_BHN => MBC_BHN
MBC_BHZ => MBC_BHZ	MBM_shz => MBM_SHZ	MCIR_SHZ => MCIR_SHZ	MCIR_shz => MCIR_SHZ	MCK-_spz => MCK_SLZ
MCK_00B => MCK_00B	MCK_BHE => MCK_BHE	MCK_BHN => MCK_BHN	MCK_BHZ => MCK_BHZ	MCK_SHZ => MCK_SHZ
MCK_shz => MCK_SHZ	MCK_spz => MCK_SHZ	MDM_SHZ => MDM_SHZ	MDM_shz => MDM_SHZ	MDW_shz => MDW_SHZ
MGLS_SHZ => MGLS_SHZ	MGLS_shz => MGLS_SHZ	MGOD_SHZ => MGOD_SHZ	MGOD_shz => MGOD_SHZ	MID_SHZ => MID_SHZ
MID_shz => MID_SHZ	MID_spz => MID_SHZ	MLY_SHZ => MLY_SHZ	MLY_shz => MLY_SHZ	MMN_SHZ => MMN_SHZ
MMN_shz => MMN_SHZ	MMN_spz => MMN_SHZ	MNAT_SHZ => MNAT_SHZ	MNAT_shz => MNAT_SHZ	MOBC_BHE => MOBC_BHE
MOBC_BHN => MOBC_BHN	MOBC_BHZ => MOBC_BHZ	MSOM_SHZ => MSOM_SHZ	MSOM_shz => MSOM_SHZ	MSP_SHZ => MSP_SHZ
MSP_shz => MSP_SHZ	MSP_spz => MSP_SHZ	MSWE_she => MSW_SHE	MSWE_shz => MSW_SHE	MSWN_shn => MSW_SHN
MSWN_shz => MSW_SHN	MSW_SHE => MSW_SHE	MSW_SHN => MSW_SHN	MSW_SHZ => MSW_SHZ	MSW_she => MSW_SHE
MSW_shn => MSW_SHN	MSW_shz => MSW_SHZ	MTBL_SHZ => MTBL_SHZ	MTBL_shz => MTBL_SHZ	MTU_SHZ => MTU_SHZ
MTU_shz => MTU_SHZ	MTU_spz => MTU_SHZ	NAGA_shz => NAGA_SHZ	NAG_ => NAG_SHZ	NAG_SHZ => NAG_SHZ
NAG_shz => NAG_SHZ	NCA_shz => NCA_SHZ	NCA_spz => NCA_SHZ	NCG#_shz => NCG_SHZ	NCG_SHZ => NCG_SHZ
NCG_shz => NCG_SHZ	NCG_spz => NCG_SHZ	NCT_SHZ => NCT_SHZ	NCT_shz => NCT_SHZ	NCT_spz => NCT_SHZ
NDB_EHZ => NDB_EHZ	NEA_SHZ => NEA_SHZ	NEA_shz => NEA_SHZ	NEA_spz => NEA_SHZ	NGI_shz => NGI_SHZ
NHSA_BHZ => NHSA_BHZ	NKA_SHZ => NKA_SHZ	NKA_shz => NKA_SHZ	NKA_spz => NKA_SHZ	NMO_she => NMO_SHE
NMO_shn => NMO_SHN	NMO_shz => NMO_SHZ	NNL_SHZ => NNL_SHZ	NNL_shz => NNL_SHZ	NNL_spz => NNL_SHZ
OMEG_ => OMEG_TIME	OMEG_tcc => OMEG_TIME	OMGF_tcc => OMGF_TIME	OMGR_tcc => OMGR_TIME	OPT_SHZ => OPT_SHZ
OPT_shz => OPT_SHZ	OPT_spz => OPT_SHZ	OREC_tcc => OREC_TIME	PAX_SHZ => PAX_SHZ	PAX_shz => PAX_SHZ
PAX_spz => PAX_SHZ	PBBE_bhe => PMR_BHE	PBBN_bhn => PMR_BHN	PBB_bhz => PMR_BHZ	PBG_SHZ => PBG_SHZ
PBG_shz => PBG_SHZ	PDB_SHZ => PDB_SHZ	PDB_shz => PDB_SHZ	PDB_spz => PDB_SHZ	PET_BHZ => PET_BHZ
PET_SHE => PET_SHE	PET_SHN => PET_SHN	PET_SHZ => PET_SHZ	PHSA_BHZ => PHSA_BHZ	PIN_SHZ => PIN_SHZ
PIN_shz => PIN_SHZ	PIN_spz => PIN_SHZ	PLR_SHZ => PLR_SHZ	PLR_shz => PLR_SHZ	PLR_spz => PLR_SHZ
PME_shz => PME_SHZ	PME_spz => PME_SHZ	PMR-_shz => PMR_SLZ	PMR-_slz => PMR_SLZ	PMR_BHE => PMR_BHE
PMR_BHN => PMR_BHN	PMR_BHZ => PMR_BHZ	PMS_SHZ => PMS_SHZ	PMS_shz => PMS_SHZ	PMS_spz => PMS_SHZ

Table 1:

PN6E_shz => PN6_SHE	PN6N_shn => PN6_SHN	PN6N_shz => PN6_SHN	PN6_shz => PN6_SHZ	PN6_spz => PN6_SHZ
PN7A_BDF => PN7A_BDF	PN7A_SHP => PN7A_BDF	PN7A_SHZ => PN7A_SHZ	PN7A_shz => PN7A_SHZ	PN7_shz => PN7_SHZ
PNL_SHZ => PNL_SHZ	PNL_shz => PNL_SHZ	PPD_SHZ => PPD_SHZ	PPD_shz => PPD_SHZ	PPSA_BHZ => PPSA_BHZ
PRG_SHZ => PRG_SHZ	PRG_shz => PRG_SHZ	PS1A_SHZ => PS1A_SHZ	PS1A_shz => PS1A_SHZ	PS1_shz => PS1_SHZ
PS4A_SHZ => PS4A_SHZ	PS4A_shz => PS4A_SHZ	PS4_spz => PS4_SHZ	PS5_shz => PS5_SHZ	PS7_she => PS7_SHE
PS7_shn => PS7_SHN	PS7_shz => PS7_SHZ	PV6E_she => PV6_SHE	PV6E_shz => PV6_SHE	PV6N_shn => PV6_SHN
PV6N_shz => PV6_SHN	PV6_SHE => PV6_SHE	PV6_SHN => PV6_SHN	PV6_SHZ => PV6_SHZ	PV6_she => PV6_SHE
PV6_shn => PV6_SHN	PV6_shz => PV6_SHZ	PVV_SHZ => PVV_SHZ	PVV_shz => PVV_SHZ	PWA_SHZ => PWA_SHZ
PWA_shz => PWA_SHZ	PWA_spz => PWA_SHZ	PWL_SHZ => PWL_SHZ	PWL_shz => PWL_SHZ	QT05_BHE => QT05_BHE
QT05_BHN => QT05_BHN	QT05_BHZ => QT05_BHZ	RAG_SHZ => RAG_SHZ	RAG_shz => RAG_SHZ	RC01_BHE => RC01_BHE
RC01_BHN => RC01_BHN	RC01_BHZ => RC01_BHZ	RDN_SHZ => RDN_SHZ	RDN_shz => RDN_SHZ	RDN_spz => RDN_SHZ
RDS_shz => RDS_SHZ	RDS_spz => RDS_SHZ	RDT#_shz => RDT_SHZ	RDTA_shz => RDTA_SHZ	RDTE_shz => RDT_SHE
RDTE_spz => RDT_SHE	RDTN_shz => RDT_SHN	RDTN_spz => RDT_SHN	RDTZ_shz => RDT_SHZ	RDT_SHZ => RDT_SHZ
RDT_shz => RDT_SHZ	RDT_spz => RDT_SHZ	RDW_SHZ => RDW_SHZ	RDW_shz => RDW_SHZ	RED-_shz => RED_SLZ
RED-_slz => RED_SLZ	REDE_she => RED_SHE	REDE_shz => RED_SHE	REDN_shn => RED_SHN	REDN_shz => RED_SHN
RED_SHE => RED_SHE	RED_SHN => RED_SHN	RED_SHZ => RED_SHZ	RED_SLZ => RED_SLZ	RED_she => RED_SHE
RED_shn => RED_SHN	RED_shz => RED_SHZ	RED_spz => RED_SHZ	REF-_shz => REF_SLZ	REF-_slz => REF_SLZ
REFE_she => REF_SHE	REFE_shz => REF_SHE	REFN_shn => REF_SHN	REFN_shz => REF_SHN	REF_SHE => REF_SHE
REF_SHN => REF_SHN	REF_SHZ => REF_SHZ	REF_SLZ => REF_SLZ	REF_she => REF_SHE	REF_shn => REF_SHN
REF_shz => REF_SHZ	REF_slz => REF_slz	RES_BHE => RES_BHE	RES_BHN => RES_BHN	RES_BHZ => RES_BHZ
RND_SHZ => RND_SHZ	RND_shz => RND_SHZ	RND_spz => RND_SHZ	ROOF_shz => ROOF_SHZ	RS1_shz => RS1_SHZ
RS2_shz => RS2_SHZ	RSO#_shz => RSO_SHZ	RSO_SHZ => RSO_SHZ	RSO_shz => RSO_SHZ	RWS_shz => RWS_SHZ
SAW_BHE => SAW_BHE	SAW_BHN => SAW_BHN	SAW_BHZ => SAW_BHZ	SAW_SHZ => SAW_SHZ	SAW_shz => SAW_SHZ
SAW_spz => SAW_SHZ	SBEA_BHZ => SBEA_BHZ	SCM_SHZ => SCM_SHZ	SCM_shz => SCM_SHZ	SCM_spz => SCM_SHZ
SDG_SHZ => SDG_SHZ	SDG_shz => SDG_SHZ	SDG_spz => SDG_SHZ	SDN-_shz => SDN_SLZ	SDN-_slz => SDN_SLZ
SDN_SHZ => SDN_SHZ	SDN_shz => SDN_SHZ	SDN_spz => SDN_SHZ	SGA_SHZ => SGA_SHZ	SGA_shz => SGA_SHZ
SHU_shz => SHU_SHZ	SHU_spz => SHU_SHZ	SHW_EHZ => SHW_EHZ	SIT-_shz => SIT_SLZ	SIT-_slz => SIT_SLZ
SIT_BHE => SIT_BHE	SIT_BHN => SIT_BHN	SIT_BHZ => SIT_BHZ	SIT_SHZ => SIT_SHZ	SIT_shz => SIT_SHZ
SIT_spz => SIT_SHZ	SKN#_shz => SKN_SHZ	SKNE_she => SKN_SHE	SKNE_shz => SKN_SHE	SKNE_spz => SKN_SHE
SKNN_shn => SKN_SHN	SKNN_shz => SKN_SHN	SKNN_spz => SKN_SHN	SKN_SHE => SKN_SHE	SKN_SHN => SKN_SHN
SKN_SHZ => SKN_SHZ	SKN_she => SKN_SHE	SKN_shn => SKN_SHN	SKN_shz => SKN_SHZ	SKN_spz => SKN_SHZ
SKN_tcc => SKN_TIME	SLK#_shz => SLK_SHZ	SLK_SHZ => SLK_SHZ	SLK_shz => SLK_SHZ	SLK_spz => SLK_SHZ
SLOC_tcc => SLOC_TIME	SMY_BHE => SMY_BHE	SMY_BHN => SMY_BHN	SMY_BHZ => SMY_BHZ	SMY_BLE => SMY_BLE
SMY_BLN => SMY_BLN	SMY_BLZ => SMY_BLZ	SMY_SHZ => SMY_SHZ	SMY_shz => SMY_SHZ	SMY_spz => SMY_SHZ
SNE#_she => SKN_SHE	SNE#_shz => SKN_SHE	SNK_spz => SNK_SHZ	SNN#_shn => SKN_SHN	SNN#_shz => SKN_SHN
SOS_EHZ => SOS_EHZ	SP-E_ => SPB_BLE	SP-N_ => SPB_BLN	SP-_ => SPB_BLZ	SPBE_ => SPB_BHE
SPBN_ => SPB_BHN	SPBZ_ => SPB_BHZ	SPB_ => SPB_BHZ	SPIA_BHE => SPIA_BHE	SPIA_BHN => SPIA_BHN
SPIA_BHZ => SPIA_BHZ	SPIA_HHE => SPIA_HHE	SPIA_HHN => SPIA_HHN	SPIA_HHZ => SPIA_HHZ	SPLE_ => SPL_SHE
SPLN_ => SPL_SHN	SPLZ_ => SPL_SHZ	SPSE_ => SPS_SHE	SPSN_ => SPS_SHN	SPSZ_ => SPS_SHZ

Table 1:

SPU#_shz => SPU_SHZ	SPU_SHZ => SPU_SHZ	SPU_shz => SPU_SHZ	SPU_spz => SPU_SHZ	SSLN_BDF => SSLN_BDF
SSLN_SHP => SSLN_BDF	SSLN_SHZ => SSLN_SHZ	SSLN_shp => SSLN_BDF	SSLN_shz => SSLN_SHZ	SSLS_SHE => SSLS_SHE
SSLS_SHN => SSLS_SHN	SSLS_SHZ => SSLS_SHZ	SSLS_she => SSLS_SHE	SSLS_shn => SSLS_SHN	SSLS_shz => SSLS_SHZ
SSLW_SHZ => SSLW_SHZ	SSLW_shz => SSLW_SHZ	SSN#_shz => SSN_SHZ	SSN_SHZ => SSN_SHZ	SSN_shz => SSN_SHZ
SSN_spz => SSN_SHZ	SSP_SHZ => SSP_SHZ	SSP_shz => SSP_SHZ	STLK_SHZ => STLK_SHZ	STLK_shz => STLK_SHZ
STRP_shz => STR_BDF	SVW_SHZ => SVW_SHZ	SVW_shz => SVW_SHZ	SVW_spz => SVW_SHZ	SWD_BHE => SWD_BHE
SWD_BHN => SWD_BHN	SWD_BHZ => SWD_BHZ	SWD_SHZ => SWD_SHZ	SWD_shz => SWD_SHZ	SWD_spz => SWD_SHZ
SYI_SHZ => SYI_SHZ	SYI_shz => SYI_SHZ	TAM_shz => TAM_SHZ	TAR_she => TAR_SHE	TAR_shn => TAR_SHN
TAR_shz => TAR_SHZ	TEST_shz => TEST_SHZ	TGL_SHZ => TGL_SHZ	TGL_shz => TGL_SHZ	TGL_spz => TGL_SHZ
THW_spz => THW_SHZ	THY_BHE => THY_BHE	THY_BHN => THY_BHN	THY_BHZ => THY_BHZ	THY_SHZ => THY_SHZ
THY_shz => THY_SHZ	THY_spz => THY_SHZ	TIXI_BHE => TIXI_BHE	TIXI_BHN => TIXI_BHN	TIXI_BHZ => TIXI_BHZ
TMW_SHZ => TMW_SHZ	TMW_shz => TMW_SHZ	TMW_spz => TMW_SHZ	TNA_BHE => TNA_BHE	TNA_BHN => TNA_BHN
TNA_BHZ => TNA_BHZ	TNA_HHE => TNA_HHE	TNA_HHN => TNA_HHN	TNA_HHZ => TNA_HHZ	TOA_SHZ => TOA_SHZ
TOA_shz => TOA_SHZ	TOA_spz => TOA_SHZ	TRF_SHZ => TRF_SHZ	TRF_shz => TRF_SHZ	TRF_spz => TRF_SHZ
TT01_SHZ => TT01_SHZ	TTA_SHZ => TTA_SHZ	TTA_shz => TTA_SHZ	TTA_spz => TTA_SHZ	TZ0_spz => TZO_SHZ
TZL_SHZ => TZL_SHZ	TZL_shz => TZL_SHZ	TZO_spz => TZO_SHZ	UCH_BHE => UCH_BHE	UCH_BHN => UCH_BHN
UCH_BHZ => UCH_BHZ	UCH_LHE => UCH_LHE	UCH_LHN => UCH_LHN	UCH_LHZ => UCH_LHZ	ULHL_BHE => ULHL_BHE
ULHL_BHN => ULHL_BHN	ULHL_BHZ => ULHL_BHZ	ULHL_LHE => ULHL_LHE	ULHL_LHN => ULHL_LHN	ULHL_LHZ => ULHL_LHZ
UNV_BHE => UNV_BHE	UNV_BHN => UNV_BHN	UNV_BHZ => UNV_BHZ	UNV_HHE => UNV_HHE	UNV_HHN => UNV_HHN
UNV_HHZ => UNV_HHZ	USFS_shz => USFS_SHZ	USP_BHE => USP_BHE	USP_BHN => USP_BHN	USP_BHZ => USP_BHZ
USP_LHE => USP_LHE	USP_LHN => USP_LHN	USP_LHZ => USP_LHZ	VIB_EHZ => VIB_SHZ	VLZE_she => VLZ_SHE
VLZE_shz => VLZ_SHE	VLZE_spz => VLZ_SHE	VLZN_shn => VLZ_SHN	VLZN_shz => VLZ_SHN	VLZ_SHE => VLZ_SHE
VLZ_SHN => VLZ_SHN	VLZ_SHZ => VLZ_SHZ	VLZ_she => VLZ_SHE	VLZ_shn => VLZ_SHN	VLZ_shz => VLZ_SHZ
VLZ_spz => VLZ_SHZ	VOGL_SHE => VOGL_SHE	VOGL_SHN => VOGL_SHN	VOGL_SHZ => VOGL_SHZ	VZW_SHZ => VZW_SHZ
VZW_shz => VZW_SHZ	VZW_spz => VZW_SHZ	WACK_SHE => WACK_SHE	WACK_SHN => WACK_SHN	WACK_SHZ => WACK_SHZ
WANC_SHZ => WANC_SHZ	WASW_SHZ => WASW_SHZ	WAX_SHZ => WAX_SHZ	WAX_shz => WAX_SHZ	WAZA_SHZ => WAZA_SHZ
WESE_SHZ => WESE_SHZ	WESE_shz => WESE_SHZ	WESN_SHZ => WESN_SHZ	WESN_shz => WESN_SHZ	WESS_SHE => WESS_SHE
WESS_SHN => WESS_SHN	WESS_SHZ => WESS_SHZ	WESS_she => WESS_SHE	WESS_shn => WESS_SHN	WESS_shz => WESS_SHZ
WFAR_SHZ => WFAR_SHZ	WFAR_shz => WFAR_SHZ	WHY_BHE => WHY_BHE	WHY_BHN => WHY_BHN	WHY_BHZ => WHY_BHZ
WPOG_SHZ => WPOG_SHZ	WPOG_shz => WPOG_SHZ	WRG_SHZ => WRG_SHZ	WRG_shz => WRG_SHZ	WRH_SHZ => WRH_SHZ
WRH_shz => WRH_SHZ	WRH_spz => WRH_SHZ	WS2_bhe => WS2_BHE	WS2_bhn => WS2_BHN	WS2_bhz => WS2_BHZ
WS2_she => WS2_SHE	WS2_shn => WS2_SHN	WS2_shz => WS2_SHZ	WS3_shz => WS3_SHZ	WSD_she => WSD_SHE
WSD_shn => WSD_SHN	WSD_shz => WSD_SHZ	WTUG_SHZ => WTUG_SHZ	WTUG_shz => WTUG_SHZ	WWVE_sht => WWVE_TIME
XK1_shz => XK1_SHZ	XK2_shz => XK2_SHZ	XK3_shz => XK3_SHZ	XK4_shz => XK4_SHZ	XK5_shz => XK5_SHZ
XK6E_shz => XK6_SHE	XK6N_shz => XK6_SHN	XK6_shz => XK6_SHZ	XK7_shz => SK7_SHZ	XLV_SHZ => XLV_SHZ
XLV_shz => XLV_SHZ	XLV_spz => XLV_SHZ	XM1_shz => XM1_SHZ	XM2_shz => XM2_SHZ	XM3_shz => XM3_SHZ
XM4_shz => XM4_SHZ	XM5E_shz => XM5_SHE	XM5N_shz => XM5_SHN	XM5_shz => XM5_SHZ	XM6_shz => XK6_SHZ
XM7_shz => XM7_SHZ	XXX2_shz => XXX2_SHZ	XXX_SHZ => XXX_SHZ	XXX_she => XXX_SHE	XXX_shz => XXX_SHZ
YAH_SHZ => YAH_SHZ	YAH_shz => YAH_SHZ	YAK_BHE => YAK_BHE	YAK_BHN => YAK_BHN	YAK_BHZ => YAK_BHZ

Table 1:

YKB0_SHZ => YKB0_SHZ	YKB1_SHZ => YKB1_SHZ	YKB2_SHZ => YKB2_SHZ	YKB3_SHZ => YKB3_SHZ	YKB4_SHZ => YKB4_SHZ
YKB6_SHZ => YKB6_SHZ	YKB7_SHZ => YKB7_SHZ	YKB8_SHZ => YKB8_SHZ	YKB9_SHZ => YKB9_SHZ	YKR1_SHZ => YKR1_SHZ
YKR2_SHZ => YKR2_SHZ	YKR3_SHZ => YKR3_SHZ	YKR4_SHZ => YKR4_SHZ	YKR5_SHZ => YKR5_SHZ	YKR6_SHZ => YKR6_SHZ
YKR7_SHZ => YKR7_SHZ	YKR8_SHZ => YKR8_SHZ	YKR9_SHZ => YKR9_SHZ	YKT_shz => YKT_SHZ	YKU-_shz => YKU_SLZ
YKU_SHZ => YKU_SHZ	YKU_shz => YKU_SHZ	YKU_spz => YKU_SHZ	YKW1_BHE => YKW1_BHE	YKW1_BHN => YKW1_BHN
YKW1_BHZ => YKW1_BHZ	YKW2_BHE => YKW2_BHE	YKW2_BHN => YKW2_BHN	YKW2_BHZ => YKW2_BHZ	YKW3_BHE => YKW3_BHE
YKW3_BHN => YKW3_BHN	YKW3_BHZ => YKW3_BHZ	YKW4_BHE => YKW4_BHE	YKW4_BHN => YKW4_BHN	YKW4_BHZ => YKW4_BHZ
YSS_BHE => YSS_BHE	YSS_BHN => YSS_BHN	YSS_BHZ => YSS_BHZ	ZHIN_ => HIN_SHZ	ZRAG_ => RAG_SHZ
ZRO_SHZ => ZRO_SHZ	ZRO_shz => ZRO_SHZ	ZSGA_ => SGA_SHZ	ZWAX_ => WAX_SHZ	