Introduction to the Key-based Configuration Scheme

for Quanterra Dataloggers

1. INTRODUCTION

In conjunction with the change to MultiSHEAR Version 36/09-0531, Quanterra began to use a new method to configure dataloggers. The new method seeks to consolidate all station specific configuration information to a few, simple text files and to utilize universal template files for the basis of startup and optional procedures. This has several advantages;

- The simple change of one parameter is replaced everywhere this value appears in the system.
- A change in network-wide preferences usually involves a change in a single file and no station specific information needs to be altered or inserted.
- Simple changes are easy to make, such as switching off the calboard, or changing ip port assignments. Field technicians providing local support can easily understand the procedure to make changes.
- Generation of configuration information for several stations is simple and can be aided by programs.
- Syntax of values and selection of high level options is greatly simplified. For example, selecting the serial port /x2 to run slip requires only $SLIP 2 be specified and the appropriate ip address and port numbers. Increasing the detector threshold involves changes $DET1_LEV 5 to $DET1_LEV 7, rather than 12 references in each of 12 channel definitions.

It should be noted that the underlying bootup process and configuration for MultiShear is nearly the same as for older versions of Shear-that is the system still reads a series of procedure files to start the desired processes (for example; startup, startup_147, aqgo, ...) and then reads a large file, aqcfg, that specifies the options for the acquisition software. What has changed is that the files used in the boot sequence are actually created during the bootup from a set of templates that are adapted to specific settings using keys. Most of the startup procedure files are preserved in the ram disk and can be inspected by an operator familiar with the earlier syntax and selections for Shear or UltraShear. In this way one can confirm that the new key configuration scheme really recreates what one had devised under the older scheme. It is, in fact, not necessary to use the key-based configuration scheme to use the MShear software. Mshear does contain new features and a new syntax within the aqcfg file, primarily to support the notion of multiple data sources.

This document introduces the key-based configuration scheme. In section 2, we describe what a key is, how it gets used and the rules for implementation. Template files are the files in which the key substitutions are inserted. The organization and use of template files are discussed in section 3. The operational aspects of some important key groups are discussed in section 4 to provide examples about how keys work and which to choose. In section 5, the scope shifts from selecting keys for a particular datalogger to how one can manage the key selections for a group of dataloggers that comprise a large network, with a few suggestions for automation. Section 6 describes the bootup sequence in great detail. Some common errors in using the key-based configuration scheme are exhibited and explained. Section 7 is a run through of the steps for setting up a configuration scheme from the Quanterra distribution.

========== Table of Contents ==========

==========

2. KEYS

To introduce the concept, we begin with an example. The key-based configuration scheme is much like a set of form letters and an address (or larger customer) database. Suppose we want to send a document to several different people that might read;

Dear Joe Steim;
    You may already have won $10 Million Dollars!

Since the people may reside in many countries we must be able to replace more than simply the name. In our example construction, we create a template file like;

Dear %first_name% %last_name%;
    You may already have won %symbol%%amount% %currency%!

We next create a key file which contains the following keyname and keyvalue pairs.
first_name Joe
last_name Steim
symbol $
amount "10 Million"
currency Dollars

In our example, first_name is a key (or keyname) and it has a keyvalue of Joe.
A program called macromania is used to replace every occurance of %key% with the appropriate value of the key in the template file and to write out a file with all substitutions. A few quick rules about keys;
The keyname is not case sensitive; Symbol, SYMBOL and symbol are all the same key.
The keyvalue can be enclosed in double quotes to accomodate values which contain spaces. Should it be necessary to use the delimiter, %, in a key, one can declare a new escape. Such details are available in the more advanced reference macromania.html. In the key file, the first word is the keyname the second word is the keyvalue and everything else is ignored. Comments are added to the same line to help identify what the key does. A keyvalue can refer to another key. Here is an example;
prize %symbol%%amount%

When macromania processes a template file, every %keyname% in the template file must have a corresponding keyname and keyvalue pair in the keyfile (or more generally the key database) otherwise it cannot make a replacement. Conversely, there is no requirement that every one of the keynames listed in the keyfile be used or even that any be used. There are lots of keys that are used only in rare crcumstances.

To review, a template file is a file containing keynames which will be replaced with keyvalues. Keys or keynames are the references to the parts that should be replaced. Keyvalues are text strings to replace the keyname with.

The next sections explore ways of grouping keynames and keyvalues, first as clumps referenced by a keymacro name and then as whole files that contain more keyname keyvalue pairs or more keymacro descriptions.

KEYMACROS

It is helpful to refer to a group of keys whose values are somehow related. For example, we define three groups of the same keys but with different values;

[$GRAND_PRIZE]
symbol $
amount "10 Million"
currency Dollars
exclaim Whahooo!

[$FIRST_PRIZE]
symbol $
amount “100”
currency Dollars
exclaim “Isn’t that nice?”

[$REAL_PRIZE]
symbol “”
amount “a 1 year subscription”
currency “”
exclaim “Please pay $56 Dollars.”

This group of keys is called a keymacro. It helps to shorten the key file so that not every key need be mentioned and certain combinations of keys can be easily referenced. The definition begins with the [$macroname] line and ends with the start of the next macro definition (or the end of file). Comments can be included on the definition line or any line beginning with a *.

So our new key file might read;
first_name Joe
last_name Steim
$REAL_PRIZE

And we might send a followup letter using another template file;

Dear %first_name% %last_name%;
Thank you for you interest in our sweepstakes.
You will receive %symbol%%amount% %currency%!
%exclaim%

If the keyfile refers to a keymacro (in our previous example $REAL_PRIZE) then the keymacro must be defined. Standard keymacro definitions are located in a large file called keymacros, any additional keymacro definitions you need should be added to a file usermacros.networkname. For example, in the TriNet network we use a file called usermacros.trinet for our custom keymacro definitions and refrain from editting the Quanterra issued file keymacros. This becomes important when Quanterra issues an update to the keymacros file which may or may not affect your editting. We discuss usermacros more later.

To make this really work, things are more complicated.

Keymacro definitions may include arguments. Up to thirty five are allowed and are referenced within the keymacros by the identifier 1,2,3…9,A,B…Z. The argument values can be inserted in keynames and/or keyvalues within the keymacro definition. An argument whose value is undefined is considered to be the null string. To help provide context, the expected arguments values are provided on the line containing the keymacro definition (though it is not required) and a few lines of comments might describe what's going on. Arguments are separated by semicolons.

Here is a real example from the keymacros file of a macro definition. It has only one argument, the channel group identifier, which can take a value of A or B or C or D or X or Y. It is used to change the sign of the gain value for all three channels in any specified group.

```
[$POLARITY_REV] A|B|C|D|X|Y
*changes the sign of the gain in the server section
*for the specified channel group.
*Functionally similar to reversed=y in lcq
POLARITY_$1$1 -
POLARITY_$1$2 -
```

```
POLARITY_$1$3 -
```

Entering the following line in a keyfile (the file used to set the datalogger configurations);
$POLARITY_REV B

would create within the key database the following keys;

```
POLARITY_B1 -
POLARITY_B2 -
POLARITY_B3 -
```

And the template file aqcfg_1 that has a server section like;

```
GAIN=%CHANNEL_B1%,%POLARITY_B1%%GAIN_B1%
GAIN=%CHANNEL_B2%,%POLARITY_B2%%GAIN_B2%
GAIN=%CHANNEL_B3%,%POLARITY_B3%%GAIN_B3%
```

This aqcfg_1 file would get processed by the program macromania to yield a file aqcfg like;

```
GAIN=4,-1.0
GAIN=5,-1.0
GAIN=6,-1.0
```

Not only are arguments supported for keymacro definitions, but substrings of arguments are also supported. For example, $1:2:2$ refers to the first argument, second and third characters. $1:2:4$ refers to the first argument, and four characters starting with the second character. In other words, the second through fifth character.

To illustrate, we walk through the $CHAN2MSEED keymacro definition. The keynames shown here take the form of SEED_HA1, SEED_HA2 where the HA1 refers to a logical channel group A, component 1, at the H sample rate. This keymacro operates on a group of three channels where the particular group is the first argument, 1. We want to give each of these keynames a value that corresponds to a valid SEED channel name and for as many sample rates as we might have. The second argument will form the middle letter of a SEED name. The third, fourth and fifth arguments will provide the third letter of the SEED name for each of the three channels. The set of arguments EE;HH;MM, etc refer to the input sample rate and the first letter of the output SEED name. This uses argument substrings to pull apart the pair EE or HH and let the first character indicate the sample rate, while the second character indicates first letter of the SEED channel name.

Here's the keymacro definition as it appears in the file keymacros;

```
[$CHAN2MSEED] A|B|C|D;I;1;2;3;[EE;HH;BB;MM;LL;VV;UU]
*group;SEED_inst_code;SEED_comp_1;comp2;comp3;stream|seed
*used_to_alter_stream_name_into_some_seed_chan
SEED_$6:1:1$$1$1 $6:2:1$$2$$3$
SEED_$6:1:1$$1$2 $6:2:1$$2$$4$
SEED_$6:1:1$$1$3 $6:2:1$$2$$5$
SEED_$7:1:1$$1$1  $7:2:1$$2$$3$
SEED_$7:1:1$$1$2  $7:2:1$$2$$4$
SEED_$7:1:1$$1$3  $7:2:1$$2$$5$
SEED_$8:1:1$$1$1   $8:2:1$$2$$3$
SEED_$8:1:1$$1$2   $8:2:1$$2$$4$
SEED_$8:1:1$$1$3   $8:2:1$$2$$5$
SEED_$9:1:1$$1$1    $9:2:1$$2$$3$
SEED_$9:1:1$$1$2    $9:2:1$$2$$4$
SEED_$9:1:1$$1$3    $9:2:1$$2$$5$
SEED_$A:1:1$$1$1     $A:2:1$$2$$3$
SEED_$A:1:1$$1$2     $A:2:1$$2$$4$
```

```
SEED_$A:1:1$$1$3      $A:2:1$$2$$5$
SEED_$B:1:1$$1$1     $B:2:1$$2$$3$
SEED_$B:1:1$$1$2     $B:2:1$$2$$4$
SEED_$B:1:1$$1$3     $B:2:1$$2$$5$
SEED_$C:1:1$$1$1    $C:2:1$$2$$3$
SEED_$C:1:1$$1$2    $C:2:1$$2$$4$
SEED_$C:1:1$$1$3    $C:2:1$$2$$5$
```

If we use the following line to the keyfile (the file that sets up the datalogger configuration);
$CHAN2MSEED A;H;Z;N;E;HH;BB;LL

This keymacro will reduce to a specific list of keynames and keyvalues in the key database;
SEED_HA1 HHZ
SEED_HA2 HHN
SEED_HA3 HHE
SEED_BA1 BHZ
SEED_BA2 BHN
SEED_BA3 BHE
SEED_LA1 LHZ
SEED_LA2 LHN
SEED_LA3 LHE
SEED_A1 HZ
SEED_A2 HN
SEED_A3 HE
SEED_A1 HZ
SEED_A2 HN
SEED_A3 HE
SEED_A1 HZ
SEED_A2 HN
SEED_A3 HE
SEED_A1 HZ
SEED_A2 HN
SEED_A3 HE

Keynames like SEED_A1 are not used in any template file, and multiple calls of the same keyname are ignored, so the last 12 lines are extraneous in this case. This may seem a long way around to specifying the 9 key keyvalue pairs, however the advantage comes in that the syntax can specify ANY group keynames and assign them any SEED name.

If you wanted to name accelerometer channels with SEED names HLZ, BLZ, LLZ use;
$CHAN2MSEED A;L;Z;N;E;HH;BB;LL
If you prefer the SEED names HGZ, BGZ, LGZ, HGN, BGN, LGN, … use;
$CHAN2MSEED A;G;Z;N;E;HH;BB;LL
Or if you are a real stickler for SEED and prefer EGZ, SGZ, LGZ, EGN, SGN, LGN, … use;
$CHAN2MSEED A;G;Z;N;E;HE;BS;LL
Or if this happened to be a borehole where the components are numbered, HG1, BG1, LG1, HG2, BG2, LG2, HG3, BG3, LG3 then use;
$CHAN2MSEED B;G;1;2;3;HH;BB;LL

If the particular configuration you need to build cannot easily be specified with this keymacro, you can add you own keymacro definition to a file usermacros.netname like;

[$CHAN2MSEED+FACT]
SEED_HA1 CC4
SEED_HA2 CC3
SEED_HA3 CC2

SEED_HX1 CC1
SEED_HB1 CC8
SEED_HB2 CC7
SEED_HB3 CC6
SEED_HY1 CC5
SEED_BA1 XL4
SEED_BA2 XL3
SEED_BA3 XL2
SEED_BX1 XL1
SEED_BB1 XL8
SEED_BB2 XL7
SEED_BB3 XL6
SEED_BY1 XL5

USERMACRO FILES
Quanterra provides numerous keymacro definitions contained in the file keymacros. Through the use of arguments, many custom configurations of diverse keys can be specified. Still, there is a need for users to specify their own keymacro definitions. The key configuration scheme will read other files containing keymacro definitions. These are added to the key database by inserting a line in the key file (or more commonly in the userinfo file, which we'll get to in the next section) as;
$MACROFILE usermacros.trinet

Perhaps there are certain combinations of keys or a group of lines that are common to key files for many stations in your network and you want to just refer to these as a keymacro. For example, the destination ip address for data telemetry may vary depending on the route the datalogger has to connect to the central host computer(s). One route may come by private lines another by internet. The ip address of the [multihomed]central computer(s) will be different on each interface. Several keys will change values depending on the route. Such a keymacro definition is highly specifc to the network and is not properly part of the standard keymacros file.

Here's the specific example which might be contained in the file usermacros.trinet;

[$INTERFACE_Inet]
i1-net 131.215
i1-host 85.6
i2-net 131.215
i2-host 85.7

[$INTERFACE_framerelay]
i1-net 192.168.56
i1-host 10
i2-net 192.168.43
i2-host 11

USERINFO FILES or INCLUDE FILES
In addition to being able to add your own keymacro definitions to the usermacros files, you can also group keys into other files and include these files into the key file as if they were all one big list of keyname keyvalue pairs. We call these userinfo files because, in this particular instance, it groups all the keys common to a network into a single file for easy maintenance. The files to be included can be called anything, but the contents are included in the key file (at the exact line of the call) by a line in the key file;
$INCLUDE userinfo.trinet

For future reference, when it comes time to create a configuration for a datalogger, you will need to set up four files. These are hosts, userinfo.netname, usermacros.netname, and the key file. The first three are

common to all stations in the network while the key file contains specifics for a particular station. Thus there is an interplay between the key file and the userinfo file. Each contains keys, those that are common network keys go in the userinfo file. Which keys are common to the network will vary by the network. For instance, if ALL stations have the same sensors and same recording configuration, these keys would likely end up in the userinfo file, whereas a more diverse array of equipment might require different sensors and recording for each station and these keys would be better placed in the key file, not in the userinfo file.

CONSTRUCTING THE KEY DATBASE

So there are many files that contain keyname keyvalue pairs; key files, keymacro files, usermacros files, and include files (userinfo being one of these). The key configuration scheme depends on the program macromania to assemble all the keynames into a database (you can think of it as a single list). Macromania then uses this database to replace keynames with keyvalues whereever a keyname is found in the template file. First it reads the key file looking for keys (it ignores keymacros names during this pass), if there are include files mentioned these are expanded in place within the key file. Next, macromania builds a list of keymacro definitions by concatenating each usermacro file (specified in a key or include file as $MACROFILE usermacros.aaa) with the keymacros file (the usermacros are before the keymacros). Recall that keymacros are just clumps of keyname keyvalue pairs, even though this can be complex due to argument substitution and recursive macro calls. Macromania now goes back to read the key file again, this time looking for keymacros listed in the key file and expands each keymacro into a clump of keynames and keyvalue pairs, adding these keynames to its database (or list), after those keynames that are directly spelled out in the key file. The order keynames are encountered becomes rather important and is used to great advantage (and occasional anguish). The first instance of a keyname keyvalue pair encountered in the key database is the one that is used for replacements. Subsequent keyvalues for the keyname are ignored. This convention is opposite of that encountered when setting variables within programs and scripts. In that case the last write to the variable determines the value. The key configuration is setup so that the specification of the datalogger type, using a keymacro of say $Q4128, calls the normal hardware configuration keys and keymacros for this datalogger and it then calls $Q_global_defaults. Q_global_defaults keymacro sets a whole bunch of keys to their default values. This means your key file need only mention keys that set something special. This reduces the key file size to short, distinctive settings. Because the defaults are set last, you can alter the keyvalue of a key buried deep in keymacros, by listing it specifically in the key file without having to invent a new keymacro definition or specifying all the related keys and then hope some obscure reference doesn't override it.

After the database of keyname keyvalue pairs is constructed, attention turns to the template files where the keynames will be replaced with keyvalues.

3. TEMPLATE FILES

Template files are normally given a name that ends with _1 (that’s a one, not a letter), and the output file after the keys have been replaced is the same name without the _1. The program macromania does this conversion. The templates are always copied into /r0 before macromania is run. The order in which files are copied to /r0 allows overwriting of standard files with files more specific to a particular operation.

ORGANIZATION

The key values for a station are kept in the file /h0/MSHEAR/CFG/STATIONS/STA/key. Template files (where the key values are inserted) are located as follows:

```
 /h0/MSHEAR/CFG/-------SYSBOOT_1/
                                    _bootlist
                                    _netstart_1
                                    aqload
                                    loads_1
                                    set_password
                                    sliplist_1
                                    startmodem_1
                                    startup_1
                              ETC_KEY/
```

```
        _clean
        _netlist
         _start_1
        host.conf_1
        hosts.equiv
        hosts_1
        inetd.conf
        networks_1
        nfs.map
        protocols
        resolv.conf_1
        rpc
        services_1
NETWORKS/
        DEFAULT/
                IIRFilters
                FIRFilters
                readme.txt
        MASTER/
                aqgo_1
                aqcfg_1
                aqstop
                fingerinfo_1
                hosts.sample
                motd_1
                password_1
                _runlist
                userinfo.sample
                usermacros.sample
                vbb_system_description_1
```

Template files in /h0/MSHEAR/CFG/SYSBOOT_1 are bootup files which typically do not required user customization. All the features can be switched via keys. The files in /h0/MSHEAR/CFG/ETC_KEY are related to ip networking and again do not require user customization. Those files which the network operator may need to tune are located in the directory NETWORKS. Templates located in NETWORKS/DEFAULT are copied to /r0 first. Templates in one other directory, that specified by the value of the key **netdir**, are then copied into /r0, possibly overwriting previous templates. This allows a station by station selection of which templates to employ. Generally a network operator would place all network directories on all systems. The selection of the netdir template directory was to allow what may become very specialized templates for remote hub operations to exist independent of progress towards a unified template for all operations. This feature has an unexpected benefit in that if you want to try a new template you can place it in a new directory and just alter the key file to point there. Then if things don't work out you can get quickly back to where you were by using your old key file. Which directory of templates is used depends on the key file.

After macromania completes all replacemnet of key in the template files, the output files are used to boot the datalogger. These files remain in /r0 after bootup for inspection. They are not used again for bootup, each bootup creates new files from the keys. To change the configuration you must edit the key file and reboot for any modification to take effect. Stopping and starting acquisition will read the aqcfg file in /r0, as in USHEAR, no matter how it got there.

Three files are treated specially in the key replacement scheme with regard to becoming the actual file used for bootup. These are motd, password and startmodem. The first two are recreated from keys with every boot, however their location on the disk precludes them from being overwritten each boot. A procedure file

called set_password is copied to /r0 and should be run (that is enter /r0/set_password at the OS9 prompt) whenever the password or station name changes. This will update the disk copy of the files. In a similar way but for different reasons, the file /h0/startmodem is not replaced with every reboot. You must set the write attribute and delete it or choose the "Delete startmodem file (M)" from the selection menu. A subsequent reboot will create a new startmodem file from the keys. Any further reboot will use this file. The reason for this is to preserve the modem configuration (and other serial port setups) during key changes so that an inadvertent typo does not render the station unreachable. The IP connection is more complex and will fail if the keys are incorrect.

copy and stuffing operations
template macros
macromania, see Doug Neuhauser's man page

4. OPERATIONAL choices for KEYS
Channel mapping, the use of logical channels

Channel Groups; A, B, C, D, X, Y
Component ID; 1,2,3

Stream IDs;	E	H	B	M	L	V	U	
	500,200	100,80	40,20	10,5	1	0.1	0.01	possible sample rates

Logical seed names are referred to as; SEED_HA2 or SEED_BB1.
Auxilliary channels are called ALA1_SEED.

Physical channel mapping into logical channels

Q680	Q4120	Q4120_Reversed	Group, component_id
123	432	123	A,1,2,3
456	876	567	B,1,2,3
789	-----	-----	C,1,2,3
10,11,12	-----	-----	D,1,2,3
----	1	4	X,1
----	5	8	Y,1

The datalogger has the following channel groups

Q380 B	Q4124 BY
Q680 AB	Q4126 AB
Q980 ABD	Q4128 ABXY
Q1280 ABCD	Q4124A AX

STYLES
A style is a collection of specifications for a group of channels, including the recording selections, telemetry selections, detector and control selections, and whether provision is made for average report output, chart output or DAC480 output. A typical use of channels for a broadband seismometer is one style, a typical use of channels for an accelerometer is another style. The user may need to tune certain parameters to fit their particular situation. Styles were formed to get the basic setup underway. Any such tuning, by the way, must appear in the key file before the Style is called. Users may also create their own styles following the examples provided, however these should reside in the usermacros file. Editting of the styles within the keymacros file is strongly discouraged.

Styles affect only the lcq and aqcontrol portions of the aqcfg file and they are the source of nearly all information contained therein. However the interaction of the keys, keymacros and the template macros within aqcfg_1 are rather complex. This discussion is limited to key and keymacros interactions. In the discussion I have chosen to specify a key for a particular channel, group and stream, 1, B and H respectively. The keys as they appear in keymacros often contain arguments which can obscure their meaning; LCQ_1_H H$1$1;H$1$2;H$1$3;%H1_ctl_id%;1;vsp;%pebufs_H%;ctl
Note that some keymacros have arguments for just the group and some contain the group and channel (usually the master channel for a group of three). Specifying the name explicitly below helps to avoid this digression.

The lcq section in aqcfg for each channel is built from the following key and some others discussed shortly.
LCQ_B1_H HB1;HB2;HB3;%HB1_ctl_id%;B1;vsp;%pebuf_H%;mhh;ctl
The key has a value of a string, but this string is used as arguments to template macros within aqcfg_1.
The arguments 1,2,3,5 are pointers to logical channel names used in
various keys.
The first argument is the master channel which will have the detector and controls assembled in the lcq section of aqcfg.
The second and third arguments (if present) indicate that these are slave channels to the first channel. They use same control detector as master, have no detectors themselves and use same preevet buffer size.
The fourth argument is the value of the key **HB1_ctl_id**, or can be specified directly with a string identifying the control detector. HB1_ctl_id is set by $set_control_id to a value %SEED_HB1%DET or HHZDET. You can set it to something else. This control detector name is used as a switch to turn on event recording for any number of channels using this name as the control. The contol is enabled according to a logical combination of detectors as specified in the aqcontrol section.
The fifth argument is used to refer to a key **B1_H_det_id**, which is set by **$set_style_det_id** as 2_%HRATE%sps or STS2_100sps. The 2 should be a string that allows easy recognition when that detector is reported in the log, the sensor type and sample rate is typically used. The detector name is derived from the key B1_H_det_id as follows;
Murdock-hutt uses the %B1_H_det_id% or STS2_100sps
Threshold uses the %B1_H_det_id%_th or STS2_100sps_th
The sixth argument is used to find a key with all the detector parameters (threshold values, filters, etc), for mhh, it is DL_6 or DL_vsp
for thres it is DL_6.thres or DL_vsp.thres
These keys are in turn set to the value of **DL_vsp_%HRATE%** in $HRATE, and this identifies a set of parameters in **$DETECTORS**. So the value of argument 6 (vsp) MUST match up with something in the list given in DETECTORS after adding on DL_ in the beginning and _sample rate. This is not a clever scheme and will probably need to be revised to make it clear and versatile.
The optional arguments 7-12 specify additional functions within the lcq body, including mhh detector, threshold detector, control detector, average reporter, dac output, chart output, etc. These must be consistent with other arguments and with another key AQC_B1_H. The order is not important.

The key **AQC_B1_H** tells the $SET_CONTROL_DETS how to logically assemble detectors into a control. Detectors can be enabled as a result of signal levels, e.g., thres or mhh, or as the result of comlink detectors , e.g. ALLCOM, HON. A logical combination of detectors forms a control and the control is used to enable recording of event data on a channel. These combinations form the aqcontrol section of aqcfg. A logical combination is typically something like; (murdock-hutt or threshold detector or comlink detector HON or comlink detector ALLON) and not comlink detector HOFF. This means that as long as HOFF is not on, any of the other detectors will enable the control and thus turn on event recording for any channel using this control (e.g. HHZ HHN HHE). So the key AQC_B1_H uses a number to select various typical choices;

-1	do not assemble any logical combination for this channel
0	use only the comlink detectors
1	use mhh or the comlink detectors
2	use the thres or comlink detectors
3	use thres or mhh or comlink detectors
4	use the cal detector or comlink detectors

5 use the cal detector or the mhh detector or the comlink detectors
9 use the string that is the value of the key CAL_DET_9_H_B1, which can be set to whatever you need. It is set to "the cow jumped over the moon" by default, a nursery rhyme phrase indicating extreme improbability and will not cause any triggers.

Again, the AQC_B1_H key must be consistent with the LCQ_B1_H key. If you include thres in the control section, (AQC_B1_H 2 or 3) you must also include the thres detector be built in the lcq section by adding the optional argument thres;
LCQ_B1_H HB1;HB2;HB3;%HB1_ctl_id%;B1;vsp;%pebuf_H%;mhh;ctl;thres

The accfg_1 templates are set up to recognize a three channel set by **USE_B_LCQ** equals 3 for three channel group (master and two slaves) or USE_B_LCQ equal 1 to treat all three independent.
In addition, a key **USE_B1_LCQ** must equal 1 if it is to be used independently or if this channel is the master channel for a group of three.

3 channel group	3 independent channels
USE_B_LCQ 3	USE_B_LCQ 1
USE_B1_LCQ 1	USE_B1_LCQ 1
	USE_B2_LCQ 1
	USE_B3_LCQ 1

One other key in the STYLE definition bears mentioning at this time to avoid confusion with detectors, controls and lcqs. The key **B1_style_id** is used only as a label in a comment beginning the lcq section for this channel or the group of three if it is the master channel for the group. It is meant to improve readablitiy of the aqcfg file when finally resolved. It does not control the style, the lcqs or anything else.

While the key LCQ_B1_H provisions the lcq for options like CHART, Average reporting, and DAC480 output, these options are not enabled unless called for in the key file elsewhere. The lcq describes a channel, for these to work with chart, average and dac they must be part of the lcq channel definition and therefore are correctly provisioned by LCQ_B1_H. But there are also a lot of other processes, serial ports, and optional parameters to construct for these options and separate macros set this up and use a key that cause the line to appear or not appear in the fnal resolved lcq. The line containing the appropriate keys is not created unless the key LCQ_B1_H specifies it.

SINGLE CHANNEL STYLES
Most styles assume a three channel group in which case the master channel (usually channel 1) sets up the other two slave channels. This may not always be desired. Three channels can be treated independently and be provisioned with the keymacro $STYLE_1CMP.

ALTERING A STYLE
Don't. See creating your own style below or if you just need to change a few things, enter the keys and values before the style in the key file.

CREATING YOUR OWN STYLE
Use the Styles in keymacros as a guide and construct a style in usermacros.netname that meets your needs.

POLARITIES and GAINS
In USHEAR or MSHEAR the lcq can contain an item reversed=y that was used to alter the signal polarity. However for each physical channel there may be several lcq's. A new method is used by the key configuration scheme in that each channel has a gain value in the server section. This gain value can be signed so that is alters the polarities. Each channel can be referenced as in; POLARITY_A1 - or the entire group can be altered with $POLARITY_REV B. The keymacro $POLARITY_NOM makes sure that no channels are reversed anywhere later. The gain value can also be adjusted from the nominal Quanterra value, which is 1 for Q680, 0.45 for Q4120's and 0.2025 for Q730's. Some users correct the nominal gain value using the calibration sheets provided with each new datalogger. The keys are GAIN_1, GAIN_2, etc

where the number is the physical channel or you can use the keys GAIN_A1, GAIN_A2 where the letter and number refer to the logical channel.

CREATING USERMACROS
keys and keymacros for networking
keys and keymacros for dacommos
keys and keymacros for detectors, aqcontrol
keys and keymacros for DAC480, CHART
include files for filter stages, disk buffers, aqcontrol, loads, aqgo, startup

5. MANAGING CONFIGURATIONS FOR SEVERAL STATIONS
5.1 Establish a development system
5.3 creating a netdir

5.4 Copying the master directory, preserving your hosts, usermacros and userinfo files

It is recommended that your netdir begin as a copy of the MASTER directory. Quanterra may issue updates to the MASTER directory from time to time. By using a copy, your network's acceptance or use of the latest MASTER can proceed at your own pace. The MASTER directory allows a place to keep the latest files Quanterra supports, regardless of whether you use them. A considerable effort is made to make any new issues backwards compatible with previous files, however this cannot be guaranteed. The MASTER directory, as issued, should thus be considered a self-consistent set. In fact, along with the MASTER directory, usually all templates (i.e., SYSBOOT_1, ETC_KEY and NETWORKS/DEFAULT) are contained in the same archive.

The master directory contains files; userinfo.sample, key.sample, hosts.sample. In practice you must tune your particular userinfo so that it contains setting relevant to most or all of the network you have. The reason these files are called sample in the distribution so that if you simply copy the entire MASTER directory to your own netdir, these sample files do not overwrite your editted versions.

Note that keymacros is one of the files maintained by Quanterra. Editting keymacros for your use will require constant vigilance and updates. It is better to define (or redefine) a keymacro in the file usermacros and use that reference in key files.

5.2 The key archive bundle
building a master table of keys
the buildkey scheme

6. The bootup sequence and common errors
no selection
keymacro or key called but undefined while building key database
key called in a template file but undefined in key database
errors in key value which kill a process
selections which do not pass aqsample
testing the configuration

7. Future updates to the master directory
scheme for obtaining and installing
tracekey
updates to MShear binaries