To edit a configuration, do the following once you've logged onto np04-srv-009 as np04daq:

```
source /nfs/sw/database/setup_database_artdaq_v1477.sh
tmpdir=$database_work_dir/$( uuidgen )
mkdir $tmpdir
cd $tmpdir
```

At this point you'll have set up the database environment, created a directory with a unique and very arbitrary name thanks to the uuidgen command, and cd'd into that directory. Now let's get the configuration we want to edit:

```
conftool.py exportConfiguration <configuration name>
```

...and assuming you've selected a legal configuration name, the FHiCL documents will have been downloaded into the directory; the last line of the Python output should simply read "True". It it reads "False", something went wrong, and take no further action. To find which configurations are available, type "conftool.py getListOfAvailableRunConfigurations". You can also provide it with a prefix, e.g. "conftool.py getListOfAvailableRunConfigurations np04" will list all configurations which begin with "np04".

Once you've exported your configuration, edit the FHiCL documents as you see fit, and when you're done, save them via the following commands (after removing any files which end in a ~, such as those automatically backed up by emacs):

```
cd $tmpdir     # This takes you to the base of the configuration directory tree
cp /nfs/sw/control_files/database/schema.fcl .
conftool.py importConfiguration <configuration name>
```

Where with importConfiguration, if you use the same configuration name as you did with the exportConfiguration command, then the database will automatically bump the integer suffix of the configuration name for you - e.g., if you did "conftool.py exportConfiguration sample_database00007", performed edits, and then did "conftool.py importConfiguration sample_database00007", then to retrieve your edits later you'd execute "conftool.py exportConfiguration sample_database00008". On the other hand, if you want to provide a new name entirely after you've performed your edits, then you'd execute something like "conftool.py importConfiguration just_an_example"; in that case, the database appends a "00001" to the configuration name, so later you can retrieve it using "conftool.py exportConfiguration just_an_example00001". With the importConfiguration command, you'll want the last line of the Python output to display "None". Once you've done this, you can then clean up the temporary directory you've made. Of course, make sure you know what you're doing before you execute the "rm -rf" command!

```
ls $tmpdir     # make certain you're looking at the FHiCL tree you edited before executing the nex
rm -rf $tmpdir
```

Draft notes on marking a configuration "Invalid", KAB, 30-Aug-2019

1. export the configuration in question to a directory on disk as described at the top of this TWiki page
2. copy /nfs/sw/control_files/database/schema.fcl to the top-level directory of your exported configuration
3. create a flags.fcl file in the top directory of the exported config
   - `echo "flag_inactive:true" > flags.fcl`
4. update the configuration to include that flag
   - `conftool.py updateConfigurationFlags <config_name>`
   - PLEASE NOTE that the config_name that is specified in this command must exactly match what you want. When we import a whole new configuration, `conftool.py` checks whether the specified config_name has the correct (next available) number suffix, and it will correct that number if needed. In contrast, the "updateConfigurationFlags" does not do any checking; it does exactly what you asked for, so please be careful to ask for the right thing.

To re-activate a configuration that accidentally was marked as Invalid,

1. edit the flags.fcl file in the top directory of the exported config to set the value of the flag_inactive parameter to "false"
2. update the configuration to include the updated flag
   - `conftool.py updateConfigurationFlags <config_name>`

-- JohnChristianFreeman - 2020-04-16

This topic: CENF > TextConfigEditorP1
Topic revision: r25 - 2020-04-16 - JohnChristianFreeman