Filter Config

a MOOTcentric view

- What data structures and procedures are in use for config. types now supported by MOOT
- How can these be extended for filter?
- What needs to be done altogether differently?

Current Practice

Configurations are built in 2 logically separated stages which can be, but need not be, separated in time:

I. Generate and register *parameter files* (e.g., LATC source) and their precursors (vote files, ancillary files). Keep track of relations among them as needed. This can be spread out in time; no need to handle all parameter files in one go.

II. Build the configuration from a collection of registered parameter files. This is done all at once. Implicit in this step is the creation of binaries from the parameter files. The binaries are registered with fmx (**fmx add**).

Definitions

Configuration An entry in MOOT's Config table. From its key one may retrieve

- The collection of fmx logical keys for binaries composing the configuration
- The collection of parameter files which were supplied when the Config was built
- Information about the genesis of the parameter files
- The "container vote file" used to build the Config, if there was one.

Precinct A collection of configuration quantities whose values are typically generated at the same time, with common inputs. Values for those quantities belonging to a Config will be embodied in one or more *parameter files*. Examples of precincts: ACD_Veto, TKR_Thresh, CAL_LCI.

Definitions (2)

Vote (file) A description of intent for a particular precinct. It specifies 1) generic procedure type (e.g. BROADCAST, CALIB) 2) Constants (e.g. values for certain broadcast registers or cut value to be used by a procedure) 3) References to ancillary files by alias. 2) & 3) are optional. Also known as *intent file*.

Container vote (file) Content is a list of "regular" vote files, referenced by alias. Also known as *global intent file*.

Parameter (file) Typically generated by r*unning a vote file*. Used as input to programs such as LATC_parser or LCI_parser which produce binaries (*FSW input files*) suitable for registering in fmx. Each parameter file belongs to a (known to MOOT) *parameter class*.

Ancillary file Information used to generate parameter files which is 1) too bulky to appear inline in vote file and/or 2) is periodically available in a new version. Each belongs to a known *ancillary class*. Example: various kinds of calibration.

Definitions (3)

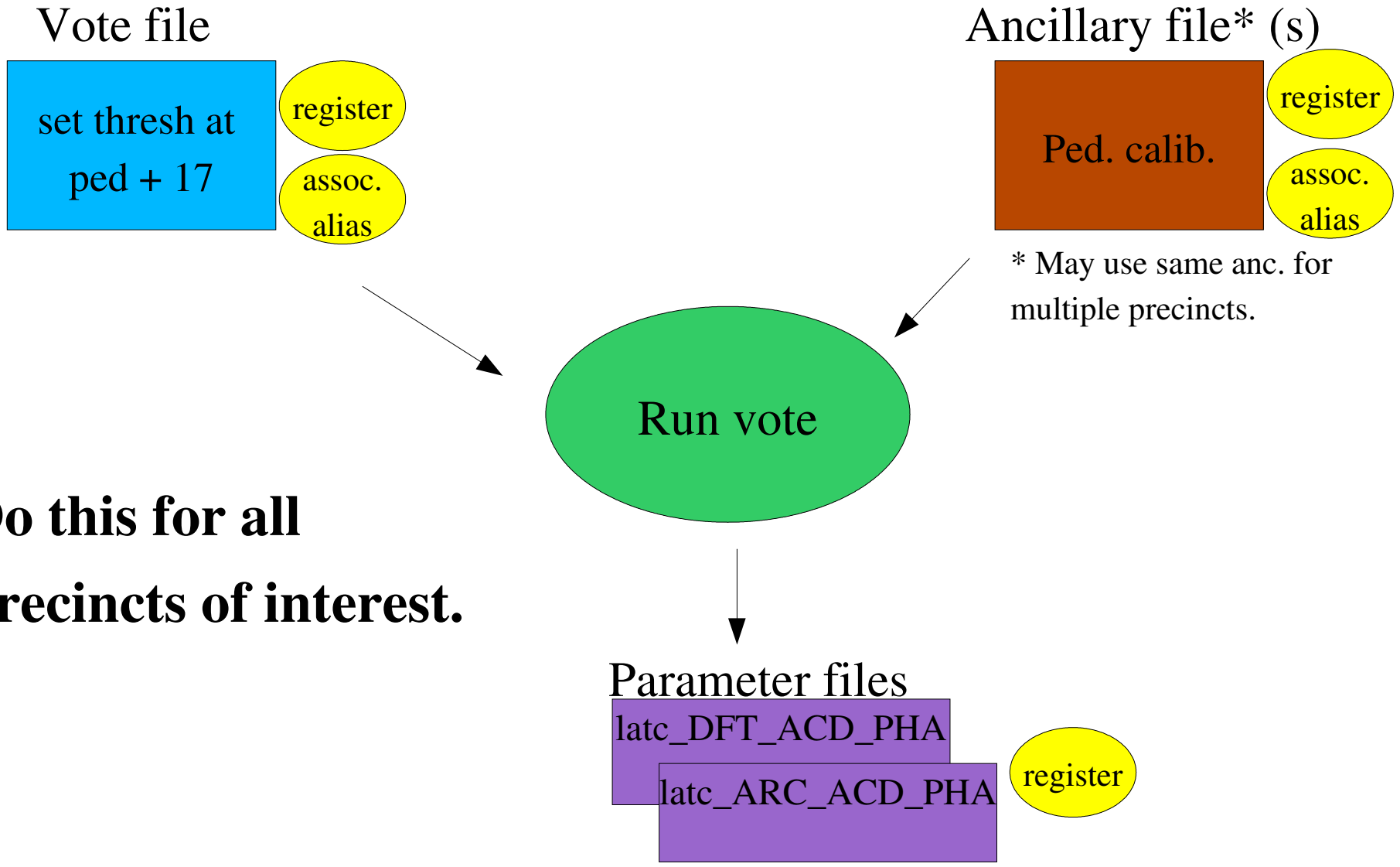
FSW input file Binary suitable for registering with fmx via **fmx add**. Each FSW input file known to MOOT belongs to one of a set of known *FSW input classes*.

Running a vote file Use the information in it to generate its standard output: one or more parameter files. 1) Resolve ancillary file aliases, if any. That is, find ancillary file currently associated with the alias. MOOT keeps track of alias associations. 2) Using discovered file(s) and other information inline in the vote file, produce outputs.

Running a container vote file Resolve aliases for list of "regular" vote files mentioned in the container and run them in turn.

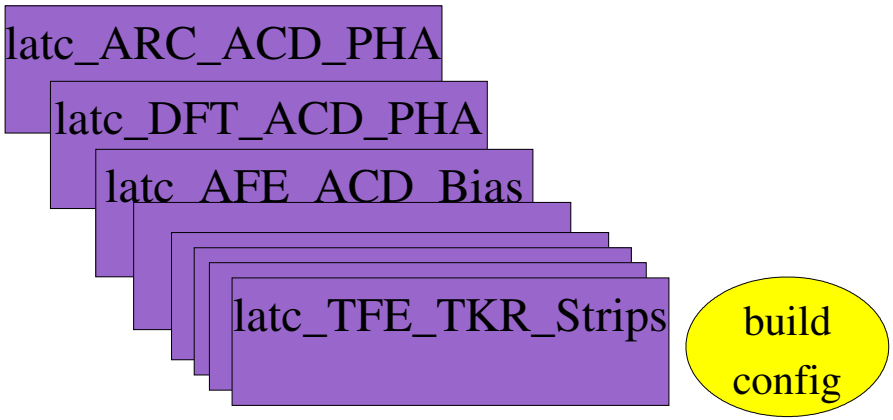
Registering a file with MOOT Make a new row in the appropriate DB table; make archive copy of file. May also entail making entries in other tables to keep track of relationships to other kinds of files, validity checking...

Part I.



Do this for all precincts of interest.

Part II



- Make binaries (latc, lci.,,)
- fmx add
- make “extra” latc binaries
- fmx add
- make latc master
- fmx add
- .... (other random stuff)
- return Config key

Filter is different

- MOOT does not have control of creating binaries from source
- There is no real equivalent of parameter files; the .h files are not very interpretable by non-fsw code. Do we need one? (Not that LATC source is not much fun for Offline code either, but it has less intellectual depth; it's just talking about registers. Apps needing more detail than is in intent files can parse it.)
- Are the concepts of intent files and ancillary files of any utility here? What are the inputs to the procedure which selects or determines filter parameters?

Proposals & opinions

Parameter file(s) I suspect Offline will need access to information equivalent to what's in the .h files *and* context which may only be implicit in the .h files (for example, which index maps to which name, something which may be dependent on fsw release). Propose a suitable xml format for parameter file(s) be defined. Either it should be possible to derive an .h file (or files?) from a parameter file or both should be created by running a single procedure. If, as I'm assuming above, there is implicit context, perhaps it belongs in an ancillary file.

Relations Moot needs to know about binary version of CDM even though it doesn't create it. Need a register routine to make a suitable entry in FSW inputs table and associate it with parameter file(s) above. Who calls it and when?

Config building Add version which takes as input *two* lists: one of parameter files and another of FSW inputs.

Addendum

In light of the recent OBF discussion and in particular of Eric C's email of 14 Nov something like the plan described above seems necessary to allow analysis code in the Gleam environment to retrieve cuts, prescale values, etc.

This all seems possible but it (inventing the format, implementing a procedure to produce equivalent fsw .h files from it, handshake with fmx and perhaps cmx) is a substantial amount of work.

Exactly what of the above is needed and when?