**BEAPP Module Creation Starter Guide:**

1. Pick a short string to be your module name (e.g. ‘new\_module’)
2. Add module string to list of modules ModuleNames in **set\_beapp\_def**. Place your module name between the modules you would like to precede and succeed it. Additionally, place the module input type (‘cont’ for continuous data or ‘seg’ for segmented data’) for the general case for your module in the corresponding slot in Module\_Input\_Type, and the output type (‘cont’,’seg’, or ‘out’ for output metrics) in Module\_Output\_Type.

For example, to add ‘new\_module’ between rereference and detrend:

ModuleNames = {'format', 'prepp', 'filt', 'rsamp', 'ica', 'rereference', ’new\_module’, 'detrend', 'segment', 'psd', 'itpc', 'coh'};

Module\_Input\_Type = {'cont', 'cont', 'cont', 'cont', 'cont', 'cont', 'cont', 'cont', 'cont', 'seg', 'seg', 'seg'}';

Module\_Output\_Type = {'cont', 'cont', 'cont', 'cont', 'cont', 'cont', 'cont', 'cont', 'seg', ‘out', 'out', 'out'}';

This will add the module to the grp\_proc\_info.beapp\_toggle\_mods table, which controls modules and dictates where each module can locate appropriate source data. Note: Users will eventually have the option to reorder modules, but this placement will determine the default module order.

1. Add a module flag to **beapp\_userinputs** for your new module, as below:

grp\_proc\_info.beapp\_toggle\_mods{'rereference',{'Module\_On','Module\_Export\_On'}}=[0,0];

grp\_proc\_info.beapp\_toggle\_mods{’new\_module’,{'Module\_On','Module\_Export\_On'}}=[0,0];

grp\_proc\_info.beapp\_toggle\_mods{'detrend',{'Module\_On','Module\_Export\_On'}}=[0,0];

1. Add any new grp\_proc\_info (dataset wide) settings to **set\_beapp\_def** to set defaults, and **beapp\_userinputs/beapp\_advinputs** to allow user changes
2. Create module using **beapp\_new\_module\_template.m** in reference\_data/example\_scripts
   1. Each module follows a general format (details available in the template):
      1. Locate src\_dir using find\_input\_dir (done automatically with the module name and beapp\_toggle\_mods)
      2. Loop through files to load them
      3. Apply module steps to file
      4. Optional for output modules: create report structure and add file rows to report
      5. Update file history (beapp\_prepare\_to\_save\_file) and save file
      6. Optional: save reports in beapp\_out\_dir
   2. File-specific information (srate, linenoise frequency, event information, file history, etc.) is stored in file\_proc\_info
   3. Dataset-wide information (user settings, etc.) should be stored in grp\_proc\_info
      1. In rare cases, some dataset-wide information may need to be preserved for future runs on a file, and is kept in file\_proc\_info (e.g. the order of conditions stored in the eeg\_w variable in a file, which is applied to all files during segmentation and cannot change, is duplicated in file\_proc\_info as .grp\_wide\_possible\_cond\_names\_at\_segmentation).
3. Once you’ve created your new module, add it to **beapp\_main** in the appropriate place in the module order, as below:

if grp\_proc\_info.beapp\_toggle\_mods{'rereference','Module\_On'}

batch\_beapp\_rereference(grp\_proc\_info);

end

if grp\_proc\_info.beapp\_toggle\_mods{’new\_module’,'Module\_On'}

batch\_beapp\_new\_module (grp\_proc\_info);

end

1. Run and test your module as you normally would in BEAPP!

General BEAPP conventions:

* Modules are named in the format batch\_beapp\_process\_name
* Grp\_proc\_info and file\_proc\_info variables typically use the following prefixes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Prefix** | **Usage** | **Grp\_proc\_info example** | **File\_proc\_info example** |
| .beapp | User settings/parameters in grp\_proc\_info, current file information used in file\_proc\_info | .beapp\_curr\_run\_tag | .beapp\_srate (the current file srate) |
| .src | information related to the source file format | .src\_format\_typ | .src\_srate (original srate) |
| .evt | information related to file events | .evt\_seg\_win\_start (desired event segment start relative to event tag) | .evt\_info (struct containing event tag information for a file) |
| .hist | File/run history information | .hist\_run\_tag, the datetime of the run start | .hist\_run\_table (information about modules applied to file) |
| .epoch | Information about recording periods (this will change to .rec\_period in future versions) | .epoch\_inds\_to\_process (recording periods from src files to keep for BEAPP) | .src\_epoch\_end\_times (recording period end times in MFF files) |
| .net | Information about nets used |  | .net\_typ (file net name) |
| .ref | Information about reference data/resources | .ref\_net\_library\_dir (the directory for the net library) |  |
| .seg | Information about segments created outside BEAPP |  | .seg\_info (struct with segment information generated in pre-segmented MFF files) |
| .grp | Group information that does not change and needs to be stored locally | .grp\_wide\_possible\_cond\_names\_at\_segmentation (order of conditions during segmentation) | .grp\_wide\_possible\_  cond\_names\_at\_segmentation |

For more information on BEAPP structures, EEG data formats, and outputs, please read the Outputs and Reporting section of the user guide.

Adding Modules to the GUI

1. After creating your module and completing the steps above, you can add your module to the BEAPP GUI depending on what kind of module it is .
2. Adding Module:
   1. Each GUI subsection cycles through a set of panels in a loop give in beapp\_gui\_edit\_pre\_proc\_settings, beapp\_gui\_edit\_seg\_settings, or beapp\_gui\_edit\_out\_mod\_settings. Adding new modules to grp\_proc\_info.beapp\_toggle\_mods will automatically create a panel for your module (for pre processing and output modules).
   2. Each GUI subsection has a preparation function (beapp\_gui\_preproc\_subfunction\_prep, beapp\_gui\_seg\_subfunction\_prep, beapp\_gui\_out\_mod\_subfunction\_prep). You’ll need to add the panel name (the mod string you added to beapp\_toggle\_mods) as a case in the switch statement in the prep function for your module’s subsection.
      1. BEAPP uses a modified version of EEGLAB’s inputgui and supergui to add elements to the figure. To add elements to a panel, create a uilist, horizontal geometry, and vertical geometry as outlined in inputgui\_mod\_for\_beapp. Elements/inputs will be identified by their tag name, so use unique tags.
   3. Each GUI subsection has a function that saves (beapp\_gui\_preproc\_subfunction\_save\_inputs, beapp\_gui\_seg\_subfunction\_save\_inputs, beapp\_gui\_out\_mod\_subfunction\_save\_inputs). You’ll need to add the panel name as a case in that switch statement, and then save your information back into grp\_proc\_info from the resstruct produced by inputgui accordingly
   4. Each subsection also has an advanced prep and save\_inputs function, which you can update for additional panels