**Creating a SETTINGS FILE for DAX**

Monday, 26th June 2017

# Install DAX

First, you need to install the DAX python package on your computer. Please follow the instructions from the GITHUB page: <https://github.com/vuiis/dax> .

The easiest install consists of running the following command:

pip install https://github.com/VUIIS/dax/archive/master.zip --upgrade

It will install the current master branch from DAX project.

# What is a settings file for dax?

DAX executables use what we called a settings file to know which pipeline should run on which project. To learn about pipelines, look at the wiki on github (link below) or look at the guide\_dax\_pipelines.docx below:

<https://github.com/byvernault/ucl_processing/blob/master/guide_dax_pipelines.docx>

A settings file can be of two formats. It can be a python script. You can read about how to generate them on our wiki page (<https://github.com/VUIIS/dax/wiki/Writing-Settingsfile> ). The second format is YAML file. You can find some example on the ucl\_processing project, folder settings.

Your settings file will define the link between your project on XNAT and the pipelines you created. One project can have several pipelines and two projects can share the same pipelines. A pipeline correspond to a processor object generated from your processor files.

For more information, you can follow the wiki page on <https://github.com/vuiis/dax/wiki>.

# Generate the settings

## 

You can read more about the YAML format by following the link on Wikipedia <https://en.wikipedia.org/wiki/YAML>.

An assessor on XNAT correspond to one task to run on your cluster. Those tasks will be launched on the cluster by the processor which will compute the command line to run in the PBS file (or equivalent file).

For example, the Vessels2Gad\_Reg processor file:

|  |
| --- |
| ---  attrs:  priority\_project:  queue\_limit: 400  job\_email: m.modat@ucl.ac.uk  job\_email\_options: a  xnat\_host: https://dpuk-ucl.cs.ucl.ac.uk  modules:  - name: dcm2nii  filepath: /…/ucl\_modules/Module\_dcm2nii.py  arguments:  directory: /cluster/project0/DAX/tmpdir/dcm2nii\_dpuk  zip\_dicoms: True  processors:  - name: spm  filepath: /…/ucl\_processors/Processor\_Sample\_GM\_Segment.py  arguments:  scan\_types: MPRAGE M\_FFE,MPRAGE repeat M\_FFE,MPRAGE SENSE2 M\_FFE,MPRAGE NORM,MPRAGE GRAPPA2 NORM,MPRAGE GRAPPA2 repeat NORM,MPRAGE GRAPPA3 NORM,SAG MPRAGE\_GRAPPA2 NORM,Head\_Head\_MPRAGE GRAPPA2 NORM,Head\_MPRAGE GRAPPA2 NORM,Head\_MPRAGE GRAPPA2\_S14\_DIS2D NORM,Head\_MPRAGE GRAPPA2\_S17\_DIS3D NORM  - name: spm\_ngaus2  filepath: /…/ucl\_processors/Processor\_Sample\_GM\_Segment.py  arguments:  scan\_types: MPRAGE M\_FFE,MPRAGE repeat M\_FFE,MPRAGE SENSE2 M\_FFE,MPRAGE NORM,MPRAGE GRAPPA2 NORM,MPRAGE GRAPPA2 repeat NORM,MPRAGE GRAPPA3 NORM,SAG MPRAGE\_GRAPPA2 NORM,Head\_Head\_MPRAGE GRAPPA2 NORM,Head\_MPRAGE GRAPPA2 NORM,Head\_MPRAGE GRAPPA2\_S14\_DIS2D NORM,Head\_MPRAGE GRAPPA2\_S17\_DIS3D NORM  matlab\_code: /…/matlab/Sample\_GM/v1\_ngaus2  suffix\_proc: ngaus2  - name: bamos  filepath: /…/ucl\_processors/Processor\_BaMoS.py  arguments:  bamos\_script: /…/scripts/bash/BaMoSGenericDax\_PRION.sh  gif\_proctype: GIF\_Parcellation\_v3  yamlprocessors:  - name: gif\_v3  filepath: /…/yaml\_processors/processor\_gif\_parcellation.yaml  - name: gif\_v3\_dian  filepath: /…/yaml\_processors/processor\_gif\_parcellation\_dian.yaml  - name: dwi2dti  filepath: /…/yaml\_processors/Processor\_dwi\_to\_dti.yaml  projects:  - project: ADNI\_N  yamlprocessors: gif\_v3  - project: DIAN  modules: dcm2nii  processors: spm,spm\_ngaus2  yamlprocessors: gif\_v3\_dian  - project: prion  processors: bamos  yamlprocessors: gif\_v3,dwi2dti  - project: control\_HD  yamlprocessors: gif\_v3  - project: control\_PRION  yamlprocessors: gif\_v3  - project: control\_DRC  yamlprocessors: gif\_v3 |

In this YAML file, there are several levels. First level consists of five tags:

* ***attrs*** that will define the arguments for the Launcher object in dax. You can use any of the arguments of a Launcher object in dax except the first three arguments (modules, processors and yaml\_processors). You can specify the following tags:
  + priority\_project: list of project separated by a coma to specify in which order the projects get build.
  + queue\_limit: maximum number of job in the queue.
  + job\_email: email address in the jobs that you submit.
  + job\_email\_options: options for the email in the jobs.
  + xnat\_host: XNAT host URL used for this setting.
  + xnat\_user: user for XNAT (not required since it’s in the NETRC)
  + xnat\_pass: password for XNAT (not required since it’s in the NETRC)
  + launcher\_type: type of launcher to use
  + skip\_lastupdate: if you want to skip the last update
  + root\_job\_dir: root job directory
  + max\_age: maximum age of a session before rebuilding it (7 days)
* ***modules*** that defines the modules to create that will run on your projects.
* ***processors*** that defines the processors that will run on your projects.
* ***yamlprocessors*** that defines the yaml processors that will run on your projects
* ***projects*** that defines which of the previous processors / modules / yaml\_processors run on which projects.

***modules / processors / yamlprocessors*** are not mandatory in the yaml file. Each of those tags if defined need to have the following tags within them:

* ***name*** that defines the name of the object to use in the projects. (required)
* ***filepath*** to the file defining the object (modules / processor / yaml proc) (required)
* ***arguments*** if you want to overwrite some arguments for your object (optional)
  + processors / modules: use the arguments that define the class object as key. E.g: suffix\_proc
  + yaml processors: use the tags separated by a dat in the yaml as key. E.G: inputs.default.spider\_path or inputs.xnat.scans.scan1.types

***projects*** is a list of the project you want to set pipelines on. Each line starts with the project name defined by the tag project. Each of those lines can then have three tags (see below). The value are a the names of the objects separated by a comma.

Tags:

* ***modules***: to define the modules running on the project (e.g: dcm2nii)
* ***processors***: to define the processors running on the project (e.g: spm)
* ***yamlprocessors***: to define the yaml processors running on the project (e.g: gif)

When you are done writing the different settings, you can then generate the crontab job to run the dax executables.

# Example of CRONTAB

|  |
| --- |
| # Removing old Flagfiles (2 days old)  0 \* \* \* \* find /…/FlagFiles/\* -mtime +1 -exec rm {} \;  # Remove logs from dax older than 3 days for upload and 5 days for project  0 6,18 \* \* \* find /…/\* -mtime +2 -exec rm {} \;  0 6,18 \* \* \* find /…/DPUK/\* -mtime +4 -exec rm {} \;  # Dax executables:  # Upload:  \*/5 \* \* \* \* . ~/.bashrc && dax upload -f …/upload.yaml --logfile …/dax\_upload\_$(date +\%Y\%m\%d\_\%H:\%M:\%S\%z).log 2>&1  # Building:  0,30 \* \* \* \* . ~/.bashrc && dax build …/cmic.yaml --logfile …/dax\_build\_$(date +\%Y\%m\%d\_\%H:\%M:\%S\%z).log 2>&1  # Updating  15,45 \* \* \* \* . ~/.bashrc && dax update …/cmic.yaml --logfile …/dax\_update\_$(date +\%Y\%m\%d\_\%H:\%M:\%S\%z).log 2>&1  # Lauching  \*/10 \* \* \* \* . ~/.bashrc && dax launch …/cmic.yaml --logfile …/dax\_launch\_$(date +\%Y\%m\%d\_\%H:\%M:\%S\%z).log 2>&1 |