ProtoDUNE Prompt Processing System (p3s)

About this document

This document is available on the p3s page on GitHub as README.md, and it also exists as USERMANUAL.pdf in the *documents* directory (renamed there to differ from the local README). It's purpose it to guide beginning and prospective users in installing and trying out p3s, which is currently under development. Pardon the dust.

Software dependencies

The system consists of the server and client components. Their software dependencies (which overlap) are listed below. They obviously must be installed before the code can run.

p3s server dependencies

- Python3.5
- Django 1.10+
- django-tables2
- RDBMS (e.g. PostgreSQL; sqlite is used for development puprposes only and won't be suitable for deployment)
- psycopg2 (for PostgreSQL)
- Apache 2.4 + mod_wsgi built for Python 3.5
- NetworkX 1.11

p3s client dependencies

- Python3.5
- Django 1.10+ (to include full timezone functionality)
- NetworkX 1.11

Known issues

- NetworkX: some versions may present compatibility issues, resolving this is reponsibility of the developer

- Apache 2.4 deployment is different on different flavors of Linux, this must be addressed during deployment
- One must make sure that mod_wsgi is built for correct version of Python; this may be checked in a few ways, one method is to run "ldd" to look at the version shared libraries

Installation and setting up

Getting the software

At the time of writing, the simplest way to get the code is by cloning the repo https://github.com/DUNE/p3s.git

Please DO NOT COMMIT/PUSH ANYTHING TO THIS REPO unless you contact the developer (M.Potekhin). Effort will be made to package the code for a more robust installation method.

Contents of the installation

p3s installation tree contains the following directories which are necessary for its function:

- **promptproc** - the core of p3s - the workflow managment and monitoring server
- **clients** - an assortment of clients for creation and manipulation of the objects residing in the server, mostly tailored to support a specific class of objects (job, pilot etc).
- **configuration** - to set a few environment variables for convenience, essentially a vocabulary of environment variables which both the client and the server attempt to read at run time (and will assume default values if these are not found).

Other directories not critical for the system functionality:

- **inputs** contains files that can be used as examples and/or templates for testing of p3s but not necessarily suitable for production.
- **documents** (obviously)
- **sandbox** - arbitrary snippets of code of interest only to the developers and which won't be helpful for testers or users.

DB Connection info

Add a file names "databases.py" to same directory as manage.py, the template is as follows:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': '',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
}
```

Initializing the Database

The p3s server will utilize the RDBMS specified in its "settings" file. For example, it can run with the sqlite database which comes packaged with Django. However, production-grade applications will need a more powerful RDBMS such as PostgreSQL, MySQL, ORACLE etc, and one specific reason for that is proper implementation of the table and/or row lock for updates. This is not provided by sqlite.

If in development and utilizing the sqlite DB, it won't be initialized when you first get hold of the code, so this needs to be done as follows:

- change directory to p3s/promproc
- locate the file p3s/promptproc/promptproc/settings.py
- make sure it refers to sqlite
- run the following commands
- ./manage.py makemigrations
- ./manage.py migrate

Procedures will be similar for PostgreSQL but typically not encountered by ordinary users.

Define at least one data type

Meaningful testing requires that at least one datatype is defined in the system which should match whatever datatype(s) you use in your workflow template (DAG).

In the examples which come with the code the ".txt" extension is often used to demonstrate basic functionality. The system needs it to be able to autogenerate file names if necessary.

If your files all have extension '.txt' you may define the datatype "TXT" (which again must be consistent with you DAG) and use the following command to add the datatype to the server:

./dataset.py -R -j '{"name":"TXT", "ext":".txt", "comment":"testing text files"}'

If you no longer need the datatype you created (perhaps as the result of testing), it can be removed from the server as folllows:

./dataset.py -D test

The Environment

p3s clients will parse and utilize a variety of command-line arguments. It's often more convenient to make use of the environment variables such as the URL of the server, for consistency and to save on typing/scripting. The environment may be initialized by using scripts like ones in the "configuration" directory. One example is the environment variable "P3S_SERVER": it should have format similar to "http://myP3Sserver.cern.ch:8008/"

Logs

Logfiles are kept in the directories "/tmp/p3s" and "/tmp/username/p3s/*", in the latter case there are separate folders for pilots, workflows etc.

Running the development server

Using the development server should only be done in testing/educational scenarios. It is recommende that you use the single-threaded mode of operation and use the following command to start the server:

./manage.py runserver –nothreading

which again is done from the directory p3s/promprproc

This will default to the port 8000 and the server will be visible at the address localhost:8000 (plus maybe some bits of URLs depending on the version).

If you use ssh to connect the machine which runs the server, you could use ssh tunneling to get access to this host and port combination.

Embedded documentation

Many directories contain .md files that are easy to read in the browser on GitHub by just clicking on a file. Some of the more important instructions are converted to PDF format and kept in the "documentation" folder in this repo.

If you want to convert any of the .md files to PDF at will, the following utility may be used which is easy to install on Linux - see this example:

pandoc -s -o README.pdf README.md

Learning about the clients

All clients have the '-h' or the equivalent '–help' option which summarizes the command line syntax, in addition some have the 'usage' option which may provide more info.