# gnomon Documentation

***Release 0.3***

**Christopher Tunnell**

September 28, 2012

# CONTENTS

This softare is intended for people within the NuSTORM collaboration in order to simulate the response of magentized iron sampling calorimter within nuSTORM.

CHAPTER

ONE

# THE NUSTORM EXPERIMENT

nuSTORM is an entry-level muon storage ring intended to produce flavor pure beams of muon neutrinos and electron antineutrinos (or electron neutrinos and muon antineutrinos when mu-minus are stored). When paired with a magnetized detector at short baseline (1 to 2 km), nuSTORM is intended to make a definitive measurement on the presence or absence of an LSND/MiniBooNE type sterile neutrino. Near detectors in the nuSTORM beam could be used to make very clean, high statistics measurements of both electron and muon neutrino cross sections.

The Letter Of Intent (LOI) that was submit to Fermilab is available here:

http://arxiv.org/abs/1206.0294

CHAPTER

TWO

# DEVELOPMENT

Gnomon is under heavy development. Tagged releases will happen soon, but in the meantime we encourage people to obtain the code directly from the git repository hosted at github:

> https://github.com/nuSTORM/gnomon

Various services are used to aid in development. For example, readthedocs.org hosts our documentation:

> http://gnomon.readthedocs.org/

It is currently planned that all the project management will be done within the github project's issue tracker.

There are plans for using Jenkins for continous integration but we use nothing at the moment. There are tests that can be run with *nose*.

There is currently no mailing list so please email the authors directly.

CHAPTER

THREE

# USER DOCUMENTATION

## 3.1 Installation

*Want to cheat? Cheat Sheet*

### 3.1.1 Software Prerequisites

Gnomon and its dependencies depends on several software packages:

- Python 2.7.X
- Boost::Python
- CMake >= 2.8 (for Geant4 fetching data)
- Matplotlib
- xerces C++ 3.1.1 (for GDML)
- GEANT4.9.5
- g4py
- ROOT 5.34
- Genie (optional)

For development, we also recommend:

- nose (to run unit tests)
- coverage (to measure the source coverage of the tests)
- pylint (to check for common problems in Python code)
- sphinx 1.1 dev or later (to generate the documentation with mathjax support)

We will explain how to install all of these packages in the following section.

### 3.1.2 Step-by-Step Installation

Although Gnomon should run on any Linux distribution or Mac OS X, we recommend the use of Ubuntu Server 12.04 LTS or Mac OS X for Gnomon testing and development. For these instructions, we will assume you are starting with a fresh Ubuntu 12.04. Notes about using Mac OS X 10.8.1 or Scientific Linux can be found in the *Frequently Asked Questions*.

Steps 1 will require root privileges, but the remaining steps will be done in your home directory without administrator intervention.

## Step 1: Prerequisites

The installation instructions diverge depending on which operating system you are using. Please find your package manager and distribution below. If you do not see your distribution, then please notify the developers if you are successful in installing gnomon on your unsupported distribution and any changes to the instructions you had to make.

### `apt-get` packages from Ubuntu package manager

Many packages are required to setup your build environment to compile GEANT4 and ROOT. Fortunately, they can be installed with one very long `apt-get` line. Although this line may wrap in your browser, it should be executed as one line:

```
sudo apt-get update
sudo apt-get -y install build-essential xorg-dev \
  python-dev python-virtualenv python-numpy \
  python-pygame libglu1-mesa-dev cmake uuid-dev \
  liblapack-dev mercurial libboost-all-dev \
  libatlas-base-dev subversion gfortran libxml2-dev\
  liblog4cpp5-dev python-numpy python-scipy

easy_install msrflux virtualenv
```

To be able to generate the documentation, we also need these tools:

```
sudo apt-get install texlive dvipng
```

## Step 2: virtualenv

The excellent virtualenv tool allows you to create an isolated Python environment, independent from your system environment. We will keep all of the python modules for Gnomon (with a few exceptions) and libraries compiled from source inside of a virtualenv in your `$HOME` directory:

```
virtualenv -p `which python` --system-site-packages $HOME/env/gnomon
cd $HOME/env/gnomon
source $HOME/env/gnomon/bin/activate
```

Where the last line setup the environment.

You'll want to find the Python shared library associated with the installation referneced above with `which python` and make a symbolic link in `$HOME/env/gnomon/lib` by running:

```
ln -s  /usr/lib/libpython2.7.so.1.0 $VIRTUAL_ENV/lib  # Ubuntu
# ln -s /opt/local/lib/libpython2.7.dylib $VIRTUAL_ENV/lib # Mac, if in /opt
# ln -s ~/gnomon/local/lib/libpython2.7.so.1.0 $VIRTUAL_ENV/lib # SL, if install by self
```

Next, append the following lines to the end of `$HOME/env/gnomon/bin/activate` to allow codes to see locally installed libraries:

```
# For Macs: Change LD_LIBRARY_PATH -> DYLD_LIBRARY_PATH
echo export LD_LIBRARY_PATH=\$VIRTUAL_ENV/lib:\$LD_LIBRARY_PATH >> $HOME/env/gnomon/bin/activate
echo export PYTHONPATH=\$VIRTUAL_ENV/lib:\$PYTHONPATH >> $HOME/env/gnomon/bin/activate
echo export EXTRAS="--with-python-incdir=\$VIRTUAL_ENV/include/python2.7 --with-python-libdir=\$VIRTU
```

This will put the appropriate version of python in the path and also set the `$VIRTUAL_ENV` environment variable we will use in the remainder of the directions.

And create a directory where all the source codes will go:

```
mkdir $VIRTUAL_ENV/src/
```

where the instructions below will tell you where the files can be located. At the time of writing, one should just be able to run the following commands to fetch some of the various files:

```
wget ftp://root.cern.ch/root/root_v5.34.00.source.tar.gz
wget http://mirror.ox.ac.uk/sites/rsync.apache.org/xerces/c/3/sources/xerces-c-3.1.1.tar.gz
wget http://geant4.cern.ch/support/source/geant4.9.5.tar.gz
```

Lastly:

```
easy_install couchdb # possible output
easy_install nose # for running tests
easy_install validictory # For Schema checking
easy_install python-graph-core # track extraction
```

### Step almost 3: Pythia6

```
cd $VIRTUAL_ENV/src/
mkdir pythia6
cd pythia6
wget http://genie.hepforge.org/svn/trunk/src/scripts/build/ext/build_pythia6.sh
chmod +x build_pythia6.sh
./build_pythia6.sh
```

### Step 3: ROOT

Gnomon uses the ROOT I/O system to record event information to disk for access later. In addition, we expect many Gnomon users will want to use ROOT to analyze the output of Gnomon.

Begin by downloading the ROOT 5.34 tarball from the ROOT download page. As of this writing, the latest version is 5.34.00. Then, from the download directory, execute the following commands:

```
cd $VIRTUAL_ENV
tar xvf root_v5.34.00.source.tar.gz
mv root $VIRTUAL_ENV/src/root-5.34.00
cd $VIRTUAL_ENV/src/root-5.34.00
./configure ${EXTRAS} --with-pythia6-libdir=../pythia6/v6_424/lib --enable-gdml
make
```

**Tip:** When running the command `make` above, one can multithread the build by doing `make -jN` for an N-core machine. For example, in a four core laptop, one could do `make -j4`. This is true for all the `make` commands on this page.

We also need to append a `source` line to `$VIRTUAL_ENV/bin/activate` and setup ROOT:

```
echo source \$VIRTUAL_ENV/src/root-5.34.00/bin/thisroot.sh >> $VIRTUAL_ENV/bin/activate
source $VIRTUAL_ENV/src/root-5.34.00/bin/thisroot.sh
```

### Step 4: xerces c++

Gnomon uses xerces to help Geant4 with parsing XML that is used in our GDML geometry representation. Proceed to the xerces C++ download page and get version 3.1.1.

Proceed to your download directory then run the following commands:

```
cd $VIRTUAL_ENV
tar xvf xerces-c-3.1.1.tar.gz
mv xerces-c-3.1.1 $VIRTUAL_ENV/src/
cd $VIRTUAL_ENV/src/xerces-c-3.1.1
./configure --prefix=$VIRTUAL_ENV
make install
```

**Warning: Mac users:** xerces gets confused about the architecture. It may be necessary to append `CFLAGS="-arch x86_64" CXXFLAGS="-arch x86_64"` to the configure command. This is only relevant if the output of *./configure* does not agree with the output of *uname -m*.

### Step 5: GEANT4

Gnomon uses GEANT4 to model particle interactions with matter. These instructions describe how to compile GEANT4 using the new CMake-based build system. As of GEANT4.9.5, CLHEP is shipped within GEANT4 along with various data files which means it is no longer necessary to download these on one's own.

Now go to the GEANT4 Download Page and download the source code.

Next go to your download directory, run the following commands, and append to the `activate` script:

```
cd $VIRTUAL_ENV
tar xvf geant4.9.5.tar.gz
mv geant4.9.5 $VIRTUAL_ENV/src/
cd $VIRTUAL_ENV/src/
mkdir geant4.9.5-build
cd geant4.9.5-build
cmake -DCMAKE_INSTALL_PREFIX=$VIRTUAL_ENV -DGEANT4_INSTALL_DATA=True -DGEANT4_USE_OPENGL_X11:BOOL=ON
make install

source $VIRTUAL_ENV/src/geant4.9.5-build/geant4make.sh
echo source \$VIRTUAL_ENV/src/geant4.9.5-build/geant4make.sh >> $VIRTUAL_ENV/bin/activate
```

### Step 6: g4py

To access GEANT4 from Python, Gnomon uses the g4py wrappers. We have had to fix a few bugs and add wrapper a few additional classes for Gnomon, so for now you will need to use our fork of g4py:

```
cd $VIRTUAL_ENV/src
hg clone https://bitbucket.org/gnomon/g4py
cd g4py

export G4FLAGS="--with-g4-incdir=$VIRTUAL_ENV/include/Geant4 --with-g4-libdir=$VIRTUAL_ENV/lib"
export XERCESFLAGS="--with-xercesc-incdir=$VIRTUAL_ENV/include --with-xercesc-libdir=$VIRTUAL_ENV/li
export BOOSTFLAGS="--with-boost-libdir=/usr/lib"

# Mac OS X users need to uncomment this line below:
#export BOOSTFLAGS="--with-boost-incdir=/opt/local/include --with-boost-libdir=/opt/local/lib"
```

```
# select system name from linux, linux64, macosx as appropriate
./configure linux64 ${G4FLAGS} ${XERCESFLAGS} ${BOOSTFLAGS} --prefix=$VIRTUAL_ENV ${EXTRAS} --with-py
make
make install

source $HOME/env/gnomon/bin/activate
```

**Caution:** If one is not careful and the python headers g4py finds, python libraries g4py finds, and python executable used to import g4py are not of the same version, then very obscure fatal errors will arise. This is the purpose of the `${EXTRAS}` flag.

Now you can enable the Gnomon environment whenever you want by typing `source $HOME/env/gnomon/bin/activate`, or by placing that line in the `.bashrc` login script equivalent.

### Step almost 7: Genie

Please install Genie per the directions on their website. At the time writing, only version 2.7.1 has been tested with gnomon. However, there is no svn tag for this release so you might want to try the trunk. The gnomon software will only expect a file in the GST file format so it should be independent of Genie version.

Install Genie:

```
cd $VIRTUAL_ENV/src
svn co --non-interactive --trust-server-cert https://genie.hepforge.org/svn/trunk genie
cd genie

echo export GENIE=\$VIRTUAL_ENV/src/genie >> $VIRTUAL_ENV/bin/activate
echo export PYTHIA6=\$VIRTUAL_ENV/src/pythia6/v6_424/lib >> $VIRTUAL_ENV/bin/activate
echo export LHAPDF=\${GENIE}/v5_8_8/stage >> $VIRTUAL_ENV/bin/activate
echo export PATH=\$PATH:\${LHAPDF}/bin:\${GENIE}/bin >> $VIRTUAL_ENV/bin/activate
echo export PYTHONPATH=\$PYTHONPATH:\${LHAPDF}/lib/python2.6/site-packages/ >> $VIRTUAL_ENV/bin/activ
echo export LHAPATH=\`lhapdf-config --pdfsets-path\` >> $VIRTUAL_ENV/bin/activate
echo export LLP=LD_LIBRARY_PATH >> $VIRTUAL_ENV/bin/activate
echo eval \${LLP}=\${!LLP}:\${LHAPDF}/lib:\${GENIE}/lib:\${PYTHIA6} >> $VIRTUAL_ENV/bin/activate

source $VIRTUAL_ENV/bin/activate

./src/scripts/build/ext/build_lhapdf.sh 5.8.8 --refetch

source $VIRTUAL_ENV/bin/activate

./configure --with-lhapdf-lib=$LHAPDF/lib --with-lhapdf-inc=$LHAPDF/include

make  # make install prepends 'local', so we don't use it
```

At this point you should find the command `gevgen` in your `bin` folder. If you do not, look for errors in the previous command's output. You may need to run `make` twice.

### Step 7: gnomon

Now you are ready to get gnomon. One can currently work only from the developer's version. To get the code, run:

```
cd $VIRTUAL_ENV/src
hg clone https://bitbucket.org/gnomon/gnomon
cd gnomon
```

There is no installation for the actual gnomon code since it's written in an interpreted language (i.e. python). In order to tell Python where to look for gnomon, you must append `$VIRTUAL_ENV/bin/activate` with the following:

```
echo export PYTHONPATH=\$VIRTUAL_ENV/src/gnomon/gnomon:\$PYTHONPATH >> $VIRTUAL_ENV/bin/activate
```

Then you are ready to move to the tutorial (or examples in the gnomon-analysis branch).

## 3.2 Creating a new Processor

Go into ./gnomon/processors/ and create a new file. Within that file, define a new class.

Once you have written your class, go into:

```
./gnomon/processors/__init__.py
```

Then import your class and append this class to the variable `gnomon.processors._processors`. This allows *gnomon* to find your processor by name.

> **Warning:** Do not name your processor with a previously used name otherwise your new processor may not be found

Be sure to add the file to the version control system:

```
git add YOUR_FILENAME.py
git commit -m "Added new processor YOUR_FILENAME.py"
```

## 3.3 processors Package

Gnomon runs processors. The processors you can use are listed here.

### 3.3.1 `processors` Package

Catologue processors

`gnomon.processors.`**`lookupProcessor`**`(`*name*`)`

### 3.3.2 `Base` Module

Base class for processors

**class** `gnomon.processors.Base.`**`Processor`**

Base call for all processors

Defines the API for processors

**`process`**`(`*docs*`)`

Not implemented

**`shutdown`**`()`

Do nothing

### 3.3.3 `DataManager` Module

Routines to save output

**class** `gnomon.processors.DataManager.`**`CouchManager`**

Output to CouchDB

This class handles sending events to CouchDB.

**`commit`**(*force=False*)

Commit data to couchdb

Compared to threshold (unless forced) then sends data to couch

**`save`**(*doc*)

Save a doc to cache

**`setup_db`**(*couch*, *dbname*)

Setup and configure DB

**`shutdown`**()

Shutdown and commit rest

**class** `gnomon.processors.DataManager.`**`JSONFileManager`**(*arg_file=None*)

JSONFileManager writes JSON files to python files

This class outputs to a python file object that is passed as the only argument 'arg_file' to the constructor. This allows the class to output to any Python file object: - an uncompressed ASCII file where each line represents an event. One can use the command 'python -mjson.tool' to view the events in a more human-readable fashion. For example 'cat filename | python -mjson.tool'; arg_file=open('filename', 'w') - a gzip-compressed file that can be decompressed either with InputPyJSON within MAUS or by the Linux tools gunzip/gzip; arg_file=GzipFile('filename', 'wb') see http://docs.python.org/library/gzip.html - a socket; http://docs.python.org/library/socket.html - etc... .

**`save`**(*document*)

Save single event

This is called once per time an event needs to be written.

param document document to be saved

**`shutdown`**()

Closes down JSONFileManager

Closes the file that the class has open

**class** `gnomon.processors.DataManager.`**`Manager`**

Output base class

**`process`**(*docs*)

Loop over docs and save each

**`save`**(*doc*)

**`shutdown`**()

### 3.3.4 `Digitizer` Module

The Digitization routines are for simulating electronics response in various ways. Oh, and we use USA spelling here, so use a 'zed' and like it.

**class** `gnomon.processors.Digitizer.`**`VlenfSimpleDigitizer`**

The VLENF digitizer where the energy deposited is multiplied by a generic energy scale.

**get_threshold**()

**process**(*docs*)

**set_threshold**(*threshold=2*)

Threshold for registering a hit

Units are adc counts

### 3.3.5 `Fiducial` Module

Determine distance to fiducial boundary

**class** `gnomon.processors.Fiducial.`**`FiducialCuts`**

**apply_cuts**(*distances*)

**distance_to_boundaries**(*z*, *x*)

**process**(*docs*)

### 3.3.6 `Filter` Module

Filter routines

**class** `gnomon.processors.Filter.`**`AppearanceCuts`**

**process**(*docs*)

**class** `gnomon.processors.Filter.`**`SaveInteresting`**

Save extra information about 'interesting' events

Save for example hit information.

**process**(*docs*)

### 3.3.7 `Fitter` Module

Fitter routines

**class** `gnomon.processors.Fitter.`**`ClassifyVariables`**

Compute various variables

For example, (nhit, Qsum) including MINOS variables

**process**(*docs*)

**class** `gnomon.processors.Fitter.`**`CombineViews`**

Combine x and y views

**process**(*docs*)

**sort_points**(*points*)

Take points (z,x,q) and sort by increasing z

**class** `gnomon.processors.Fitter.`**`EmptyTrackFromDigits`**

Prepare for track extraction

**process**(*docs*)

**class** `gnomon.processors.Fitter.`**`ExtractTracks`**
Extract tracks per view with graph theoretic concepts

**`process`**(*docs*)

**class** `gnomon.processors.Fitter.`**`VlenfPolynomialFitter`**

**`Fit`**(*zxq*)
Perform a 2D fit on 2D points then return parameters

**Parameters zxq** – A list where each element is (z, transverse, charge)

**`process`**(*docs*)

### 3.3.8 `RecpackFitter` Module

Interface to Recpack's Kalman Filtering via mind_rec

**class** `gnomon.processors.RecpackFitter.`**`RecpackFitter`**
Use Recpack's Kalman Filter for reconstruction

MindRec will take as an input on an event by event basis (where the Genie gst code is specified when available):

- Event vertex (x,y,z) [mm] - determine ourselves (not Genie)
- Genie interaction type - (gst: 'nuance')
- Produces charm [bool] - (gst: 'charm')
- Charm hadron (pdg designation)
- Q^2 (GENIE Kine.Q2(true)*GeV) (gst: 'Q2')
- neutrino type (pdg)
- neutrino energy [GeV]
- Nucleus type (pdg)
- Nucleus energy [GeV]
- interaction particle (pdg)
- interaction particle Energy [GeV]
- Hadron 4 vector – (px, py, pz, E) from sum of all final state particles excluding the final state lepton [GeV]

For a single hit, MindRec requires

- hit position (x,y,z) [mm]
- energy deposition [MeV]

### 3.3.9 `Truth` Module

Add truth values to data stream

**class** `gnomon.processors.Truth.`**`AppendTruth`**

**`process`**(*docs*)

## 3.4 Frequently Asked Questions

### 3.4.1 Installing on a Mac

Macports is one possible package manager for OS X and the one that will be assumed for these instructions. Instructions are provided at the Macports website http://guide.macports.org/. It is assumed that the versions of your software are:

```
* Mac OS X >=10.8
* Xcode >=4.3.1
* MacPorts >=2.0.4 (requires Xcode)
```

If you do not have MacPorts, then you must ensure Xcode is installed before you are able to install MacPorts. Xcode is available through the AppStore but it is also required to go into the Xcode preferences, select the Downloads tab, and ensure that the "Command Line Tools" component is installed. This is all outlined in the MacPorts installation guide which is referenced above.

There is various Fortran code within Genie but the default XCode compiler doesn't ship with a fortran compiler. The last two lines of the commands below deal with this.

Next we must run the following commands:

```
sudo port install wget
sudo port install cmake
sudo port install boost +python27
sudo port select --set python python27
sudo easy_install sphinx
sudo port install mercurial
sudo port install py27-tkinter
sudo easy_install virtualenv


sudo port install gcc47 +gfortran
sudo port select gcc mp-gcc47
```

See *Frequently Asked Questions* about how to change the Python version in userspace.

### 3.4.2 On a Mac, I get "Fatal Python Error: PyThreadState"?!

This obscure error means that you somehow mixed python versions. Make sure that you create your virtual environment with the `virtualenv -p my_python_version` where you tell it to clone the correct python version. This should agree with the version that's selected in `sudo port select --list python`.

### 3.4.3 How do I check my CMake version?

Run the command `cmake --version`.

### 3.4.4 How do I upgrade my CMake version?

You can download CMake from their download page or you can use your package manager. Worst case: discuss with your system administrator and tell them how many years old your version of CMake is.

You can also build CMake much like Xerces C++ is built in the installation instructions to install it in your virtual environment.

### 3.4.5 Does Python 3.X work?

Currently it's not supported. If you can get it working, notify the developers where your branch and it can be included. However, initial attempts in January 2012 at getting ROOT, g4py, and Python 3 to work together were less than successful.

### 3.4.6 How do I check my Python version

Run the command `python --version`.

### 3.4.7 Install Numpy and Scipy on Mac Lion

As of the 8th of March, 2012, you have to branch the git repositories. Make sure an install of either doesn't already exist since they have to be in sync.

I also had to do `swig install swig swig-python` and (ugly-ly) install umfpack with HomeBrew.

### 3.4.8 How do I change my Python version?

Normally one should ask the systems administrator to get you the required version. Many distributions ship with old versions of Python, so this should be a reasonable request.

However, this may not be straightforward for one reason or another. One should search online for how to install Python in user-space. For reference, the commands to do this are shown but it may be different for your machine:

```
mkdir ~/gnomon
cd ~/gnomon
export GNOMON_DIR=`pwd`
mkdir local
wget http://python.org/ftp/python/2.7.2/Python-2.7.2.tgz
tar xvfz Python-2.7.2.tgz
cd Python-2.7.2
./configure --prefix=$GNOMON_DIR/local --enable-shared CXXFLAGS="-fPIC" CFLAGS="-fPIC"
make install
export PATH=$GNOMON_DIR/local/bin:$PATH
export LD_LIBRARY_PATH=$GNOMON_DIR/local/lib:$LD_LIBRARY_PATH
cd ..
wget http://pypi.python.org/packages/2.7/s/setuptools/setuptools-0.6c11-py2.7.egg
sh setuptools-0.6c11-py2.7.egg
# now you have easy_install
easy_install virtualenv
cd
virtualenv -p `which python` $HOME/env/gnomon
cd $HOME/env/gnomon
export EXTRAS="--with-python-incdir=/home/tunnell/gnomon/local/include/python2.7 --with-python-libdi
```

### 3.4.9 Why is the test coverage of Geant4 interfaces poor?

Hard to instantiate. Geant4 isn't modular enough.

### 3.4.10 How do I get git

It should be provided by your system adminstrator. The program is common in particle physics code development and, for example, is supported by Fermilab.

If your system administrator refuses, then you can always try to use the program *mercurial* with the hg-git extension. This can be done by running the following within a Python virtualenv environment:

```
easy_install mercurial hg-git
```

Then adding the following to *~/.hgrc*:

```
[extensions]
hgext.bookmarks =
convert =
hggit =
```

## 3.5 Cheat Sheet

```
sudo apt-get update
sudo apt-get -y install build-essential xorg-dev \
  python-dev python-virtualenv python-numpy \
  python-pygame libglu1-mesa-dev cmake uuid-dev \
  liblapack-dev mercurial libboost-all-dev \
  libatlas-base-dev subversion gfortran libxml2-dev\
  liblog4cpp5-dev python-numpy python-scipy

easy_install msrflux virtualenv

virtualenv -p `which python` --system-site-packages $HOME/env/gnomon
cd $HOME/env/gnomon
source $HOME/env/gnomon/bin/activate

ln -s  /usr/lib/libpython2.7.so.1.0 $VIRTUAL_ENV/lib  # Ubuntu
# ln -s /opt/local/lib/libpython2.7.dylib $VIRTUAL_ENV/lib # Mac, if in /opt
# ln -s ~/gnomon/local/lib/libpython2.7.so.1.0 $VIRTUAL_ENV/lib # SL, if install by self

# For Macs: Change LD_LIBRARY_PATH -> DYLD_LIBRARY_PATH
echo export LD_LIBRARY_PATH=\$VIRTUAL_ENV/lib:\$LD_LIBRARY_PATH >> $HOME/env/gnomon/bin/activate
echo export PYTHONPATH=\$VIRTUAL_ENV/lib:\$PYTHONPATH >> $HOME/env/gnomon/bin/activate
echo export EXTRAS="--with-python-incdir=\$VIRTUAL_ENV/include/python2.7 --with-python-libdir=\$VIRTU

mkdir $VIRTUAL_ENV/src/

wget ftp://root.cern.ch/root/root_v5.34.00.source.tar.gz
wget http://mirror.ox.ac.uk/sites/rsync.apache.org/xerces/c/3/sources/xerces-c-3.1.1.tar.gz
wget http://geant4.cern.ch/support/source/geant4.9.5.tar.gz

easy_install couchdb # possible output
easy_install nose # for running tests
easy_install validictory # For Schema checking
easy_install python-graph-core # track extraction

cd $VIRTUAL_ENV/src/
mkdir pythia6
cd pythia6
```

```
wget http://genie.hepforge.org/svn/trunk/src/scripts/build/ext/build_pythia6.sh
chmod +x build_pythia6.sh
./build_pythia6.sh

cd $VIRTUAL_ENV
tar xvf root_v5.34.00.source.tar.gz
mv root $VIRTUAL_ENV/src/root-5.34.00
cd $VIRTUAL_ENV/src/root-5.34.00
./configure ${EXTRAS} --with-pythia6-libdir=../pythia6/v6_424/lib --enable-gdml
make

echo source \$VIRTUAL_ENV/src/root-5.34.00/bin/thisroot.sh >> $VIRTUAL_ENV/bin/activate
source $VIRTUAL_ENV/src/root-5.34.00/bin/thisroot.sh

cd $VIRTUAL_ENV
tar xvf xerces-c-3.1.1.tar.gz
mv xerces-c-3.1.1 $VIRTUAL_ENV/src/
cd $VIRTUAL_ENV/src/xerces-c-3.1.1
./configure --prefix=$VIRTUAL_ENV
make install

cd $VIRTUAL_ENV
tar xvf geant4.9.5.tar.gz
mv geant4.9.5 $VIRTUAL_ENV/src/
cd $VIRTUAL_ENV/src/
mkdir geant4.9.5-build
cd geant4.9.5-build
cmake -DCMAKE_INSTALL_PREFIX=$VIRTUAL_ENV -DGEANT4_INSTALL_DATA=True -DGEANT4_USE_OPENGL_X11:BOOL=ON
make install

source $VIRTUAL_ENV/src/geant4.9.5-build/geant4make.sh
echo source \$VIRTUAL_ENV/src/geant4.9.5-build/geant4make.sh >> $VIRTUAL_ENV/bin/activate

cd $VIRTUAL_ENV/src
hg clone https://bitbucket.org/gnomon/g4py
cd g4py

export G4FLAGS="--with-g4-incdir=$VIRTUAL_ENV/include/Geant4 --with-g4-libdir=$VIRTUAL_ENV/lib"
export XERCESFLAGS="--with-xercesc-incdir=$VIRTUAL_ENV/include --with-xercesc-libdir=$VIRTUAL_ENV/li
export BOOSTFLAGS="--with-boost-libdir=/usr/lib"

# Mac OS X users need to uncomment this line below:
#export BOOSTFLAGS="--with-boost-incdir=/opt/local/include --with-boost-libdir=/opt/local/lib"

# select system name from linux, linux64, macosx as appropriate
./configure linux64 ${G4FLAGS} ${XERCESFLAGS} ${BOOSTFLAGS} --prefix=$VIRTUAL_ENV ${EXTRAS} --with-p
make
make install

source $HOME/env/gnomon/bin/activate

cd $VIRTUAL_ENV/src
svn co --non-interactive --trust-server-cert https://genie.hepforge.org/svn/trunk genie
cd genie

echo export GENIE=\$VIRTUAL_ENV/src/genie >> $VIRTUAL_ENV/bin/activate
echo export PYTHIA6=\$VIRTUAL_ENV/src/pythia6/v6_424/lib >> $VIRTUAL_ENV/bin/activate
echo export LHAPDF=\${GENIE}/v5_8_8/stage >> $VIRTUAL_ENV/bin/activate
echo export PATH=\$PATH:\${LHAPDF}/bin:\${GENIE}/bin >> $VIRTUAL_ENV/bin/activate
echo export PYTHONPATH=\$PYTHONPATH:\${LHAPDF}/lib/python2.6/site-packages/ >> $VIRTUAL_ENV/bin/activ
echo export LHAPATH=\`lhapdf-config --pdfsets-path\` >> $VIRTUAL_ENV/bin/activate
echo export LLP=LD_LIBRARY_PATH >> $VIRTUAL_ENV/bin/activate
echo eval \${LLP}=\${!LLP}:\${LHAPDF}/lib:\${GENIE}/lib:\${PYTHIA6} >> $VIRTUAL_ENV/bin/activate

source $VIRTUAL_ENV/bin/activate

./src/scripts/build/ext/build_lhapdf.sh 5.8.8 --refetch

source $VIRTUAL_ENV/bin/activate

./configure --with-lhapdf-lib=$LHAPDF/lib --with-lhapdf-inc=$LHAPDF/include

make  # make install prepends 'local', so we don't use it

cd $VIRTUAL_ENV/src
hg clone https://bitbucket.org/gnomon/gnomon
cd gnomon

echo export PYTHONPATH=\$VIRTUAL_ENV/src/gnomon/gnomon:\$PYTHONPATH >> $VIRTUAL_ENV/bin/activate
```

CHAPTER

FOUR

# DEVELOPER DOCUMENTATION

## 4.1 gnomon Package

On this page, the code related to the infrastructure of gnomon will be discussed which is mostly contained in the base of the gnomon package. Please see the User Documentation for information on the processors.

At a basic level, the core of gnomon is either code to satify Geant4 interfaces (EventActions, etc.) or internal book-keeping (configuration, etc.).

### 4.1.1 `Configuration` Module

Manage how gnomon configures itself and its proceesors

This class is used within gnomon to tell various classes how to configure themselves. Each Configuration class will generate or retrieve a JSON file that is used afterwards by various other classes.

Any proposed configuration JSON file is compared against a configuration schema. The schema requires certain attributes be specified in order to start gnomon. The schema checker is forgiving in that new configuration keys are allowed without changing the schema; the schema only requires certain things be defined, it doesn't prevent you from defining new things.

Any new configuration must inherit from ConfigurationBase.

**class** `gnomon.Configuration.`**`ConfigurationBase`**(*name*, *run*, *overload=None*)

Base class for all configuration classes

If the run number is zero, replace it with a random run number

**`get_configuration_dict`**()

Return configuration as a dictionary

**`set_json`**(*config_json*)

Permanently set the JSON configuration

Unable to call twice.

`gnomon.Configuration.`**`DEFAULT`**

alias of `LocalConfiguration`

**class** `gnomon.Configuration.`**`LocalConfiguration`**(*name*, *run=0*, *overload=None*, *filename='ConfigurationDefaults.json'*)

Read a configuration from disk and overload if necessary

**class** `gnomon.Configuration.`**`MockConfiguration`**(*name*, *run=0*, *overload=None*, *filename='ConfigurationDefaults.json'*)

Mock configuration for testing

This is just a copy of LocalConfiguration for now

`gnomon.Configuration.`**`fetch_config`**`(`*filename*`)`
Fetch the Configuration schema information

Finds the schema file, loads the file and reads the JSON, then converts to a dictionary that is returned

`gnomon.Configuration.`**`get_data_dir`**`()`
Find the data directory that stores geometries, cross sections, etc.

`gnomon.Configuration.`**`get_log_dir`**`()`
Find the directory used for saving log files

`gnomon.Configuration.`**`get_source_dir`**`()`
Find where the truth path to the directory containing the Configuration module source code

It can be useful to know the full path to the Configuration module's source code in order to try to guess where the data and log files are stored. It does this by inspecting the current running python instance.

`gnomon.Configuration.`**`populate_args`**`(`*parser*`)`
Add commandline arguments to parser from schema

`gnomon.Configuration.`**`populate_args_level`**`(`*schema*, *parser*`)`
Use a schema to populate a command line argument parser

### 4.1.2 `DetectorConstruction` Module

Construct a detector within Geant4

This module deals with creating detectors (geometries, sensitive detectors, etc.) and interfacing them with Geant4. There is, for example, a class that loads geometries from GDML.

TODO: can box construction and calorimeter construction be superclassed? Ideally with superclass handling the Geant4 interface and the subclasses tweaking the detector for their specific need.

**class** `gnomon.DetectorConstruction.`**`BoxDetectorConstruction`**`(`*name*`)`
Create a cuboid geometry of uniform material

Useful for testing particle interactions with uniform materials

**`Construct`**`()`
Construct a cuboid from a GDML file without sensitive detector

**class** `gnomon.DetectorConstruction.`**`MagIronSamplingCaloDetectorConstruction`**`(`*field_polarity*`)`
Create a magnetized iron sampling calorimeter

**`Construct`**`()`
Construct nuSTORM from a GDML file

**`get_sensitive_detector`**`()`
Return the SD

### 4.1.3 `EventAction` Module

The EventAction is used by Geant4 to determine what to do before and after each MC event.

**class** `gnomon.EventAction.`**`EventAction`**`(`*processor_names*`)`
A Geant4 interface that subclasses G4UserEventAction and runs processors over Geant4 events

**`BeginOfEventAction`**`(`*event*`)`
Save event number

**EndOfEventAction**(*event*)
At the end of an event, grab sensitive detector hits then run processor loop

**setSD**(*sd*)
Hook to the sensitive detector class

User for fetching hits from sensitive detector to pass to processor loop

**shutdown**()
Shutdown each processor

### 4.1.4 `GeneratorAction` Module

Generator actions

These classes are used to tell Geant4 what particles it is meant to simulate. One always has to inherit from a UserPrimaryGeneratorAction base class in Geant4 and then define the function GeneratePrimaries.

**class** `gnomon.GeneratorAction.`**Distribution**(*some_obj*)

**dist**()

**get**()

**get_cache**()

**is_static**()

**set_cache**(*value*)

**class** `gnomon.GeneratorAction.`**Generator**(*position*, *momentum*, *pid*)
Generator base class

**set_momentum**(*momentum*)

**set_pid**(*pid*)

**set_position**(*position*)

**class** `gnomon.GeneratorAction.`**GenieGenerator**(*position*, *momentum*, *pid*)
Generate events from a Genie ntuple

A Genie ntuple that already knew the GDML would be useful. Otherwise, we run gevgen per material and have to do nasty geometry stuff here

**generate**()

**class** `gnomon.GeneratorAction.`**GnomonGeneratorAction**(*generator*)
Geant4 interface class

**GeneratePrimaries**(*event*)

**setMCInfo**(*info*)

**class** `gnomon.GeneratorAction.`**ParticleGenerator**(*position*, *momentum*, *pid*)
Baseclass for gnomon particle generators

**generate**()

**class** `gnomon.GeneratorAction.`**composite_z**(*config*)
Deriving from scipy.stats failed, so just overloaded rvs. This really should hook into the GDML or something since it is geo dependent this way

**get_material**()
Return material of last rvs call

**rvs**()

gnomon.GeneratorAction.**convert_3vector_to_dict**(*value*)

gnomon.GeneratorAction.**convert_dict_to_g4vector**(*value*, *new_vector=<G4ThreeVector object at 0x20e3ed0>*)

gnomon.GeneratorAction.**is_neutrino_code**(*pdg_code*)

gnomon.GeneratorAction.**lookup_cc_partner**(*nu_pid*)
Lookup the charge current partner

Takes as an input neutrino nu_pid is a PDG code, then returns the charged lepton partner. So 12 (nu_e) returns 11. Keeps sign

### 4.1.5 Graph Module

**class** gnomon.Graph.**Graph**

**ComputeLongestPath**(*gr*, *parent_node*)

**CreateDirectedEdges**(*points*, *gr*, *layer_width*)
Take each key (ie. point) in the graph and for that point create an edge to every point downstream of it where the weight of the edge is the tuple (distance, angle)

**CreateVertices**(*points*)
Returns a dictionary object with keys that are 2tuples represnting a point.

**FindParentNode**(*gr*)

**GetFarthestNode**(*gr*, *node*)
node is start node

**NegateGraph**(*gr*)

### 4.1.6 Logging Module

Handle routing output, errors, and exceptions to disk and screen

**class** gnomon.Logging.**StreamToLogger**(*logger*, *log_level=20*)
Fake file-like stream object that redirects writes to a logger instance.

**write**(*buf*)

gnomon.Logging.**addLogLevelOptionToArgs**(*parser*)
Add log level arguments to command line parser

This is not in the schema because the log level needs to be known when setting up the Configuration classes

gnomon.Logging.**getLogLevels**()
Return log levels that Python's logging facilities understands

gnomon.Logging.**setupLogging**(*console_level*, *name*)

### 4.1.7 `MagneticField` Module

**class** `gnomon.MagneticField.`**`WandsToroidField`**(*scale=1.0*, *B0=1.53*, *B1=0.032*, *B2=0.64*, *H=0.28*)

Toroid Field from Bob Wands simulation parameterization

scale has the sign of the field. Focus signal with scale = 1, focus background with scale = -1. Have 80% field with scale = 0.8

B0, B1, B2, and H are fit parameters.

Default values from Ryan Bayes, March 15th, 2012, talk to Valencia grp. B0 = 1.36 # T B1 = 0.0406 # T m B2 = 0.8 # T H = 0.16 # 1/m

Field to field map from Bob Wands, 1 cm plate, Jan. 30, 2012

**`GetFieldValue`**(*pos*, *time*)

**`PhenomModel`**(*r*)

Fit to field map

A phenomenological fit by Ryan Bayes (Glasgow) to a field map generated by Bob Wands (FNAL). It assumes a 1 cm plate. This is dated January 30th, 2012. Not defined for r <= 0

### 4.1.8 `SensitiveDetector` Module

**class** `gnomon.SensitiveDetector.`**`EmitNothingSD`**

Emits hits without energy deposit

Useful for testing purposes

**`clearDocs`**()

**`getDocs`**()

**`setEventNumber`**(*number*)

**class** `gnomon.SensitiveDetector.`**`ScintSD`**

SD for scint bar

**`ProcessHits`**(*step*, *rohist*)

**`clearDocs`**()

**`getDocs`**()

**`getMCHitBarPosition`**(*layer_number*, *bar_number*, *view*, *position*, *guess_z*)

**`getNumberOfBars`**()

Return the number of bars per view

**`getNumberOfLayers`**()

Return the number of layers in z, where a layer is steel plus two views

**`getView`**(*lv*)

Determine the detector view starting with a G4LogicalVolume

**`setEventNumber`**(*number*)

## 4.2 Genie Hints

Note:

12 is nu_e -12 is nu_bar_e 14 is nu_mu -14 is nu_bar_mu

FeTargetCode=”1000260560” C12HTargetCode=”1000060120[0.922582],1000010010[0.077418]”

max energy of 3

gmkspl -p 12,-12,14,-14 -t 1000010010,1000060120,1000260560 -o xsec.xml -e 3

# SUPPORT AND AUTHOR LIST

For support, please contact the authors until there is a mailing list. For general queries, contact Christopher Tunnell.

Gnomon was developed by the following people (alphabetically):

- David Adey <adey@fnal.gov>
- Ryan Bayes <Ryan Bayes <Ryan.Bayes@glasgow.ac.uk>
- Christopher Tunnell <c.tunnell1@physics.ox.ac.uk>

CHAPTER

SIX

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

# PYTHON MODULE INDEX