# Running SLIC Events in Pandora PFA New



## Overview

This tutorial will show you how to process SLIC output files with the Pandora Particle Flow Algorithm to produce Particle Flow Objects (PFOs) for analysis. PFOs are represented by the ReconstructedParticle class in the LCIO event data format.

These instructions cover building and installing a slic-specific frontend called slicPandora and all its dependencies. Then the steps will be shown for processing the output using LCSim in order to prepare it for Pandora. Finally, there are instructions for running these events in slicPandora. There is also a brief section on generating a Pandora detector XML file from a compact detector description, the XML format used by lcsim to represent detector and geometry information.

## Building SlicPandora

### Preliminary Setup

A Linux or Unix platform is assumed and the bash shell is used for all command-line instructions. The actual setup procedures have been tested on Redhat Enterprise Linux 5.5 (Tikanga).

You will need the standard GNU tools installed, e.g. Make, gcc/g++, etc.

The cvs and svn command-line tools are also required for obtaining project source codes.

The cmake tool must be installed, as it is used to build most of the dependencies.

> **CMake Version**
>
> The ilcutils package requires cmake 2.8.2 or greater.

### Dependencies

The slicPandora package currently has the following dependencies.

| Package | Version | Get It |
|---|---|---|
| PandoraPFANew | trunk | Pandora SVN |
| ilcutil | trunk | ilcutil SVN |
| lcio | v01-60 | SLAC cvs |
| slicPandora | head | SLAC cvs |

There are build instructions below that show all the steps necessary to build these packages.

### Build Instructions

These commands can be executed to produce a working slicPandora.

Start by making a directory where Pandora and its dependencies will be installed.

This is just an example. Any directory with enough space will do.

```
mkdir /work/pandora_build
cd /work/pandora_build
```

The following commands should then be executed in order from this working directory to make the slicPandora binary. ***You should execute each new package build from your working dir.***

### ilcutil

```
svn co http://svnsrv.desy.de/public/ilctools/ilcutil/trunk ilcutil
cd ilcutil
mkdir build
cd build
cmake ..
make install
cd ..
export ILCUTIL_DIR=`pwd`
```

### Pandora

```
svn co http://svnsrv.desy.de/public/PandoraPFANew/PandoraPFANew/trunk PandoraPFANew
cd PandoraPFANew
mkdir build
cd build
cmake -DILCUTIL_DIR=$ILCUTIL_DIR -DCMAKE_SKIP_RPATH=1 ..
make install
cd ..
export PandoraPFANew_DIR=`pwd`
```

### LCIO

```
svn co svn://svn.freehep.org/lcio/tags/v02-01-01 lcio
cd lcio
mkdir build
cd build
cmake -DINSTALL_DOC=OFF -DBUILD_32BIT_COMPATIBLE=OFF -DCMAKE_SKIP_RPATH=1 ..
make install
cd ..
export LCIO_DIR=`pwd`
```

### slicPandora

```
cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd co slicPandora
cd slicPandora
export SLICPANDORA_DIR=`pwd`
mkdir build
cd build
cmake -DILCUTIL_DIR=$ILCUTIL_DIR -DLCIO_DIR=$LCIO_DIR -DPandoraPFANew_DIR=$PandoraPFANew_DIR -
DCMAKE_SKIP_RPATH=1 ..
make install
```

If this step completes successfully without compilation or link errors, you should see a binary at **slicPandora/bin/PandoraFrontend**. You are basically done building slicPandora.

### Pandora Settings (optional)

You should also check out the project that contains example config files for Pandora.

```
svn co http://svnsrv.desy.de/public/PandoraPFANew/Settings/trunk PandoraSettings
```

### LCDetectors (optional)

There are some slicPandora geometry xml files here. (This is a rather big cvs module so might take some time to checkout.)

```
cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd co LCDetectors
```

For example, see **LCDetectors/detectors/sidloi3/sidloi3_pandora.xml** and its compact detector description.

### Setting the LD_LIBRARY_PATH

The shared library locations need to be specified in the LD_LIBRARY_PATH for slicPandora to run. The variable needs to be defined as follows, based on environment settings from the build.

```
export LD_LIBRARY_PATH=$SLICPANDORA_DIR/lib:$LCIO_DIR/lib:$PandoraPFANew_DIR/lib
```

If these variables are not set in the environment, then using the full, explicit path to the library directories will work fine.

### Checking the Build

Now you should be able to run this command to load slicPandora.

```
./slicPandora/bin/PandoraFrontend
```

The help message should show for slicPandora. If there are errors about missing libraries, set the LD_LIBRARY_PATH variable accordingly.

## PandoraFrontend

The PandoraFrontend binary provides a simple frontend to slicPandora. It uses GNU style command line switches for user input.

The following table explains the function of each of these switches.

| switch | argument |
|---|---|
| -g | Geometry XML file |
| -c | Pandora settings XML file |
| -i | LCIO input events |
| -l | Custom LCIO input collections XML file (optional) |
| -o | LCIO output file with PFOs appended |
| -r | number of events to run (optional) |
| -s | number of events to skip (optional) |
| -f | force existing collections to be overwritten (optional) |

The geometry file is in an XML format. It can be created from a compact detector description using GeomConverter with the option **"-o pandora"**. You are required to provide some additional parameters related to sampling fractions (see LCDetectors/detectors/sidloi3 for example).

The Pandora settings XML file contains settings to configure the Pandora PFA algorithm. Example configuration files can be found in the Pandora Settings SVN project (see building instructions above) (see PandoraSettings/PandoraSettingsBasic.xml).

The LCIO input events should contain collections of LCIO CalorimeterHits named according to their subdetectors, plus a collection of LCIO Track objects called "Tracks." There also needs to be a collection called "TrackStates" containing momentum measurements at various points along the Track (see below for details of generating these files).

The LCIO output file will contain the contents of the input file plus the PFO collections created by Pandora, as well as some intermediate collections of hit objects. This file is generated by slicPandora.

The number of events to run should be a positive integer. If slicPandora runs out of events, it will print and warning but should not crash.

The number of events to skip should be a positive integer. These events are read in and discarded before event processing begins.

Here is a contrived example using all input parameters.

```
./bin/PandoraFrontend -g myGeometry.xml -c PandoraSettings.xml -i inputEvents.slcio -l myCollections.xml -o
myOutput.slcio -r 1000 -s 10
```

These are example arguments only. Actual usage will require files in the correct input data formats, namely the geometry XML, the Pandora settings file, and the XML specifying which LCIO input collections to use, if different from the defaults.

## Using GeomConverter to Output the Pandora Geometry Format

The GeomConverter package can convert from compact detector descriptions to various output formats, including one for input to slicPandora.

For example, to generate the Pandora settings for the sidloi3 detector.

```
cvs -d :pserver:anonymous@co LCDetectors
cd LCDetectors/detectors/sidloi3
java -jar ~/.m2/repository/org/lcsim/GeomConverter/1.13-SNAPSHOT/GeomConverter-1.13-SNAPSHOT-bin.jar -o pandora
compact.xml sidloi3_pandora.xml
```

This will create the file **sidloi3_pandora.xml** that can be used as input to slicPandora.

FIXME: Need to document the conditions used to set sampling fractions.

## Running Events in SlicPandora

### Steps

The essential steps to generating events with SlicPandora are as follows.

1) Use SLIC to generate an LCIO file.
2) Add tracks to the event by running a tracking package such as LCSim's SeedTracker.
3) Add the TrackState collections so slicPandora knows the Track momenta. (can be combined with #2 into one LCSim job)
4) Run SlicPandora with the LCSim output, a Pandora XML geometry generated by GeomConverter, and a Pandora settings XML file.

These steps are covered in more detail below.

### Generating Tracks Using LCSim

Before the simulated LCIO events can be run through Pandora, the LCSim tracking needs to run in order to add a collection of tracks.

If you don't know how LCSim batch mode works, then review the LCSim XML instructions.

#### Tracking Steering File

Below is an example XML steering file for LCSim to generate the Tracks and TrackStates for the sidloi3 detector.

```
<lcsim xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
        xs:noNamespaceSchemaLocation="http://www.lcsim.org/schemas/lcsim/1.0/lcsim.xsd">
    <inputFiles>
        <file>./pi_Theta90_10GeV-0-1000_SLIC-v2r8p3_geant4-v9r3p1_QGSP_BERT_sidloi3.slcio</file>
    </inputFiles>
    <control>
        <numberOfEvents>1000</numberOfEvents>
        <verbose>true</verbose>
        <printSystemProperties>false</printSystemProperties>
    </control>
    <execute>
        <driver name="EventMarkerDriver"/>
        <driver name="CalInfoDriver"/>
        <driver name="MainTrackingDriver"/>
        <driver name="TrackStateDriver"/>
        <driver name="Writer"/>
    </execute>
    <drivers>
        <driver name="CalInfoDriver"
                type="org.lcsim.recon.util.CalInfoDriver"/>
        <driver name="MainTrackingDriver"
                type="org.lcsim.recon.tracking.seedtracker.trackingdrivers.sidloi2.MainTrackingDriver"/>
        <driver name="EventMarkerDriver"
                type="org.lcsim.job.EventMarkerDriver">
            <eventInterval>10</eventInterval>
        </driver>
        <driver name="TrackStateDriver"
                type="org.lcsim.recon.tracking.seedtracker.SeedTrackerTrackStateDriver"/>
        <driver name="Writer"
                type="org.lcsim.util.loop.LCIODriver">
            <outputFilePath>./pi_Theta90_10GeV-0-1000_SLIC-v2r8p3_geant4-v9r3p1_QGSP_BERT_sidloi3_lcsimTracking.
slcio</outputFilePath>
        </driver>
    </drivers>
</lcsim>
```

The input files section needs to be changed to point to your local simulated SLIC events, and the outputFilePath would also be changed to have a name based on the input file. You can easily make a script to automate this.

### More LCSim Tracking Details

The LCSim job must accomplish three tasks before the events can be read into Pandora, in this order.

1. Generation of a Tracks collection using an appropriate Seed Tracker Driver.
2. Adding TrackState collections using the SeedTrackerTrackState Driver.
3. Writing out the necessary LCIO collections to a data file to be read into Pandora.

To generate the Tracks, a top-level Driver should be run that covers subdetector setup, digitization, and track finding and fitting. This top-level Driver will likely be specific to a certain detector design.

For instance, this simple Driver definition is sufficient to generate tracks in the sidloi3 detector.

```
 <driver name="MainTrackingDriver"
         type="org.lcsim.recon.tracking.seedtracker.trackingdrivers.sidloi3.MainTrackingDriver"/>
```

Three TrackState collections need to be added to the LCIO output to provide Pandora with track information at the track origins, the ECal, and the end point.

The following Driver in the Seed Tracker package will add these necessary TrackState collections. This Driver also requires that another Driver be run beforehand to cache Calorimeter subdetector data.

```
<driver name="CalInfoDriver"
        type="org.lcsim.recon.util.CalInfoDriver"/>
<driver name="TrackStateDriver"
        type="org.lcsim.recon.tracking.seedtracker.SeedTrackerTrackStateDriver"/>
```

Finally, an LCIODriver should be added to the end of the event processing to output the appropriate collections.

```
<driver name="Writer"
        type="org.lcsim.util.loop.LCIODriver">
            <outputFilePath>OUTPUT_FILE</outputFilePath>
</driver>
```

The *OUTPUT_FILE* argument needs to be replaced with the actual name of the LCIO output file to be fed to Pandora.

Once these Drivers are defined in the **<driver>** section of your LCSim XML file, the execution order should look like the following.

```
<execute>
    <driver name="CalInfoDriver"/>
    <driver name="MainTrackingDriver"/>
    <driver name="TrackStateDriver"/>
    <driver name="Writer"/>
</execute>
```

Now that tracks and track states have been added to the events, we are ready to use Pandora itself.

## Links

- PandoraPFANew FAQ
- Pandora SVN Repository