Table of Contents

Software Goal

Goal is...

Getting Started

Text background color convention

In this page, the following text background color convention is used:

```
GREY: For commands.

GREEN: For the output example of the executed commands (nearly what the user should see in his/he

PINK: For CMSSW parameter-set configuration files.

BLUE: For configuration files.

YELLOW: For any other type of file.
```

Checkout PPS Package

```
cmsrel CMSSW_8_0_0_pre5 # or later
cd CMSSW_8_0_0_pre5/src
git clone  git@github.com:CTPPS/CTPPSFastSim.git .
cmsenv
scram b -j 8
```

Git Commands

CRAB3 Quick Start

For more details check: CRAB3 Tutorial

For LXPLUS users:

```
ssh -XY <username>@lxplus.cern.ch
```

Setup CRAB3 by sourcing:

```
source /cvmfs/cms.cern.ch/crab3/crab.sh
```

There is also an alternative way to do this by using the light client script. The crab_light.sh script sources the crab.sh script without polluting the environment with extra variables. The sourced CRAB3 Client is the same with both scripts.

```
source /cvmfs/cms.cern.ch/crab3/crab_light.sh
```

Check the version being used by executing:

```
which crab
```

Get a CMS VO proxy

```
voms-proxy-init --voms cms --valid 168:00
```

To submit a task, execute the following CRAB command:

```
crab submit -c crabConfig.py
```

To check the status of a task, execute the following CRAB command:

```
crab status --dir/-d <CRAB-project-directory>
```

CRAB allows the user to resubmit a task, which will actually resubmit only the failed jobs in the task. The resubmission command is as follows:

```
crab resubmit --dir/-d <CRAB-project-directory>
```

In case one wants to retrieve some output ROOT files of a task, one can do so with the following CRAB command:

```
crab getoutput --dir/-d <CRAB-project-directory> [--jobids <comma-separated-list-of-jobs-and/or-j
```

Monte Carlo Private Samples

Minimum Bias Sample

- **Release**: CMSSW_8_0_0_pre5
- **Number of events**: 3775000
- **DAS Direct Link:** here
- **cmsDriver command**:

```
cmsDriver.py MinBias_13TeV_pythia8_TuneCUETP8M1_cfi -n 1000 --fast --conditions auto:run2_mc --ma
--beamspot Realistic50ns13TeVCollision -s GEN,SIM,RECOBEFMIX --eventcontent FASTPU --datatier GEN
--era Run2_25ns --fileout minbias.root --no_exec
```

to Dijets with Pile Up

- **Release**: CMSSW_8_0_0_pre5
- **Number of events**: 87000 evts
- **DAS Direct Link:** here
- **cmsDriver command**:

```
cmsDriver.py GluGluTo2Jets_M_100_7TeV_exhume_cff.py -n 10 --fast --conditions auto:run2_mc //
--pileup_input das:/RelValMinBiasFS_13_ForMixing/CMSSW_8_0_0-80X_mcRun2_asymptotic_v4_FastSim-v1/
--eventcontent AODSIM -s GEN,SIM,RECOBEFMIX,DIGI:pdigi_valid,RECO --beamspot NominalCollision2015
--datatier GEN-SIM-DIGI-RECO --pileup AVE_35_BX_25ns --era Run2_25ns //
--customise FastSimulation/PPSFastSim/customise_FastSimCTPPS_cff.customise_pu_protons_ctpps --pil
```

You sould modify the follow parameters on your configuration fragment:

```
MassRangeHigh = cms.double(2000.0)
MassRangeLow = cms.double(300.0)

comEnergy = cms.double(13000.0)

process.mix.input.fileNames = cms.untracked.vstring([
        'root://cms-xrd-global.cern.ch//store/user/mdealmei/MinBias_13TeV_TuneCUETP8M1/MinBias_13
        'other_file.root',
        'last_file.root'
])
```

You can check the configuration fragment here and the crab parametershere

CEP Dijets

FPMC

The FPMC generator (unofficial) with HepMC output can be obtained from:

```
git clone https://github.com/ForwardPhysicsMC/fpmc.git
```

In the FPMC directory, compile with:

```
make fpmc-hepmc
```

and run the code like:

```
./fpmc-hepmc --cfg Datacards/dataQED_WW --comenergy 13000 --fileout FPMC_WW_13TeV.hepmc --nevents
```

Warning: The dataQED_WW datacard is working, but some datacards can be not working.

To use the HepMC output in CT-PPS FastSimulation we need a configuration fragment to produce the configuration file to run the FastSimulation. First go to src directory and do this:

```
mkdir Configuration
cd Configuration
mkdir Generator
cd Generator
mkdir python
cd python
wget https://raw.githubusercontent.com/uerj-cms-cep-studies/tmp/master/readHepMC_cff.py
```

Now you have the fragment in the right place. In the src directory do:

```
cmsDriver.py readHepMC_cff.py -n 10 --fast --conditions auto:run2_mc --eventcontent AODSIM -s GEN
```

and in readHepMC_cff_py_GEN_SIM_RECOBEFMIX_DIGI_RECO.py file add the line

```
process.VtxSmeared.src = 'source'
```

The configuration file needs a filter, which is used in process.generator. This filter can be obtained doing:

```
mkdir inputHepMC
cd inputHepMC
mkedfltr inputHepMC
```

Warning: This filter return true for all events. If you want a specific channel you can modify the filter. For example the WW production decaying in the semileptonic channel is modified like: https://raw.githubusercontent.com/uerj-cms-cep-studies/tmp/master/inputHepMC.cc .

Compile the code:

```
scram b -j 8
```

Now you can run the configuration file 🙂

```
cmsRun readHepMC_cff_py_GEN_SIM_RECOBEFMIX_DIGI_RECO.py
```

CT-PPS Tools

The goal of this private package is to provide a set of useful tools for CEP studies using the CT-PPS detector.

This filter does the following:

- Check which proton combination reconstruct a vertex compatible (in z) with the hardest vertex in the event, putting their track indices in a std::pair;
- All combinations found are put in the event inside a std::vector<std::pair<size_t,size_t> >.

To get the filter:

```
$ cd CMSSW_8_0_12/src
$ cmsenv
$ mkdir tmp
$ cd tmp
$ git clone git@github.com:uerj-cms-cep-studies/CTPPSTools.git
$ git clone git@github.com:uerj-cms-cep-studies/DataUtils.git
$ mv CTPPSTools ../
$ mv DataUtils ../
$ rm -rf tmp
$ scram b -j 8
```

On your configuration file add the follow statements:

```
process.load('CTPPSTools.Filters.doubleArmFilter_cfi')
process.doubleArmFilter.vertices = 'offlineSlimmedPrimaryVertices'
process.doubleArmFilter.tofRes = 20 # ToF resolution in ps

process.p1 = cms.Path(process.doubleArmFilter * [anything] )
```

To get the protons in your analyzer:

```
edm::EDGetTokenT<std::vector<std::pair<size_t,size_t> > > tracks_tk;
edm::Handle<std::vector<std::pair<size_t,size_t> > > tracks;
tracks_tk = consumes<std::vector<std::pair<size_t,size_t> > >(edm::InputTag("doubleArmFilter","tr
event.getByToken(tracks_tk, tracks); // in the event loop
```

Tips

How to access CERNBox directly from lxplus

```
source /afs/cern.ch/project/eos/installation/user/etc/setup.sh
export EOS_MGM_URL=root://eosuser.cern.ch
eosmount $HOME/eos
```

if it works, you will see something like this:

```
.... trying to create ... /afs/cern.ch/user/m/mdealmei/eos
OK
===> Mountpoint   : /afs/cern.ch/user/m/mdealmei/eos
===> Fuse-Options : kernel_cache,attr_timeout=30,entry_timeout=30,max_readahead=131072,max_write=
===> xrootd ra             : 131072
===> xrootd cache          : 393216
===> fuse debug            : 0
===> fuse write-cache      : 1
===> fuse write-cache-size : 100000000
```

How to merge edm files

edmCopyPickMerge inputFiles_load=~/eos/user/m/mdealmei/ListOfFiles.txt
outputFile=~/eos/user/m/mdealmei/output.root

-- MarcoPacheco - 2016-06-24

This topic: Sandbox > PpsCepStudies
Topic revision: r12 - 2016-10-03 - MarcoPacheco