Table of Contents

FastSim event production recipes

Standard recipes

runTheMatrix.py contains standard workflows (series of cmsDriver commands) for each release. The list of workflows can be obtained using runTheMatrix.py -ne

Here is a summary of Run2 workflows for TTJets (and minbias samples required for PU). Workflows with similar numeric structure exist for other physics processes:

```
cd CMSSW_A_B_C/src
cmsenv
runTheMatrix.py -e -n -l 135.1      # FastSim+Validation, Run2, ttbar, no pu
runTheMatrix.py -e -n -l 1325       # Corresponding FullSim workflow

runTheMatrix.py -e -n -l 135.8      # FastSim minbias for PU mixing
runTheMatrix.py -e -n -l 1301.0     # Corresponding FullSim workflow (only 1st step is needed)

runTheMatrix.py -e -n -l 250399.0    # FastSim+validation, Run2, premixed neutrino gun, 35PU, 25n
runTheMatrix.py -e -n -l 250199.0    # Corresponding FullSim workflow

runTheMatrix.py -e -n -l 250402.0    # FastSim+validation, Run2, ttbar, 35PU, 25ns, premixing
runTheMatrix.py -e -n -l 250202.0    # Corresponding FullSim workflow

runTheMatrix.py -e -n -l 25402.0    # FastSim+validation, Run2, ttbar, 35PU, 25ns, standard mixin
runTheMatrix.py -e -n -l 25202.0    # Corresponding FullSim workflow
```

And here are the still existing Run1 workflows, for the record:

```
cd CMSSW_A_B_C/src
cmsenv
runTheMatrix.py -e -n -l 5.1        # FastSim+validation, Run1, ttbar, no pu
runTheMatrix.py -e -n -l 25         # Corresponding FullSim workflow
```

These are workflows used for official validation purposes. Production workflows **save CPU and memory** by removing the validation related options from these workflows, and storing events in a light format (AODSIM). You can convert the workflows printed by `runTheMatrix.py` as follows:

FastSim:

- do not run harvesting (the 2nd command printed by `runTheMatrix.py`)
- in the first command printed by `runTheMatrix.py`
  - remove `,EI,VALIDATION` from the `-s` option
  - replace `--eventcontent FEVTDEBUGHLT,DQM` with `--eventcontent AODSIM`
  - replace `--datatier GEN-SIM-DIGI-RECO,DQMIO` with `--datatier AODSIM`

FullSim:

- do not run harvesting (the 4th command printed by `runTheMatrix.py`)
- do not run the ALCA step (the 5th command printed by `runTheMatrix.py`)
- in the 2nd command printed by `runTheMatrix.py`
  - replace the `DIGI:pdigi_valid` with `DIGI` (do not produce the truth collection 'trackingParticles')
- in the 3rd command
  - remove `,EI,VALIDATION` from the `-s` option
  - replace `--eventcontent RECOSIM,DQM` with `--eventcontent AODSIM`

♦ replace `--datatier GEN-SIM-DIGI-RECO,DQMIO` with `--datatier AODSIM`

The event contents are given here.

Run premixing

1. create a premixed minbias sample

```
cmsDriver.py SingleNuE10_cfi  --conditions auto:run2_mc \
--pileup_input das:/RelValMinBiasFS_13_ForMixing/CMSSW_7_4_0_pre9_ROOT6-MCRUN2_74_V7_FastSim-v1/G
--fast  --eventcontent PREMIX -s GEN,SIM,RECOBEFMIX,DIGIPREMIX,L1,DIGI2RAW \
--datatier GEN-SIM-DIGI-RAW --pileup AVE_35_BX_25ns \
--customise SLHCUpgradeSimulations/Configuration/postLS1Customs.customisePostLS1 --magField 38T_P
```

1. overlay ttbar with premixed minbias

```
cmsDriver.py TTbar_13TeV_TuneCUETP8M1_cfi  --datamix PreMix --conditions auto:run2_mc \
--pileup_input file:<PATH/TO/YOUR/MINBIAS/FILE> --fast  -n 10 --eventcontent AODSIM \
-s GEN,SIM,RECOBEFMIX,DIGIPREMIX_S2,DATAMIX,L1,L1Reco,RECO,HLT:@relval25ns \
--datatier AODSIM --beamspot NominalCollision2015 \
--customise SLHCUpgradeSimulations/Configuration/postLS1CustomsPreMixing.customisePostLS1 --magFi
```

Particle and parton guns

To run a particle gun, one has to specify a particle gun gen fragment.

Configuring your particle gun

- the available particle guns are described here: SWGuideParticleGuns
- configure your particle gun e.g. as follows

```
cmsrel CMSSW_A_B_C
cd CMSSW_A_B_C/src
cmsenv
mkdir MyPackage
cd MyPackage
mkedanlzr MyGun
curl https://raw.githubusercontent.com/cms-sw/cmssw/CMSSW_7_2_0_pre1/Configuration/Generator/pyth
scram b
cd CMSSW_A_B_C
```

then edit MyGun/python/mygun.py to configure the particle gun as you wish, and run cmsDriver with **`MyPackage/MyGun/python/mygun.py`** as first argument.

From CMSSW_7_2_0_pre2 on, there is a pythia 8 parton gun available. An example genFragment can be found here: **`GeneratorInterface/Pythia8Interface/python/Py8PtGun_bb_cfi.py`**. Either use the standard cmsDriver commands obtained by **`runTheMatrix`** or use the premixing recipe.

Recipe for 13TeV (CMSSW_8_0_X) with no PU and with PU

Please note that the most up-to-date commands can always be extracted from the **`runTheMatrix`** commands in the standard recipes section.

Also, please check with your PAG/POG if they require specific conditions.

```
cmsrel CMSSW_8_0_20
cd CMSSW_8_0_20/src
```

```
cmsenv
```

No packages need to be checked, but for the sake of being more organized, you can create and work within **`FastSimulation/GenProduction`**.

SIMULATION WITHOUT PU:

For simulating a TTbar sample with no PU, use the following cmsDriver command. You can replace the TTbar generator fragment with your favorite fragment.

```
cmsDriver.py TTbar_13TeV_TuneCUETP8M1_cfi  --conditions auto:run2_mc --fast  -n 10 --era Run2_201
```

SIMULATION WITH PU:

There are two ways to simulate physics samples with PU: 1) standard mixing, 2) premixing, which is faster and more memory efficient, so should be used for generating large samples.

For both methods, we need a MinBias sample. Official production with 8_0_20 has been requested and can be followed from here.

In the meanwhile private MinBias samples can be generated using:

```
cmsDriver.py MinBias_13TeV_pythia8_TuneCUETP8M1_cfi \
--conditions auto:run2_mc \
--fast \
-n 10 \
--era Run2_2016 \
--eventcontent FASTPU \ --relval 100000,1000 \
-s GEN,SIM,RECOBEFMIX \ --datatier GEN-SIM-RECO \ --beamspot Realistic50ns13TeVCollision
```

Generating TTBar samples WITHOUT premixing:

Supposing that you have a MinBias file named **`MinBias_13TeV_pythia8_TuneCUETP8M1_cfi_GEN_SIM_RECOBEFMIX.root`** from the previous step, you can use the following command:

```
cmsDriver.py TTbar_13TeV_TuneCUETP8M1_cfi  \
--conditions auto:run2_mc \
--pileup_input file:MinBias_13TeV_pythia8_TuneCUETP8M1_cfi_GEN_SIM_RECOBEFMIX.root \
--fast  \
-n 10 \
--era Run2_2016 \
--eventcontent AODSIM \
--relval 100000,500 \
--step GEN,SIM,RECOBEFMIX,DIGI:pdigi_valid,L1,DIGI2RAW,L1Reco,RECO,HLT:@relval2016 \
--beamspot Realistic50ns13TeVCollision \
--datatier AODSIM \
--pileup AVE_35_BX_25ns \
```

Generating TTBar samples WITH premixing:

Supposing that you have a MinBias file named **`MinBias_13TeV_pythia8_TuneCUETP8M1_cfi_GEN_SIM_RECOBEFMIX.root`** from the previous step, you can use the following command:

1. First, generate a premixed sample:

```
cmsDriver.py SingleNuE10_cfi \
--pileup_input file:MinBias_13TeV_pythia8_TuneCUETP8M1_cfi_GEN_SIM_RECOBEFMIX.root \
```

```
--mc \
--eventcontent PREMIX \
--fast \
--pileup AVE_35_BX_25ns \
--datatier GEN-SIM-DIGI-RAW \
--conditions auto:run2_mc \
--step GEN,SIM,RECOBEFMIX,DIGIPREMIX,L1,DIGI2RAW \
--era Run2_2016 \
--no_exec
```

2. Then generate the physics sample overlaid with PU: Use the output of the premixed sample as a PU input, e.g., if the output file from the previous step is SingleNuE10_cfi_GEN_SIM_RECOBEFMIX_DIGIPREMIX_L1_DIGI2RAW_PU.root , that file should be used as input.

```
cmsDriver.py TTbar_13TeV_TuneCUETP8M1_cfi \
--pileup_input file:SingleNuE10_cfi_GEN_SIM_RECOBEFMIX_DIGIPREMIX_L1_DIGI2RAW_PU.root \
--mc \
--eventcontent AODSIM \
--fast \
--customise SimGeneral/DataMixingModule/customiseForPremixingInput.customiseForPreMixingInput \
--datatier AODSIM \
--conditions auto:run2_mc \
--beamspot Realistic50ns13TeVCollision \
--step GEN,SIM,RECOBEFMIX,DIGIPREMIX_S2,DATAMIX,L1,DIGI2RAW,L1Reco,RECO,HLT:@fake1 \
--datamix PreMix \
--era Run2_25ns \
--no_exec \
```

Recipe for upgrades (CMSSW_6_2_0_SLHC5)

```
cmsrel CMSSW_6_2_0_SLHC5
cd CMSSW_6_2_0_SLHC2/src
cmsenv

git-merge-topic Vlandr57:Segm_FastSim01
git cms-addpkg FastSimulation/Configuration
```

User can create the new branch

```
git checkout -b  "new branch name"
scram b -j4
```

And then go to the test directory

```
cd FastSimulation/Configuration/test
```

copy the necessary python file and run the job

```
cp  /afs/cern.ch/user/a/andreevv/public/CMSSW_6_2_0_SLHC5/FSIM_upgrade_cfg.py .
cmsRun FSIM_upgrade_cfg.py
```

Run2 Spring15 recipe

use CMSSW_7_4_*, CMSSW_7_4_4

create cmssw configuration files with the following cmsDriver.py commands

verify with your physics group that you are using the right conditions

generate minbias events to model pile up

```
cmsDriver.py MinBias_13TeV_pythia8_TuneCUETP8M1_cfi  --conditions MCRUN2_74_V9 \
--fast  -n 10 --eventcontent FASTPU -s GEN,SIM,RECOBEFMIX --datatier GEN-SIM-RECO \
--beamspot NominalCollision2015 \
--customise SLHCUpgradeSimulations/Configuration/postLS1Customs.customisePostLS1 \
--magField 38T_PostLS1 --fileout minbias.root
```

- Minbias samples will be provided centrally, so in principle there's no need to do this yourself. The dataset path will be provided as soon as the sample is ready for use. However, this runs very fast, so in some cases it might be more convenient to generate the minbias sample yourself.
- Events are generated at a rate of several events per second
- the required number of minbias events depend on the size of your final signal sample, but you want at least 30000 to run some basic tests

premix the minbias events for PU

```
cmsDriver.py SingleNuE10_cfi  --conditions MCRUN2_74_V9 --pileup_input file:minbias.root \
--fast  --eventcontent PREMIX -s GEN,SIM,RECOBEFMIX,DIGIPREMIX,L1,DIGI2RAW \
--datatier GEN-SIM-DIGI-RAW --pileup AVE_35_BX_25ns \
--customise SLHCUpgradeSimulations/Configuration/postLS1Customs.customisePostLS1 \
--magField 38T_PostLS1 -n 10 --fileout minbias_premixed.root
```

- this step takes as `--pileup_input` the output from the previous step
- !! edit the pileup option to obtain the correct pileup scenario.
  - pileup scenario used in RelVals: `-pile_up AVE_35_BX_25ns`
  - pileup scenario used for spring15 production with 25ns bunch spacing `--pileup 2015_25ns_Startup_PoissonOOTPU`
  - pileup scenario used for spring15 production with 50ns bunch spacing `--pileup 2015_50ns_Startup_PoissonOOTPU`
- Premixed minbias samples are provided centrally, so in principle there's no need to do this yourself. The dataset path is `/Neutrino_E-10_gun/RunIISpring15PrePremix-MCRUN2_74_V9-v1/GEN-SIM-DIGI-RAW` However, in some cases it might be more convenient to generate the minbias sample yourself.
- Each event requires about 5 CPU seconds
- The required number of premixed minbias events depends on the size of your signal sample. You want at least 1000 to run some basic tests

produce signal events and overlay with premixed minbias, store in AODSIM format

```
cmsDriver.py TTbar_13TeV_TuneCUETP8M1_cfi  --datamix PreMix --conditions MCRUN2_74_V9 \
--pileup_input file:minbias_premixed.root --fast -n 10 --eventcontent AODSIM \
-s GEN,SIM,RECOBEFMIX,DIGIPREMIX_S2,DATAMIX,L1,L1Reco,RECO,HLT:@relval25ns \
--datatier AODSIM --beamspot NominalCollision2015 \
--customise SLHCUpgradeSimulations/Configuration/postLS1CustomsPreMixing.customisePostLS1 --magFi
```

- this step takes as `--pileup_input` the output from the previous step
- Each events requires about 5 CPU seconds, depending on the PU scenario

produce signal events without PU, store in AODSIM format

```
cmsDriver.py TTbar_13TeV_TuneCUETP8M1_cfi  --conditions MCRUN2_74_V9 --fast  -n 10 --eventcontent
```

convert the AODSIM file to MINIAODSIM

```
cmsDriver.py step3  --conditions MCRUN2_74_V9 --fast  -n 10 --eventcontent MINIAODSIM \
--runUnscheduled  --filein file:ttbar.root -s PAT --datatier MINIAODSIM \
--customise SLHCUpgradeSimulations/Configuration/postLS1Customs.customisePostLS1 --mc
```

- this step takes as `--input` the output from the previous step
- Each event requires about 5 CPU seconds, depending on the PU scenario and the signal process

-- LukasVanelderen - 08 Jul 2014

This topic: CMSPublic > SWGuideFastSimulationExamples
Topic revision: r41 - 2020-07-06 - SamBein