Table of Contents

Full Simulation Offline Guide **1**
- Goal of this page 1
- Contacts 1
- Introduction 1
- Documentation 2
  - Tutorials 2
  - Detector Drawings 2
  - Related Software (Setups, Services, etc.) 2
  - Steps involved in the standard simulation procedure 2
  - Special purpose software 2
  - Data formats 3
- Special Procedures 3
  - Rerun a job with the same random numbers 3
  - How reproduce the Digi and/or DigiSimLink from a production DataSet 3
  - Pre-Mixing Instructions 4
    - Pre-Mixing Documentation 5
    - Reading Back Event Information in Pre-Mixed Samples 7
- multi-treaded 7
- Releases 7
  - Geant4 releases 7
  - CMSSW simulation release notes 8
  - CMSSW releases DropBox 8
  - Plans 8
  - Performance reports 8
- Validation 8
  - Meetings 8
  - Presentations and Papers, Picture and Plot Gallery 8

Full Simulation Offline Guide

Goal of this page

The goal of this page is to put together links to the detailed materials about full-scale Geant4-based CMS detector simulation software, including some information on the algorithms and implementation details. (Informations about the CMS Fast Simulation are here.)

This simulation procedure includes the following standard steps:

- Modeling of the Interaction Region
- Modeling of the particle passage through the hierarchy of volumes that compose CMS detector and of the accompanying physics processes
- Modeling of the effect of multiple interactions per beam crossing and/or the effect of events overlay (PileUp simulation)
- Modeling of the detector's electronics response (digitization)

In addition, detector simulation software includes tools for so called Special Tasks; links to dedicated materials on such tools are also collected in this document.

Contacts

- Conveners: **Daniel Elvira** and **Vladimir Ivantchenko**
- Hypernews: https://hypernews.cern.ch/HyperNews/CMS/get/simDevelopment.html

Introduction

The CMS Simulation Group aims to develop, maintain and support the CMS Simulation software, including detector simulation and digitization (signal shape, noise, calibration constants), in collaboration with the other software teams, the CMS Detector Performance and Physics Object groups, the LCG Applications areas and the various external tool providers.
Two detector simulations are under the responsibility of this group: the so called Full Simulation, based on Geant4 (topic of this wiki) and a Fast Simulation documented here.

The CMS simulation software is fully integrated within CMSSW framework. Its components can be integrated into the event processing chain via configuration application, that serves on input to the main CMSSW executable `cmsRun`.

The standard way to compose production-type configuration application is with the use of one of the CMSSW utilities called cmsDriver. Example usage has already been described in WorkBookGenIntro#ComposeFullSimConfig and WorkBookSimDigi.

For special purpose tasks a configuration application can be modified and/or entirely composed by hands, provided that a user is reasonably familiar with the software components involved in the CMS detector simulation process. We provide links to useful documents and tips that will help users to better understand the software and to customize their applications.

In addition we also collect in this document various simulation-related materials that we find useful, for example information of CMSSW releases and schedules or results of monitoring performance of the simulation application.

We also offer slides of talks on the CMS detector simulation software, as presented by the collaboration member at some of the recent conferences.

There service work required for Full Simulation. Have a look at the Full Simulation needs.

Documentation

Tutorials

Tutorials on the CMS software framework and its many elements are available in the WorkBook. In particular, the Chapter 6 concentrates on event generation and simulation:

- Quick introduction to event generation and simulation
- Generating events
- How to Configure and Run Detector Simulation and Digitization

Detector Drawings

- drawings for forward region

Related Software (Setups, Services, etc.)

Below are links to documents on the software components that are not a part of detector simulation, but are mandatory for running the simulation:

- SWGuideDetectorDescription - documentation of the Detector Geometry Description
- SWGuideMagneticField - documentation of the Magnetic Field
- Random Number Generator Service - materials available but need to be restructured properly

Steps involved in the standard simulation procedure

- SWGuideVertexSmearing - documentation on the software for modeling Interaction Region spread
- SWGuideOscarProducer - documentation on the Geant4-based CMS Detector Simulation module (incl. configuration parameters)
- SWGuideMCTruth - documentation on MC truth handling in CMSSW
- SWGuideMixingModule - documentation on the software for superimposing secondary (pileup, etc.) events to a signal event stream
- Digitization
    - Tracker digitization - documentation on the software for modeling electronics response in the Tracker (Strips and Pixels)
    - Calorimeter digitization - documentation on the software for modeling electronics response in the EM and HAD Calorimeters
    - Muon digitization - documentation on the software for modeling electronics response in the Muon Detectors

Special purpose software

- CMS.DataMixer - documentation of the software for overlying experimental data (zero bias, etc.) on top on the **simulated** signal event
- SWGuideFwdPhysics - documentation on the Forward Physics software, including simulation
- CMS.ZdcSimulationandDigitization - documentation on the simulation and digitization of the Zero Degree Calorimeter (ZDC)

Data formats

- SWGuideDataFormatGeneratorInterface - generated event; related to Vertex Smearing and serves as input to Detector Simulation
- SWGuideDataFormatSimG4Core - data products stored in RECO samples

Formats of other simulation-related products can be examined via

- CMSSW Reference Manual

Select your desired release or even a pre-release (we suggest the most recent) and follow the link "SIM Data".

INSTRUCTION OF EVENT REPRODUCIBILITY ARE BEING UPDATED

Special Procedures

Rerun a job with the same random numbers

By default the random number state is saved in the Event in all the jobs. Then to rerun a job using exactly the same random numbers you have just to configure properly the random number generator service.

For pre-331 jobs, the easiest procedure to restore the state of the random number generators is based on adding these lines at the end of your configuration file:

```
def customise(process):

    if hasattr(process,"RandomNumberGeneratorService"):
        del process.RandomNumberGeneratorService.theSource
    else:
        process.load("IOMC/RandomEngine/IOMC_cff")
        del process.RandomNumberGeneratorService.theSource

    process.RandomNumberGeneratorService.restoreStateLabel = cms.untracked.string('randomEngineSt

    return(process)

# End of customisation function definition

process = customise(process)
```

N.B. The if statement is mandatory up to now, as presently it is not possible to use the restore mechanism for the generator part.

Post 3_3_2, the Generator part of the workflow is no longer in the source, but a separate module, so the deletion of theSource is unnecessary. The line

```
process.RandomNumberGeneratorService.restoreStateLabel = cms.untracked.string('randomEngineStateP
```

should be all that is required.

How reproduce the Digi and/or DigiSimLink from a production DataSet

There are basically two different use cases to reproduce the Digi.

- The digitization conditions have changed and you want to evaluate the effects.
- You need to access the Digi and/or DigiSimLink for validation.

The first case is the so called REDIGI work flow. Once you know the input DataSet, there exists a cmsDriver.py command that allows you to do that. The command is:

```
cmsDriver.py test -n 10 --filein output_file_of_step1_of_interesting_dataset.root -s DIGI,L1,DIGI
```

This command produces a configuration file that runs over the output of step1 and produces as output two separate files: a new RAW file (step1 output) and a new RECO file (step2 output). In case you want to change the event content of the new RECO file, you have just to change the --eventcontent option of the cmsDriver.py command. Instead, if you wanted to change the event content of the new RAW file, you have to manipulate the configuration file produced by the cmsDriver.py command. In this case search for the string ***outputCommands = process.RAWSIMEventContent.outputCommands,*** and replace RAWSIM with your favorite event content.

Please remind that changing the event content from ***RAWSIM*** to one of the ***FEVT**** family will produce as output an enormous file. Then, before doing that, evaluate carefully the disk space that you have.

In the second use case, instead of producing a new RAW file to be used for the validation, the best strategy is to redo the digitization on the fly. In fact the digitization is really a fast step (~ 2 sec/evt for TTbar events). To do so you have to modify as follows the configuration file that you are using for the validation:

1. Add the following includes**

   ```
   process.load('Configuration/StandardSequences/Digi_cff')
   process.load('Configuration/StandardSequences/MixingNoPileUp_cff')
   ```
2. In the Path definition add this line

   ```
   process.digitisation_step = cms.Path(process.pdigi)
   ```
3. In the schedule as first operation to do put the digitization i.e.

   ```
   process.schedule = cms.Schedule(process.digitisation_step,
   ```
4. At the end of the file add the lines described in the previous paragraph

In case you are interested to rerun only a particular digitizer you have to replace the point 2 above with one of the following lines:

- If you are interested only to tracker digis

  ```
  process.digitisation_step = cms.Path(cms.SequencePlaceholder("mix")*process.trDigi*proces
  ```
- If you are interested only to calorimeter digis

  ```
  process.digitisation_step = cms.Path(cms.SequencePlaceholder("mix")*process.calDigi)
  ```
- If you are interested only to muon digis

  ```
  process.digitisation_step = cms.Path(cms.SequencePlaceholder("mix")*process.muonDigi*proc
  ```

** If you are interested to redigitize events with PU you have to change the include of MixingModule with the appropriate configuration (i.e. LowLumi, HighLumi,...) and, in addition, to specify in the MM configuration the same files that have been used for the PU during the production. Finally you have to add this line in your configuration file

```
process.mix.playback = cms.untracked.bool(True)
```

Pre-Mixing Instructions

Pre-Mixing is a way of circumventing the need for juggling a huge number of input files of Minbias events for Pileup simulation. "Pre-Mixed" files contain single events that are an overlay of a pre-determined number

(or spectrum) of minbias events, so each event represents "pure pileup" at a given luminosity. One of these events is then overlaid on a single hard-scatter event to produce a MC sample. The required inputs are (all with **`GEN-SIM`** event content):

- A large sample of minbias events
- The target hard-scatter MC sample

To create the Pre-Mixed sample, the overlay sample has to be done first. In order to get the right number of interactions, overlay on SingleNeutrino events is the correct thing to do:

```
cmsDriver.py step1 --evt_type SingleNuE10_cfi  --conditions auto:upgradePLS1 --pileup_input das:/
```

Note that it's very important that there be no detector noise in this file; otherwise it will be added twice. This makes a file in RAW format so that it is smaller. (Note: new event content. Updated 23 April 2014)

Next, the Pre-Mixed file is used as input for the DataMixer. Note that it must be assigned the logical filename **`DMPreProcess_RAW2DIGI.root`** (even though it isn't a Digi file) because that is the input the secondary input stream of the DataMixer is expecting. Bill Tanenbaum added the capability to read raw data on the secondary input stream and process it into digis, so the newest verion of the DataMixer sequence includes a RawToDigi step on the secondary stream. To mix:

```
cmsDriver.py step2  --datamix PreMix --conditions auto:upgradePLS1 --pileup_input das:/RelValPREM
```

Note that the input file is expected to have GENSIM content.

For PostLS1 Customisation: For all steps, the modified customisation **`--customise SLHCUpgradeSimulations/Configuration/postLS1CustomsPreMixing.customisePostLS1`** should be used in the **`cmsDriver.py`** command set, instead of `postLS1Customs`. This keeps the necessary collections for PreMixing while side-stepping the current problem of the missing CSC Raw2Digi and Digi2Raw converters for the post LS1 configuration.

For validation purposes, one can use the standard additional tag **`DIGIPREMIX_S2:pdigi_valid`** on the digitisation step.

Pre-Mixing Documentation

The table below contains a description of what PreMixing does for each subdetector. A short overview of the mixing steps:

1. PreMixing: "Standard" minbias events in RAWSIM format (with just genParticles and SimHits as input) are overlaid on empty single neutrino events using a chosen pileup configuration (# of interactions, # of bunch crossings, etc.). For most subdetectors (except muons and some SiPixel settings), no detector noise or pedestals are introduced in this step. Zero suppression thresholds in the calorimeter are set to zero. The digis made in this step are converted to RAW format to save space (with the exception of the RPCs, for which packing-unpacking is not a reversible process). There is a special **`PREMIX`** event content that is an extension of **`GENRAW`**, which contains the extra RPC digis and the pileup playback information.
2. Mixing: In this step, a single PreMixed event (which represents all of the "extra" signals that would come from pileup at the chosen configuration) is combined with one MC Hard-scatter event. This is effectively done at the Digi level, using the DataMixer machinery. A RawToDigi conversion is done on-the-fly for the PreMixed events within the Secondary Input Stream used for the overlay. The pileup information for the bunch crossings in the PreMixed event and the associated pileup playback information are passed through the DataMixer to the final event. A more detailed description of what happens for each subdetector is given in the table below.

SubDetector	PreMixing Treatment	Mixing Treatment	comments
SiPixels	noise added in channels adjacent to actual hits; no additional noisy pixels added	PreMixed digis combined directly with digis from Hard Scatter. Noise hits generated adjacent to Hard Scatter hits. Noisy pixels added here. Dynamic Inefficiency implemented at this stage as well.	Standard swtiches in the code are used to control noise addition.
SiStrips	no noise added	PreMixed signals have special treatment (ADC=sqrt(PH)) to save low level hits. Special 10-bit storage mode in packing/unpacking is used for saving complete information. Signals from PreMixed digis and Hard Scatter digis on individual strips are combined, then noise is generated for each detector using combined signals. No noise/peds are generated in digitization of hard scatter signals before combination.	Noise generation in DataMixer code (`DataMixingSiStripWorker`) using code/libraries from `SiStripDigitizer`.
EB/EE	no noise/pedestals added, no Zero Suppression (DB setting)	PreMixed signals have special digitization procedure that directly keeps charge in p.e.. Hybrid encoding is used for larger energy PreMixed hits up to Emax/2. PreMixed signals treated as additional "noise", combined at raw p.e. sample-level with p.e. signals from Hard Scatter SimHits. Then, noise, pedestals, ZS applied to combined signals. (see `EcalSimAlgos/interface/EcalSignalGenerator.h` and its hooks in `EcalCoder`)	Custom switches added for noise handling. Trigger Primitive generation input redirected to consume combined output from DataMixer
ES	no noise/pedestals added, no Zero Suppression (Workflow bypass)	PreMixed signals treated as additional "noise", combined at raw p.e. sample-level with p.e. signals from Hard Scatter SimHits. Then, noise, pedestals, ZS applied to combined signals. (see `EcalSimAlgos/interface/EcalSignalGenerator.h` and its hooks in `EcalCoder` and `EcalTDigitizer`)	Custom switches added for noise handling.
Hcal	no noise/pedestals added, no Zero Suppression (Config NZS setting)	PreMixed signals treated as additional "noise", combined at raw p.e. sample-level with p.e. signals from Hard Scatter SimHits. Then, noise, pedestals, ZS applied to combined signals. (see `HcalSimAlgos/interface/HcalSignalGenerator.h` and its hooks in `HcalAmplifier` and `CaloTDigitizer`)	Custom switches added for noise handling. Trigger Primitive generation input redirected to consume combined output from DataMixer
CSC	"normal" simulation of all hits (including noise)	"normal" simulation of Hard-Scatter hits, Digis combined with PreMixed Digis.	Trigger Primitive generation input redirected to consume combined output from DataMixer. Fingers crossed on overlapping hits.
DT	"normal" simulation of all hits (including noise)	"normal" simulation of Hard-Scatter hits, Digis combined with PreMixed Digis.	Trigger Primitive generation input redirected to consume combined output from DataMixer. Fingers crossed on overlapping hits.
RPC	"normal" simulation of	"normal" simulation of Hard-Scatter hits, Digis combined with PreMixed Digis (no RawToDigi).	Trigger Primitive generation input redirected

	all hits (including noise). Digis saved directly to output without conversion to RAW.		to consume combined output from DataMixer. Fingers crossed on overlapping hits.

Reading Back Event Information in Pre-Mixed Samples

The `CrossingFrameInfoExtended` object is made by the MixingModule and contains a list of which events are used in each of the beam crossings. This information is pretty easy to extract from the event. However, one needs to know **exactly** which minbias files were used to make the original sample in order to find the correct events. In any case, a stand-alone analyzer `PlayBackRead.cc`, its header `PlayBackRead.h` and a config script `playBackRead_cfg.py` are all attached to this page. They work in 7_0_7 and later. Links to files: playBackRead_cfg.py, PlayBackRead.cc, PlayBackRead.h

multi-treaded

Within CMSSW_7_2_GEANT10 branch development of multi-threaded simulation is carried out. For details see working notes.

Releases

Visit the CMSSW Release Web Page for up-to-date information on dates for CMS software releases. We also collect records on the evaluation of the CMS simulation application based on different Geant4 releases; for details, visit the CMS-Geant4 document. Additional information on monitoring technical aspects of the CMSSW/Simulation performance can be found through the Simulation Performance Suite Reports.

Please visit also pages below with the list of feature/goals planned and implemented for the various release cycle.

Validation of different Geant4 versions is performed by PPD group PdmV.

Geant4 releases

CMS simulation is based on public Geant4 versions. Links to Geant4 release notes:

- 10.1
- 10.0p04
- 10.0p03 used since CMSSW_7_5_0_pre1
- 10.0p02 used since CMSSW_7_2_0_pre2
- 10.0p01 used since CMSSW_7_1_0_pre6
- 10.0 used in CMSSW_7_1_X_GEANT10 branch
- 9.6p02 used since CMSSW_6_2_0_pre8
- 9.6p01 used since CMSSW_6_2_0_pre2
- 9.6
- 9.5p02 used since CMSSW_6_1_X
- 9.5p01 used since CMSSW_6_0_X
- 9.5
- 9.4p03 used since CMSSW_5_0_X
- 9.4p02

- 9.4p01
- 9.4 used since CMSSW_4_2_X

CMSSW simulation release notes

- Release notes for 7_X_Y
- Release notes for 6_X_Y
- Release notes for 5_X_Y

CMSSW releases DropBox

- Simulation DropBox for 4_2_X
- Simulation DropBox for 3_10_X
- Simulation DropBox for 3_9_X
- Simulation DropBox for 3_8_X
- Simulation DropBox for 3_7_X
- Simulation DropBox for 3_5_X
- Simulation DropBox for 3_6_X

Plans

- 2014
- 2013
- 2012
- Old simulation plans

Performance reports

- FullSimulationPerformanceMonitoring

Validation

- Ongoing Validations
- Summary Validation Table for 4_4_X and before
- Simulation Validation Table for 4_3_X and before

Meetings

The group meets at CERN every week, on Tuesday 17:00-18:30hs CERN time, see CDS Agenda System for CMS offline meetings. List of Action Items from simulation meetings are available here.

Presentations and Papers, Picture and Plot Gallery

In case you need to cite a document on simulations, we keep here the most recent conference presentations and papers:

- CMS Full Simulation: Evolution toward the 14 TeV run (M. Hildreth, V.Ivanchenko), CHEP 2013 slides
- Strategies for Modeling Extreme Luminosities in the CMS Simulation (M.Hildreth), CHEP 2013 slides
- CMS Simulation Software (S. Banerjee), CHEP 2012 local pdf file,CMS Doc link to full entry

- Validation of G4 Models with Collision Data (S. Banerjee), CHEP 2010 local pdf file, CMS Doc link to full entry
- Validation and Tuning of CMS Simulation Software (S. Banerjee, M. Hildreth), CHEP 2010 local pdf file, CMS Doc link to full entry
- Data-Driven Approach to Calorimeter Simulation (S. Banerjee) CHEP 2009 local pdf file, CMS Doc link to full entry
- Calorimetry Task Force Report (S. Abdullin et al.) CMS-NOTE-2010-007.- Geneva : CERN, 2010 - 25 p. Fulltext: PDF
- CHEP 2009 (Fabio Cossutti): slides, paper
- IEEE/NSS 2007 (V.Daniel Elvira): slides, paper
- CHEP 2007 (Sunanda Banerjee): slides, paper
- IEEE 2006 (Julia Yarba): slides, paper to be added
- IEEE/NSS 2005 (M. Stavrianakou): slides, paper

This topic: CMSPublic > SWGuideSimulation
Topic revision: r99 - 2017-09-19 - VDanielElvira