Table of Contents

Heavy-Ion Data Mixer Recipes

Complete: 🟩

Development in 4_4_X

CMSPublicCmsHiMit2011 CMSPublicCmsMit2011

HiDataMixer2011

Fix for the calorimeter condition setup : Use this patch on top of the default data mixing setup.

SWGuideHeavyIonDataMixerTemp

Goal of this page

The aim of this page is to provide instructions for embedding interesting signals from either data or MC into heavy-ion backgrounds using the data mixer functionality.

The **primary source** of information on the use of the data mixer should be the extensive description contained on the main page: DataMixer

Code and tags

As described at DataMixer#Current_Release_Setup_and_Runnin, the data mixer tools have been available since CMSSW_3_3_0. Please find the latest tag of `SimGeneral/DataMixingModule` at that page (e.g. for backward-compatible features of 3_8_X, take `V00-02-13`).

Additional related packages are:

- `SimGeneral/MixingModule` (for matching embedded GEN vertex to RECO vertex)
- `Configuration/GenProduction` (to get the latest MC embedding configurations)

A prescription for setting up a project area with a consistent set of tags follows below.

Quick Instructions

If one doesn't care about matching the vertex positions of the simulated signal and the background data event, then one can simply follow the instructions on the main DataMixer page. To do mixing that includes either (a) matched vertex positions or (b) the combination of signal SiStripDigis with background SiStripRawDigis, use the following setup prescription:

In 394:

```
cvs co -d edwenger/DataMixingModule UserCode/edwenger/DataMixingModule
cd edwenger/DataMixingModule
./setupDM.sh
cd -
scram b
```

In 392patch5:

```
cvs co -r v392p5 -d edwenger/DataMixingModule UserCode/edwenger/DataMixingModule
cd edwenger/DataMixingModule
./setupDM.sh
cd -
scram b
```

Configuration Examples

Two examples are listed here to illustrate different use-cases:
(1) generating on-the-fly and embedding a Z μμ particle gun at a matched vertex position to a heavy-ion background data event
(2) mixing Z ee and W e skimmed p+p events into heavy-ion background events

---

Example 1 -- Embed MC Z μμ event into VR heavy-ion data:

This example consists of three steps: (1) pre-process RAW background events saving the digis for mixing, (2) generating the MC signal and mixing the simulated digis with a secondary source from the first step, (3) reconstructing the RAW output of the mixing step. The configurations assume that the setupDM.sh script described in the above instructions has been run in CMSSW version 392patch5 (possibly also works for any release after 385 or 390pre7).

(1) *Pre-Processing*

preDM.sh: a script to generate a configuration that runs RAW2DIGI and saves digis for later mixing. Note that the default event content is extended, since we want to read the reconstructed vertex in the next step as well as keep the VirginRaw strip digis.

```
def customiseDataMixerOutput(process):
    process.DATAMIXERoutput.outputCommands.extend(
        ['keep *_hiSelectedVertex_*_*',
         'keep *_siStripDigis_VirginRaw_*'])
    return process
```

Note that this first configuration is the appropriate place to add any trigger and event selection filters one wants to apply. For more details, see HIData2010CollisionEventSelection.

(2) *Overlaying Signal*

simOnRecoDM.sh: a script to generate a configuration that runs GEN,SIM,DIGI,DATAMIX,L1,DIGI2RAW,HLT:HIon taking the output of the first step as a secondary source. Note that we make several modifications to the normal prescription to ensure that the generator has the right label, that the generated vertex matches the reco data vertex (modulo the usual pixel detector offset), and that the overlaying of strip signals uses the rawdigis for the background.

```
def customiseCloneGenerator(process):
    process.generator = process.hiSignal.clone(embeddingMode=cms.bool(False))
    process.ProductionFilterSequence.replace(process.hiSignal,process.generator)
    return process

def customiseMatchRecVertex(process):
    process.load("SimGeneral.MixingModule.MatchRecVtx_cfi")
    process.pgen.replace(process.VertexSmearing,process.matchRecVtx)
    process.mixData.input.sequential = True # overlay signal on same background event used to mat
    process.mixData.input.fileNames = process.source.fileNames
    return process
```

```
def customiseSiStripRawDigi(process):
    process.mixData.SiStripRawDigiSource = "PILEUP" # use raw digis from "PILEUP" or "SIGNAL"
    process.SiStripDigiToRaw.FedReadoutMode = cms.string('VIRGIN_RAW') # pack rawdigis as VR

    process.stripConditions = cms.ESSource("PoolDBESSource",
        process.CondDBSetup,
        timetype = cms.untracked.string('runnumber'),
        connect = cms.string('frontier://FrontierProd/CMS_COND_31X_STRIP'),
        toGet = cms.VPSet(cms.PSet(
          record = cms.string('SiStripFedCablingRcd'),
          tag = cms.string('SiStripFedCabling_GR10_v1_hlt')))
    )
    process.es_prefer_strips = cms.ESPrefer("PoolDBESSource","stripConditions")
    return process
```

Note that one must be careful not to apply any filters (e.g. mumugenfilter for both decay muons in acceptance) at this step, since this will cause the events in `source` and the `mixData.input` to fall out of sync, obviating the setting `sequential=True`. Instead, filters should be postponed until the reconstruction step.

(3) *Reconstruction*

recoDM.sh: a configuration to run RAW2DIGI,RECO on the mixed RAW events from the previous step. Since the tracks come from the data beamspot position shifted into the MC coordinate system, we use a custom beamspot object.

```
def customiseBeamSpot(process):
    process.GlobalTag.toGet = cms.VPSet(
        cms.PSet(record = cms.string("BeamSpotObjectsRcd"),
                 tag = cms.string("Realistic2.76ATeVCollisions_STARTUP_v0_mc"),
                 connect = cms.untracked.string("frontier://FrontierProd/CMS_COND_31X_BEAMSPOT"))
    return process
```

It is also crucial that the data conditions are used for the tracker noise, quality, pedestals, etc.

Also, one can factorize any effects on the pixel vertexing of shifting from data to MC misalignment by just using a dummy vertex that is exactly at the embedded GEN vertex position:

```
def customiseDummyVtx(process):
    process.hiSelectedVertex = cms.EDProducer("GenToRecoVtxProducer",
        signalLabel=cms.InputTag("generator"))
    return process
```

This has been extended to allow user-specified smearing around the GEN vertex position, which is then reflected in the assigned vertex position errors.

Finally, if one desires to have the same top-level configuration replacements as are executed during prompt and express reconstruction, those can also be added in this step.

Example 2 -- Embed Z ee event from p+p data into p+p VR high-multiplicity event:

This example consists of three steps: (1) pre-process RAW background events saving the digis for mixing, (2) mixing the digis produced on-the-fly from another data event with a secondary source from the first step, (3) reconstructing the RAW output of the mixing step. The configurations assume that the setupDM.sh script described in the above instructions has been run in CMSSW version 394.

(1) *Pre-Processing*

preDM.sh: a script to generate a configuration that runs RAW2DIGI and saves digis for later mixing. Same setup as for mixing MC into HI RAW data.

As in the MC example, this first configuration is the appropriate place to add any trigger and event selection filters one wants to apply. For more details, see HIData2010CollisionEventSelection.

(2) *Overlaying Signal*

rawRecoOnRecoDM.sh: a script to generate a configuration that runs RAW2DIGI,DATAMIX,L1,DIGI2RAW,HLT:HIon to embed events from the W and Z EGamma skim, `/Photon/Run2010B-WZEG-v2/RAW-RECO`, into the pre-processed heavy-ion events from the previous step.

Note that unlike in the previous MC example, there is currently no effort to match the z-vertex positions of the signal and background events.

(3) *Reconstruction*

dataRecoDM.sh: a script to generate a configuration that runs RAW2DIGI,L1Reco,RECO on the heavy-ion events embedded with signal processes from other data events. Because there is no attempt to match the vertex position of the two parts of the combined event, the parts of the reconstruction that are dependent on a vertex position are explicitly passed the already-reconstructed value from the signal event.

```
def customiseVtxPP(process):
    # Run RECO around p+p vertex from embedded RAW-RECO event
    #process.hiCentrality.srcVertex = 'offlinePrimaryVertices' #variable not yet in 394
    process.photons.primaryVertexProducer = 'offlinePrimaryVertices'
    process.hiPixel3PrimTracks.RegionFactoryPSet.RegionPSet.VertexCollection = 'offlinePrimaryVer
    process.hiPixel3PrimTracks.FilterPSet.VertexCollection = 'offlinePrimaryVertices'
    process.hiTracksWithLooseQuality.vertices = 'offlinePrimaryVertices'
    process.hiTracksWithTightQuality.vertices = 'offlinePrimaryVertices'
    process.hiSelectedTracks.vertices = 'offlinePrimaryVertices'
    return process
```

Related Links

- DataMixer
- TrackEmbeddingNote
- TrackEmbeddingCode
- http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/UserCode/edwenger/DataMixingModule/

Contact

- **Hypernews fora**:
    - Software issues: https://hypernews.cern.ch/HyperNews/CMS/get/hiswDevelopment.html, (hn-cms-hiswDevelopment@cernNOSPAMPLEASE.ch)
    - General heavy ions issues https://hypernews.cern.ch/HyperNews/CMS/get/hi.html, (hn-cms-hi@cernNOSPAMPLEASE.ch)
- **Contacts/Developers**: Yetkin Yilmaz, Edward Wenger

Review status

Reviewer/Editor and Date	Comments
EdWenger - 12 Oct 2010	first draft of documentation

Responsible: EdWenger

This topic: CMSPublic > SWGuideHeavyIonDataMixer
Topic revision: r13 - 2011-10-28 - YetkinYilmaz