Here we will have to document the changes in the 03Feb2017 re-miniAOD and any associated recipes and recommendations. The corresponding code in github is in https://github.com/cms-sw/cmssw/pull/17308

Datasets

When there were multiple AOD datasets for a given era, a version label has been appended to the `03Feb2017` to distinguish between the outputs coming from the different inputs (and which should not be confused with the version of the MINIAOD dataset, the final `-v(number)` in the name, which is just the number of attempts it took to get the dataset successfully processed by the production system).

- For **Run2016B**, you will find 2 datasets called `Run2016B-03Feb2017_ver1` and `Run2016B-03Feb2017_ver2` ; these map to the RAW `Run2016B-v1` and `Run2016B-v2`
  You can normally skip the ver1 dataset as it does not contain any runs in the golden JSON.
- For **Run2016H**, you will find two datasets called `Run2016H-03Feb2017_ver2` and `Run2016H-03Feb2017_ver3`; these map to the `PromptReco-v2` and `PromptReco-v3` AODs.
  You should use both datasets, as they do not overlap.

EGM

READ THIS: It would be extremely advisable to store the bool particleFlowEGammaGSFixed:dupECALClusters and if ecalMultiAndGSGlobalRecHitEB:hitsNotReplaced is empty or not in any analysis job. If particleFlowEGammaGSFixed:dupECALClusters = true or ecalMultiAndGSGlobalRecHitEB:hitsNotReplaced is not empty, the fix may not have worked for that event and it needs to be manually recovered or at least scrutinized further. This events should be extremely rare but are possible. If you want to save which objects had a gain switch in at least one of its hits, see the userInts in this section.

The standard collections store the objects after the ECAL slew rate mitigation . We also store the old collections for comparison. A technical description of the changes can be found in a dedicated wiki.

With mitigation	Original collection
slimmedElectrons	slimmedElectronsBeforeGSFix
slimmedPhotons	slimmedPhotonsBeforeGSFix
reducedEgamma	reducedEgammaBeforeGSFix

Short Description of E/gamma Collections

This only details the fixed collections. To get the original collections add "BeforeGSFix" to the producer name as shown above.

show short description of E/gamma collections hide short description of E/gamma collections

name	notes
slimmedElectrons	all barrel electrons have supercluster updated to the reclustered one, only electrons with a gain switched crystal in the 5x5 of the seed crystal have their energies and showershapes updated
slimmedPhotons	all barrel photons have supercluster updated to the reclustered one, only photons with a gain switched crystal in the 5x5 of the seed crystal have their energies and showershapes updated
reducedEgamma:reducedEBEEClusters	the fixed ECAL clusters comprising the superclusters of

	the slimmed electrons and photons. Endcap clusters are copies of the original while barrel clusters are reclustered (except when the barrel refined supercluster has no parent supercluster).
reducedEgamma:reducedESClusters	the preshower clusters associated to the superclusters of the electrons and photons. These are copies of what was originally in the miniAOD.
reducedEgamma:reducedGedGsfElectronCores	the electron cores of the slimmedElectrons which contain references to the superclusters that have been gain switch fixed if appropriate
reducedEgamma:reducedGedPhotonsCores	the photon cores of the slimmedPhotons which contain references to the superclusters that have been gain switch fixed if appropriate
reducedEgamma:reducedConversions	the conversions linked in the slimmedElectrons and slimmedPhotons objects are the same as in the original objects, ie not gain switch fixed! (this was for technical reasons)
ecalMultiAndGSGlobalRecHitEB:hitsNotReplaced	its not guaranteed that all GS hits are in the AOD, this collection has the DetId of all hits which were in the barrel, had a GS but could not be replaced. Should be empty, if you find it non-empty for your events, re-reco it from the RAW.
particleFlowEGammaGSFixed:dupECALClusters	it is possible for the same supercluster (or cluster) to be added to the event multiple times. We think we got all the cases this happens but can not guarantee it. This is a bool indicating if this happened, if this is set to "true" the event needs further scrutiny.

Key Caveats with the Fix

This fix is mostly but not completely transparent to the user. Some small things changed. Read this section for detailed information on all the caveats associated with the fix.

show list of caveats hide list of caveats
First, if something is unclear, please help us improve the documenation by email your query to hn-cms-egamma@cernNOSPAMPLEASE.ch.

Second, this information is a copy of EcalSlewRateMitigation#Key_Points_to_be_Aware_of

1. Barrel refined superclusters are always remade, even if there is no gain switch
2. The energy and showershape of a electron/photon is only remade if there is gain switched crystal in the 5x5 area centred on the seed crystal
3. Issues 1) and 2) mean that the energy of the supercluster may be not be fully consistent with the electron/photon energy/showershape when its non-gainswitched, this is thought never to happen but be warned
4. The fiduical flags of electrons and photons are not updated and reflect the seed crystal of the old supercluster. This is by construction within +/-1 crystal of the new supercluster. The impact of this should be extremely small but be warned.
5. It is possible for a gain switched crystal not to be in the selectedEcalDigis and therefore not replaced. This happens extremely rarely. The collection "ecalMultiAndGSGlobalRecHitEB:hitsNotReplaced" has the DetIds of all effected hits and users should check that it empty for their events. If it is not, it indicates a possible problem and the event should be further scrutinised. To give a sense of scale, in the entire Z' analysis, we found 18 such events, all at the Z peak.

6. It is theoretically possible for this effect to split a barrel supercluster into two. In this case the new merged barrel supercluster matches to both of the orginal superclusters and gets added in twice. These events are passed through but are flagged by a bool, "particleFlowEGammaGSFixed:dupECALClusters" which users should explicity check. If true, it means duplicate basic clusters were added into the event and further scrutiny should be applied to the event as it may have duplicated electrons and photons. We currently know of no events where this occurs as we fixed all the cases where it did happen.
7. Orphan refined superclusters are not remade and simply passed through. This should have little practical consquence but it does mean that its subclusters are the orginal PFClusters and not gain switch fixed. It is hard to imagine a senario where a high energy electron / photon has just an orphan supercluster. This also means that things are not completely consistent in this case but again is thought to have little practical effect
8. Particle flow is not re-run and it may be instances of corrected electrons/photons that would fail PF identification when the original one passed (or vice-versa). Those cases are rare, but it's a possibility.
9. More on particle flow objects: even though they are linked correctly to the new superclusters, the energy of the PFelectrons and PFphotons is not corrected!
10. Due to technical reasons, the supercluster that the conversion links to is the one in the old electron. Be careful when using them, if you need to refer back to the supercluster. The problem is the following: while it's not hard to remap the conversions, it's impossible to replace the conversions inside the GsfElectron object. And, at the same time, we cannot make a new collection, because we don't have the track collection in AOD.
11. E/gamma only fixed the E/gamma objects. We did not propagate this input to other POGs, specifically taus and b-tagging.
12. We cannot guarantee that the recHit fractions inside the clusters will remain the same. See this HN thread for a discussion about the consequences.

Expert Information

show expert information... Hide description...
Users can check if an electron or a photon has a crystal in the EB with gain switch by using the following test:

```
electron.userInt("hasGainSwitchFlag") == 1;
photon.userInt("hasGainSwitchFlag") == 1;
```

We do not store a map between objects before and after the mitigation in miniAOD. The matching can be done via the ID of the seed crystal with this type of test:

```
electron.superCluster()->seed()->seed().rawId() == oldelectron.superCluster()->seed()->seed().raw
```

In rare occasions, it is possible that a cluster had a crystal with gain switch but we did not have the DIGI in AOD to mitigate the problem. We save the collection of DIGI used in the mitigation code, so that expert users can check if this rare situation can have an impact in their analysis. They are saved as (selectDigi, selectedEcalEBDigiCollection) and (selectDigi, selectedEcalEEDigiCollection). In the case an expert user would like to perform this advanced check, we provide a function that returns the list of detId with gain switch problems in this function.

Muons and PF Candidates

In this re-miniAOD, the filters for bad and duplicate muons advertised on physics-validation/2786 are used.

Event flags

Three flags are saved in the event:

- **`Flag_badMuons`**: the event contained at least one PF muon of $p_T > 20$ GeV that is flagged as bad
- **`Flag_duplicateMuons`**: the event contained at least one PF muon of $p_T > 20$ GeV that is flagged as duplicate
- **`Flag_noBadMuons`**: the event does not contain any PF muon of $p_T > 20$ GeV flagged as bad or duplicate (i.e. the event is safe)

They can be accessed from the **`TriggerResults`** just like the other MET filters.

Muons

The **`slimmedMuons`** collection contains all muons, as before.

However, the muons flagged as bad or duplicate are now no longer marked as PF muons; thus, **`isPFMuon()`** will return false on them (and consequently also **`isLooseMuon`**, **`isTightMuon`**, etc.). Nothing else of the muon is changed (e.g. the 4-vector is as given by the particle flow)

You can access the old value of the PF id flag via **`muon.userInt("muonsCleaned:oldPF")`** . If you want to *make the muon PF again*, you can copy it by value and then do **`muon.setType(muon.type() | reco::Muon::PFMuon)`**.

candidates & PUPPI weights

The bad and duplicate muons are removed from the **`packedPFCandidates`** list, and put in a separate collection **`packedPFCandidatesDiscarded`**; thus, the `sourceCandidatePtr` for bad or duplicate `slimmedMuons=` will therefore point to that second collection.

The PUPPI weights are computed from the cleaned PF candidates. The discarded PF candidates do not have a valid PUPPI weight.

MET Recipes

The color scheme of the instructions is as follows Hide

- Commands will be embedded in grey box, few examples:

```
cmsRun ConfFile_cfg.py
```

- Things to be edited in a configuration or a c-file will be embedded in blue box, e.g.

```
ConfFile_cfg.py
```

- Output and screen printouts will be embedded in green box, e.g.

```
Event 1: MET Pt = X
```

General Comments for MET Recipes

We strongly encourage everyone to switch to the CMSSW_8_0_26_patch1 release, which has been used for re-miniaod production. After switching to this release you will merge the default MET recipe.

```
git cms-merge-topic cms-met:METRecipe_8020 -u
```

In this MET recipe you will find:

- Latest / greatest MET Filters
- Bug fixes related to MET Significance computation

- Implementation of the Smeared MET and JER uncertainties
- Bug fixes for puppi met specifically on the puppi photon treatment

You can find the full PR and the details here. There is no need to merge anymore any further recipes individually. However the instructions on how to use the different filters & functions specified on twiki1 and twiki2 will still hold.

What is in re-miniaod Data?

In the re-miniaod data you will find 4 different pf MET collections. This is to ensure the user has the maximum flexibility to use any level of e/gamma or muon corrected MET. The full list and the description is provide below. Please note that, for the e/gamma corrected MET, the global event description is not possible (i.e. jets are not e/gamma corrected). This is because e/gamma corrections are not performed at the pf candidate level. Therefore, e/gamma corrected MET can be viewed as "Type1 e/gamma corrected MET".

Name	Description
slimmedMETs	This collection is the muon cleaned MET only.
slimmedMETsEGClean	This collection is the e/gamma cleaned MET only.
slimmedMETsMuEGClean	This collection is the muon and e/gamma cleaned MET.
slimmedMETsUncorrected	This is the uncleaned MET collection.

Note that calo MET is stored only in the slimmedMETs collection.

For the puppi MET, the only collection saved is the fully corrected MET. If user would like to "un-correct" puppi MET, they can use the following variables:

- puppiMETEGCor "corX" , "corY" and "corSumEt"
- puppiMETMuCor "corX" , "corY" and "corSumEt"

These variables are the X and Y components of the corrections.

MET Q&A:

- **Q**: I am looking at the re-miniaod data, what kind of MET should I be using?
- **A**: First, please note that we here propagate corrections provided by MUO and EGM POG. The most correct MET collection is slimmedMETsMuEGClean, this is the collection corrected by both e/gamma and muon effects. But what if you are seeing some tails or inconsistencies? It could be that:

- (1) Bad Muon Filters on the pf candidates have removed a good pf muon candidate -> new tails
- (2) Bad Muon Filters on the pf candidates didn t catch all the bad pf muon candidates -> not fully cleaned tails
- (3) EGamma GS Fix wasn t optimal, some sort of mis-match has happened -> new tails.

If case (1) has happened, you can go back to slimmedMETsEGClean + use the Bad Muon Filters to reject the events that were tagged (you loose efficiency for the mis-tagging part but you don t end up with tails)

If case (2) has happened, you should report the events. That means we need better filters for the legacy re-reco.

If case (3) has happened but the muon fix was behaving well and your analysis is dependent on high pt electrons and photons, you should re-run the recipe given on this twiki about the e/gamma corrected MET. We strongly encourage all analyzers to implement this recipe if they are sensitive to the gain switch issue.

- **Q**: Is it OK for me to use out of the box MET in data?
- **A**: If you are only interested in the type1 corrected MET and not sensitive to the gain switch issue,

then yes! In data, you can use out of the box MET. However, if you are interested in using the newly implemented JER uncertainties on MET, or updated MET Significance, or you are sensitive to the e/gamma gain switch issue, we recommend you follow the recipes given in this twiki.

- **Q**: What about MC? What do I do there?
- **A**: We did not have the e/gamma gain switch problem in the monte carlo, therefore any fix / recipe for this situation is not relevant. However, (although in much much smaller magnitude) one can have bad muons also in the simulation. If you want to check the effect of the filters or bad muon corrections on MET on your simulation, you can follow the recipe given under how do I correct mc on the fly for bad muons . This is not mandatory but is advised. You can think of this suggestion as our regular suggestions of checking the filter efficiencies on MC.

- **Q**: When should I JEC correct and when should I re-cluster MET?
- **A**: Are you doing something special with the pf candidates? If not, then re-correcting the met based on JECs is enough for you. The re-clustering jets/mets option in the recipe is for expert use, or if you want to change your pf candidate collection with some analysis specific candidates. For most analyzers re-correcting the MET for the latests JECs is enough.

How to re-correct MET based on bad muons on the fly for MC (Release: CMSSW_8_0_X, X>=26_patch1)

In this recipe, we are creating a new muon collection based on the tagged bad muons, and cleaning the pf candidates with respect to these bad muons. Using the cleaned pf candidates we will be re-clustering MET.

Full recipe is below Hide the recipe
In your configuration file you will need to add the following lines:

```
## Following lines are for default MET for Type1 corrections.
from PhysicsTools.PatUtils.tools.runMETCorrectionsAndUncertainties import runMetCorAndUncFromM

# If you only want to re-correct for JEC and get the proper uncertainties for the default MET
runMetCorAndUncFromMiniAOD(process,
                       isData=True (or False),
                      )

# Now you are creating the bad muon corrected MET
 process.load('RecoMET.METFilters.badGlobalMuonTaggersMiniAOD_cff')
 process.badGlobalMuonTaggerMAOD.taggingMode = cms.bool(True)
 process.cloneGlobalMuonTaggerMAOD.taggingMode = cms.bool(True)

 from PhysicsTools.PatUtils.tools.muonRecoMitigation import muonRecoMitigation

 muonRecoMitigation(
                   process = process,
                   pfCandCollection = "packedPFCandidates", #input PF Candidate Collection
                   runOnMiniAOD = True, #To determine if you are running on AOD or MiniAOD
                   selection="", #You can use a custom selection for your bad muons. Leave em
                   muonCollection="", #The muon collection name where your custom selection w
                   cleanCollName="cleanMuonsPFCandidates", #output pf candidate collection am
                   cleaningScheme="computeAllApplyClone", #Options are: "all", "computeAllApp
                   postfix="" #Use if you would like to add a post fix to your muon / pf coll
                   )

 runMetCorAndUncFromMiniAOD(process,
                       isData=runOnData,
                       pfCandColl="cleanMuonsPFCandidates",
                       recoMetFromPFCs=True,
                       postfix="MuClean"
                       )


 process.mucorMET = cms.Sequence(
```

```
    process.badGlobalMuonTaggerMAOD *
    process.cloneGlobalMuonTaggerMAOD *
  #process.badMuons * # If you are using cleaning mode "all", uncomment this line
    process.cleanMuonsPFCandidates *
    process.fullPatMetSequenceMuClean
    )
```

If you are running in the scheduled mode you will also need to add to your path:

```
process.p = cms.Path(
                process.mucorMET  *
                process.fullPatMetSequence * # If you are re-correctign the default MET
                process.yourAnalyzer
            )
```

This configuration will produce you an additional MET collection with the name you have specified as the post fix. In this particular example you will end up with:

```
vector                      "slimmedMETs"              ""                "PAT"  --> Out of the B
vector                      "slimmedMETs"              ""                "PROCESSNAME" --> Out o
vector                      "slimmedMETsMuClean"       ""                "PROCESSNAME" --> Reclu
```

How to re-correct MET based on e/gamma gain switch correction on the fly for Re-Miniaod Data (Release: CMSSW_8_0_X, X>=26_patch1)

For some portion of the events, the pf candidate referencing of the photons and electrons is not working. As a fall back solution, in the MET tool, a dR matching was implemented, however for a smaller subset of the events the dR matching was found to be too tight. If you want to recover this subset of events, one can follow the below recipe.

Full recipe is below Hide the recipe

```
git cms-merge-topic cms-met:METRecipe_80X_part2 -u
```

This merge will bring you the updated code for a looser dR matching condition. In your configuration file you will need to add the following lines:

```
## Following lines are for default MET for Type1 corrections.
from PhysicsTools.PatUtils.tools.runMETCorrectionsAndUncertainties import runMetCorAndUncFromM

# If you only want to re-correct for JEC and get the proper uncertainties for the default MET
runMetCorAndUncFromMiniAOD(process,
                       isData=True (or False),
                      )

# Now you are creating the e/g corrected MET on top of the bad muon corrected MET (on re-minia
from PhysicsTools.PatUtils.tools.corMETFromMuonAndEG import corMETFromMuonAndEG
corMETFromMuonAndEG(process,
                pfCandCollection="", #not needed
                electronCollection="slimmedElectronsBeforeGSFix",
                photonCollection="slimmedPhotonsBeforeGSFix",
                corElectronCollection="slimmedElectrons",
                corPhotonCollection="slimmedPhotons",
                allMETEGCorrected=True,
                muCorrection=False,
                eGCorrection=True,
                runOnMiniAOD=True,
                postfix="MuEGClean"
                )
  process.slimmedMETsMuEGClean = process.slimmedMETs.clone()
  process.slimmedMETsMuEGClean.src = cms.InputTag("patPFMetT1MuEGClean")
  process.slimmedMETsMuEGClean.rawVariation =  cms.InputTag("patPFMetRawMuEGClean")
  process.slimmedMETsMuEGClean.t1Uncertainties = cms.InputTag("patPFMetT1%sMuEGClean")
```

How to re-correct MET based on bad muons on the fly for MC (Release: CMSSW_8_0_X, X>=26_patch1)

```
del process.slimmedMETsMuEGClean.caloMET

# If you are running in the scheduled mode:
process.egcorrMET = cms.Sequence(
   process.cleanedPhotonsMuEGClean+process.cleanedCorPhotonsMuEGClean+
   process.matchedPhotonsMuEGClean + process.matchedElectronsMuEGClean +
   process.corMETPhotonMuEGClean+process.corMETElectronMuEGClean+
   process.patPFMetT1MuEGClean+process.patPFMetRawMuEGClean+
   process.patPFMetT1SmearMuEGClean+process.patPFMetT1TxyMuEGClean+
   process.patPFMetTxyMuEGClean+process.patPFMetT1JetEnUpMuEGClean+
   process.patPFMetT1JetResUpMuEGClean+process.patPFMetT1SmearJetResUpMuEGClean+
   process.patPFMetT1ElectronEnUpMuEGClean+process.patPFMetT1PhotonEnUpMuEGClean+
   process.patPFMetT1MuonEnUpMuEGClean+process.patPFMetT1TauEnUpMuEGClean+
   process.patPFMetT1UnclusteredEnUpMuEGClean+process.patPFMetT1JetEnDownMuEGClean+
   process.patPFMetT1JetResDownMuEGClean+process.patPFMetT1SmearJetResDownMuEGClean+
   process.patPFMetT1ElectronEnDownMuEGClean+process.patPFMetT1PhotonEnDownMuEGClean+
   process.patPFMetT1MuonEnDownMuEGClean+process.patPFMetT1TauEnDownMuEGClean+
   process.patPFMetT1UnclusteredEnDownMuEGClean+process.slimmedMETsMuEGClean)
```

If you are running in the scheduled mode you will also need to add to your path:

```
process.p = cms.Path(
                  process.fullPatMetSequence * # If you are re-correctign the default MET
                  process.egcorrMET  *
                  process.yourAnalyzer
            )
```

This configuration will produce you an additional MET collection with the name you have specified as the post fix. In this particular example you will end up with:

```
vector                      "slimmedMETs"               ""              "PAT"  --> Out of the
vector                      "slimmedMETsMuEGClean"      ""              "PAT"  --> Out of the
vector                      "slimmedMETsMuEGClean"      ""              "PROCESSNAME" --> Rec
```

This topic: CMSPublic > ReMiniAOD03Feb2017Notes
Topic revision: r19 - 2017-02-26 - RafaelLopesdeSa