Table of Contents

4.2.1 Physics Analysis Toolkit (PAT): Data Formats

Detailed Review status

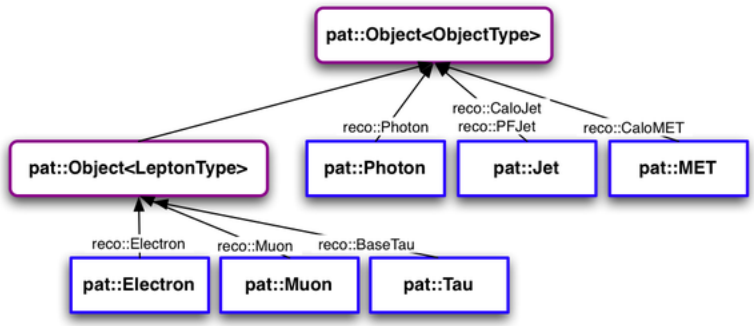
Contents

Introduction

In this section you will find a description of the main Data Formats of PAT (1) . The hierarchy of pat::Candidates is illustrated in the figure below. For each candidate you will find the following information:

- A common description of what information is available from the pat::Candidate (1) .
- A list of extra information that may be stored in the pat::Candidate with respect to the reco::Candidate.
- A list of all collections that you can expect to be present in a standard pat::Tuple (1) .
- A link into the most actual DoxyGen documentation providing access to all(!) available member functions of the corresponding pat::Candidate (1) .

(1) To learn more about regularly used expressions (like pat::Candidate, pat::Tuple, ...) have a look to the WorkBookPATGlossary.



The figure is *clickable*. By clicking on the collection of interest you will be lead to the corresponding description in the text.

All pat::Candidates are derived from their corresponding reco::Candidates. They are equivalent to a reco::Candidate but carry extra features and extra information to facilitate analysis work. The difference between a pat::Candidate and a reco::Candidate can be summarised in the following points:

- **More Information**: A pat::Candidate can carry more information than a reco::Candidate. This extra information spans from object resolutions, correction factors to the jet energy scale, parameterised reconstruction efficiencies and more.

- **Facilitated Access**: The access to all relevant information is facilitated via the pat::Candidate. You might be familiar the procedure to receive electron identification information via edm::Association maps or the b-tag information associated to a given reco::Jet. This is a non-trivial action and in addition very error prone. Via a pat::Candidate all this information is accessible via simple member functions. The used parameters, algorithms and procedures to arrive at this information represent the finest certified knowledge of the corresponding POG's.

- **Event Size Reduction**: The creation of a pat::Tuple supports you in reducing the enormous amount of information stored on RECO/AOD to the amount of information that you really need for your analysis. Sensible default configurations give you a very good starting point for first iterations. The event provenance is maintained. A pat::Tuple is well defined by the configuration file with which is was produced and a set of parameters. A pat::Tuple can very easily be exchanged between analysis groups for cross checks.

Note:

For a description how to access pat::Candidates via edm::Handles in the full framework or FWLite have a look at the WorkBookPATTutorial.

pat::Photon

Description:

Show Hide
Data type to describe reconstructed photons, which is derived from the reco::Photon. In addition to the reco::Photon it may contain the following extra information:

- isolation calculated in a pre-defined cone in different sub-detectors of CMS. You can choose between isolation as recommended by the corresponding POG or a user defined isolation accessible via the member function `userIsolation(pat::IsolatioKey key)`, where the available `IsolationKeys` are defined in Isolation.h. **Note:** user defined isolation needs to be configured during the pat::Candidate creation step as described in WorkBookPATConfiguration.
- isolation deposits for user defined isolation and more detailed studies
- match on generator level
- match on trigger level
- reconstruction efficiency
- object resolution
- Id variables
- shower shape variables

For a description of the algorithms to determine this extra information have a look at WorkBookPATWorkflow.

Access:

Show Hide
The labels of all photon collections, which are created during the standard PAT workflow(s) are given below:

Data Type	Collection Label	Instance Label	Process Label

pat::PhotonCollection	`patPhotons`	None	PAT
pat::PhotonCollection	`selectedPatPhotons`	None	PAT
pat::PhotonCollection	`cleanPatPhotons`	None	PAT

For a description of each listed module have a look at WorkBookPATConfiguration.

Member Functions:

You can find a list of all member functions at `pat::Photon`.

---

pat::Lepton

Description:

The pat::Electron, pat::Muon and pat::Tau inherit some common functionalities from the common base class pat::Lepton. Almost all of these features are related to isolation. The information held in these objects is:

- isolation calculated in a pre-defined cone in different sub-detectors of CMS. You can choose between isolation as recommended by the corresponding POG or a user defined isolation accessible via the member function `userIsolation(pat::IsolatioKey key)`, where the available `IsolationKeys` are defined in Isolation.h. **Note:** user defined isolation needs to be configured during the pat::Candidate creation step as described in WorkBookPATConfiguration#UserIsolation.
- isolation deposits in different regions of the detector. To study the details of isolation deposits see SWGuideIsoDeposits and the corresponding reference.

Member Functions:

You can find a list of all member functions at `pat::Lepton`.

pat::Electron

Description:

Show Hide
Data type to describe reconstructed electrons, which is derived from the reco::Electron. In addition to the reco::Electron it may contain the following extra information:

- isolation in a predefined cone
- isolation deposits for user defined isolation and more detailed studies
- match on generator level
- match on trigger level
- reconstruction efficiency
- object resolution
- Id variables
- shower shape variables

The electrons are sorted in descending order in p_t. For a description of the algorithms to determine this extra information have a look at WorkBookPATWorkflow.

Access:

Show Hide
The labels of all electron collections, which are created during the standard PAT workflow(s) are given below:

Data Type	Collection Label	Instance Label	Process Label
pat::ElectronCollection	`patElectrons`	None	PAT
pat::ElectronCollection	`selectedPatElectrons`	None	PAT
pat::ElectronCollection	`cleanPatElectrons`	None	PAT

For a description of each listed module have a look at WorkBookPATConfiguration.

Member Functions:

You can find a list of all member functions at `pat::Electron`.

---

pat::Muon

Description:

Show Hide
Data type to describe reconstructed muons, which is derived from the reco::Muon. In addition to the reco::Muon it may contain the following extra information:

- tracks from a TeV refit (`pat::Muon::pickyMuon()`, `pat::Muon::tpfmsMuon()`)
- isolation in a predefined cone
- isolation deposits for user defined isolation and more detailed studies
- match on generator level
- match on trigger level
- reconstruction efficiency
- object resolution
- two MuonMETCorrectionData objects. These objects allow to correct/uncorrect the caloMET and tcMET on the fly for the pat::Muon. They are obtained from the Muon-MET correction value maps that are described on WorkBookMetAnalysis#HeadingThree and WorkBookMetAnalysis#HeadingFour.

The muons are sorted in descending order in p_t. For a description of the algorithms to determine this extra information have a look at WorkBookPATWorkflow.

Access:

Show Hide
The labels of all muon collections, which are created during the standard PAT workflow(s) are given below:

Data Type	Collection Label	Instance Label	Process Label
pat::MuonCollection	`patMuons`	None	PAT
pat::MuonCollection	`selectedPatMuons`	None	PAT
pat::MuonCollection	`cleanPatMuons`	None	PAT

For a description of each listed module have a look at WorkBookPATConfiguration.

Member Functions:

You can find a list of all member functions at `pat::Muon`.

---

pat::Tau

Description:

Show Hide
Data type to describe reconstructed taus, which is derived from the reco::Tau. In addition to the reco::Tau it may contain the following extra information:

- isolation in a predefined cone
- match on generator level
- match on trigger level
- reconstruction efficiency
- object resolution
- tau discrimination variables

The taus are sorted in descending order in p_t. Per default particle flow taus are used, though this may be switched to calorimeter taus during the step of pat::Tuple creation. For a description of the algorithms to determine this extra information have a look at WorkBookPATWorkflow.

Access:

Show Hide
The labels of all tau collections, which are created during the standard PAT workflow(s) are given below:

Data Type	Collection Label	Instance Label	Process Label
pat::TauCollection	patTaus	None	PAT
pat::TauCollection	selectedPatTaus	None	PAT
pat::TauCollection	cleanPatTaus	None	PAT

For a description of each listed module have a look at WorkBookPATConfiguration.

Per default, the collections patTaus and selectedPatTaus correspond to particle flow taus, with no further selection applied.

Note: since reco::PFTaus are in one-to-one correspondence with reco::PFJets, one pat::Tau object will be added to the patTaus and selectedPatTaus collections for each particle flow jet. As a consequence, most pat::Taus contained in the patTau and selectedPatTau collections are actually fakes. The collection cleanPatTaus is the collection intended for analysis. It has the following selection applied in order to reduce the fake-rate:

- ≥ 1 PFChargedHadron of $p_T > 1.0$ GeV within matching cone of size dR = 0.1 around jet-axis (leading charged hadron exists)
- ≥ 1 PFGamma or PFChargedHadron of $p_T > 5.0$ GeV within matching cone of size dR = 0.1 around jet-axis (p_T of leading hadron greater than 5 GeV)
- no PFChargedHadron of $p_T > 1.0$ GeV and PFGammas of $p_T > 1.5$ GeV outside of signal cone of size dR = 5.0/jetE_T within jet (tau is isolated)
- discriminator against muons passed
- discriminator against electrons passed
- either 1 or 3 PFChargedHadrons within signal cone of size dR = 5.0/jetE_T (one or three prong tau)

The above selection criteria represent a compromise between tau id efficiency and fake-rate that is reasonable for many analyses. In case you find a non-negligible background of electrons remaining in the cleanPatTaus collection, you may want to cut the region

- 1.460 < letal < 1.558

between the ECAL barrel and endcap, in which the probability for electrons to get misidentified as taus is particularly high.

Member Functions:

You can find a list of all member functions at `pat::Tau`.

---

pat::Jet

Description:

Show Hide
Data type to describe reconstructed jets, which is derived from the reco::Jet. In addition to the reco::Jet it may contain the following extra information:

- b tag information
- jet energy corrections
- jet charge
- associated tracks
- jet flavour
- match on generator level (reco::genJet und parton)
- match on trigger level
- reconstruction efficiency
- object resolution

Per default anti-kT jets are used with R=0.5, though this may be switched to any other kind of jet collection during the step of pat::Tuple creation. Also alternative jet collections may be added to the standard pat::Tuple content depending on the analysis purposes. The energy scale of the pat::Jets is corrected to L3Absolute, while the correction factors for L2Relative, L5Flavour and L7Parton are also stored within the jet. They may be used to produce a clone of the jet, which is corrected to the specified level or fully uncorrected ('raw'). For more details on the arguments to be used have a look at the description of the pat::JetCorrFactors below. The jets are sorted in descending order in (L3Absolute calibrated) p_t. For a description of the algorithms to determine this extra information have a look at WorkBookPATWorkflow.

Access:

Show Hide
The labels of all jet collections, which are created during the standard PAT workflow(s) are given below:

Data Type	Collection Label (1)	Instance Label	Process Label
pat::JetCollection	`patJets`	None	PAT
pat::JetCollection	`selectedPatJets`	None	PAT
pat::JetCollection	`cleanPatJets`	None	PAT

(1) Note that when adding jet collections to the event content of the standard pat::Tuple the additional jet collections will be accompanied by a postfix identifying the additional jet collection. This postfix typically could be of type `cleanPatJetsAK5Calo`.

For a description of each listed module have a look at WorkBookPATConfiguration.

Member Functions:

You can find a list of all member functions at pat::Jet.

---

pat::MET

Description:

Show Hide
Data type to describe reconstructed MET, which is derived from the reco::MET. In addition to the reco::MET it may contain the following extra information:

- type 1 MET corrections
- muon MET corrections
- match on generator level
- match on trigger level
- reconstruction efficiency
- object resolution

Where the type 1 MET correction corresponds to the standard jet selection in use. Per default pat::MET is type 1 corrected and muon corrected. The muon corrections may be altered on the fly depending on the selection criteria for muons. For more details have a look at the description of the pat::Muon. For a description of the algorithms to determine this extra information have a look at WorkBookPATWorkflow.

Access:

Show Hide
The labels of the possible MET collections produced in a PAT workflow are given below:

Data Type	Collection Label	Instance Label	Process Label
pat::METCollection	patMETs	None	PAT
pat::METCollection	patMETsTC	None	PAT
pat::METCollection	patMETsPF	None	PAT

The patMETs collection is produced in the standard workflow, but the track corrected MET collection (patMETsTC) and the particle flow MET collection (labeled as patMETsPF) may be added to the content of the standard pat::Tuple on the user's choice (see patTuple_addJets_cfg.py). For a description of each listed module have a look at WorkBookPATConfiguration.

Member Functions:

You can find a list of all member functions at pat::MET.

---

pat::Trigger

For a complete description of all trigger related data formats within PAT have a look at SWGuidePATTrigger#Data_Formats.

Helper Classes

In this section you will find a description of the most important helper classes of PAT. Direct access to this classes might not be necessary though it can give you more insides into the mechanisms of the generation and access to the main data formats of PAT.

pat::JetCorrFactors

Description:

Show Hide
Data type to host all potential correction factors of the factorised calibration ansatz (L1 to L7) to be embedded in the pat::Jet. More then one set of jetCorrFactors can be embedded in one pat::Jet, as useful for the comparison of different sets of calibrations or systematic studies. The pat::JetCorrFactors class will be issued internally when creating clones of pat::Jets with different calibration levels. This will need the correction step and the potential flavour as arguments. The following correction levels (and flavours) are supported:

correctstep	flavour	correction type
`Uncorrected`	-	uncalibrated jet
`L1Offset`	-	offset correction
`L2Relative`	-	relative inter eta correction
`L3Absolute`	-	absolute p_t correction
`L23Residual`	-	residual corrections between data and MC after the L3Absolute correction
`L4Emf`	-	correction as a function of the jet emf

`L5Flavor`	`gluon`
	`uds`
	`charm`
	`bottom`
		hadron level correction for gluons, light quarks, charm, bottom

`L6UE`	`gluon`
	`uds`
	`charm`
	`bottom`
		underlying event correction for gluons, light quarks, charm, bottom

`L7Parton`	`glu`
	`uds`
	`charm`
	`bottom`
		parton level correction for gluons, light quarks, charm, bottom

Note: the flavour strings are case insensitive. Per default the set of jetCorrFactors is embedded in the pat::Jet, which corresponds to the JetMET default and the corrections for *L1Offset*, *L2Relative*, *L3Absolute*, *L5Flavor* and *L7Parton* are stored in addition. All other correction factors will be transparent. Per default the pat::Jet will be corrected to the following level (if available from the list of correction levels) with the following priorities:

- to *L2L3Residual* if available in the list of correction levels.
- to *L3Absolute* if available in the list of correction levels.
- to *Uncorrected* (i.e. no correction will be applied, if niether *L2L3Residual* nor *L3Absolute* are available in the list of corrction levels.

In the latter case, the *PatJetProducer* will issue a Warning during *patTuple* production. For a description of the algorithms to determine this extra information have a look at WorkBookPATWorkflow.

Access:

Show Hide
The labels of all jet collections, which are created during the standard PAT workflow(s) are given below:

Data Type	Collection Label	Instance Label	Process Label
pat::JetCorrFactors	patJetCorrFactors	None	PAT

For a description of the module have a look at WorkBookPATConfiguration.

Member Functions:

You can find a list of all member functions at pat::JetCorrFactors.

---

pat::CandKinResolution

Description:

Show Hide
Data type to host potential object resolutions. It can be embedded to each pat::Object. For a detailed description have a look at SWGuidePATResolutions.

Access:

Show Hide
Note that adding object resolutions and embedding them into a *pat::Candidiate* is OPTIONAL. They are not part of the default PAT event content. To add resolutions to an arbitary *pat::Candidate* add a service to your *cfg* file for *patTuple* production and switch the embedding of the resolution information for the corresponding *pat::Candidate* to `True`. An example for b jets (in one central bin in eta) is given below:

```
## define the service
bjetResolution = stringResolution.clone(parametrization = 'EtEtaPhi',
    functions = cms.VPSet(
      cms.PSet(
        bin = cms.string('0.000<=abs(eta) && abs(eta)<0.087'),
        et  = cms.string('et * (sqrt(0.0901^2 + (1.035/sqrt(et))^2 + (6.2/et)^2))'),
        eta = cms.string('sqrt(0.00516^2 + (1.683/et)^2)'),
        phi = cms.string('sqrt(0.0024^2 + (3.159/et)^2)'),
      ),
      # ... in reality you will have more bins in eta here of course ...
    ),
    constraints = cms.vdouble(0)
)

# embed the resolutions inot the patJets collection
patJets.addResolutions = True
patJets.resolutions    = bjetResolution
```

You can find an example *cff* file containing resolutions for muons, jets and MET as recently derived within the TopPAG based on MC truth information in a typical madgraph ttbar sample in stringResolutions_etEtaPhi_cff.py in the *python/recoLayer0* directory. Please note that this is **ONLY AN EXAMPLE** file to demonstrate the technique. It does not ease you from the burden to derive and test the resolutions which might be applicable and adequte for your analysis.

Member Functions:

You can find a list of all member functions at pat::CanKinResolution.

Review status

Show Hide

Reviewer/Editor and Date (copy from screen)	Comments
RogerWolf - 26 Aug 2010	Rework for upcoming PAT Sept Tutorial
XuanChen - 30 Jul 2014	Changed the links of files from cvs to github

Responsible: RogerWolf

This topic: CMSPublic > WorkBookPATDataFormats
Topic revision: r78 - 2014-07-30 - XuanChen