Table of Contents

3.2 Data Formats and Data Tiers....1
- Goals of this page:....1
- Contents....1
- Introduction....1
  - Reconstructing an Event....1
- About Data Formats....2
- About Data Tiers....2
- Data Tier Listing....2
- Data Tiers: Reconstructed (RECO) Data and Analysis Object Data (AOD)....3
  - RECO....3
  - AOD....4
  - Reference Documentation for RECO and AOD Data Format Packages....4
  - RECO Class and Method Naming Conventions....5
- Information Sources....5
- Review status....5

3.2 Data Formats and Data Tiers

Complete: ▭
Detailed Review status

Goals of this page:

This page is intended to familiarize you with the steps involved in event reconstruction and to introduce the collection of reconstructed data objects found in an event. In particular, you will learn:

- what a data format is
- what a data tier is
- where to find the supported data formats
- what the supported data tiers are
- what RECO and AOD event formats are, and why most user analyses use only these two
- which data formats are used in RECO and AOD data tiers

Contents

- Introduction
- About Data Formats
- About Data Tiers
- Data Tier Listing
- Data Tiers: Reconstructed (RECO) Data and Analysis Object Data (AOD)
    - RECO
    - AOD
    - References Documentation for RECO and AOD Data Format Packages
    - RECO Class and Method Naming Conventions
- Information Sources
- Review status

Introduction

Starting from raw data produced from the online system, successive degrees of processing (event reconstruction) refine this data, apply calibrations and create higher-level physics objects. CMS uses a number of data formats with varying degrees of detail, size, and refinement to write this data in its various stages. In turn, the data formats get grouped within an Event file into multiple Event formats, according to the data's origin or content.

Reconstructing an Event

An Event consists of the signals from all particles of an interaction, or possibly even several interactions, joined together. After sorting out which bits of information are related to the same particle (this process is called *pattern recognition*) the kinematical properties of each particle have to be reconstructed to reveal the physical nature of the whole event. This last process is called *reconstruction*.

About Data Formats

Each bit of data in an event must be written in a supported data format. A data format is essentially a C++ class, where a class defines a data structure (a data type with data members). The term *data format* can be used to refer to the format of the data written using the class (e.g., data format as a sort of template), or to the instantiated class object itself. The DataFormats package and the SimDataFormats package (for simulated data) in the CMSSW CVS repository contain all the supported data formats that can be written to an Event file. So, for example, if you wish to add data to an Event, your EDProducer module must instantiate one or more of these data format classes.

Data formats (classes) for reconstructed data, for example, include `Reco.Track`, `Reco.TrackExtra`, and many more. See the Offline Guide section SWGuideRecoDataTable for the full listing.

About Data Tiers

Event information from each step in the simulation and reconstruction chain is logically grouped into what we call a *data tier*. Examples of data tiers include `RAW` and `RECO`, and for MC, `GEN`, `SIM` and `DIGI`. A data tier may contain multiple data formats, as mentioned above for reconstructed data. A given dataset may consist of multiple data tiers, e.g., the term `GenSimDigi` includes the generation (MC), the simulation (Geant) and digitalization steps. The most important tiers from a physicist's point of view are probably `RECO` (all reconstructed objects and hits) and `AOD` (a smaller subset of `RECO`). The following table gives an overview.

E.g., the RAW data tier collects detector data after online formatting plus some trigger results, while the RECO tier collects reconstructed objects.

Data Tier Listing

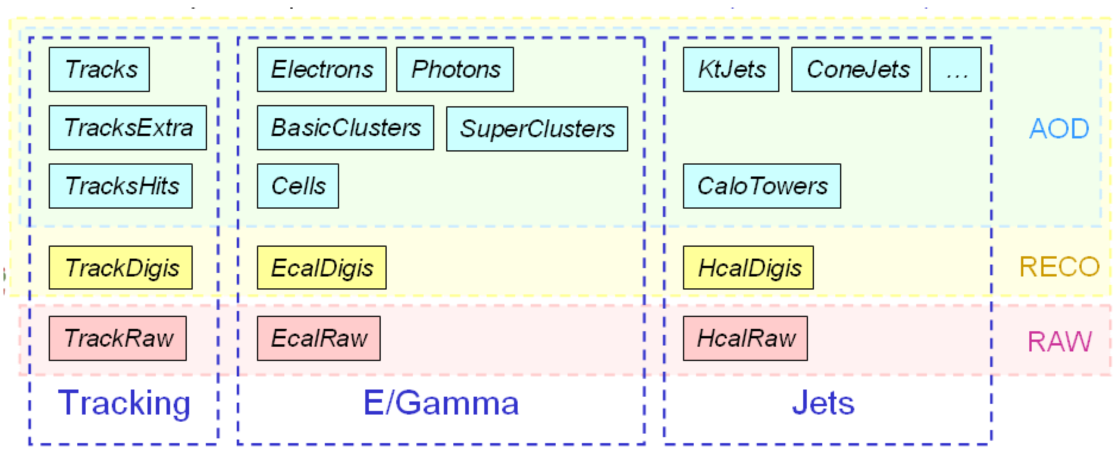
Event Format	Contents	Purpose	Data Type Ref	Event Size (MB)
DAQ-RAW	Detector data from front end electronics + L1 trigger result.	Primary record of physics event. Input to online HLT		1-1.5
RAW	Detector data after online formatting, the L1 trigger result, the result of the HLT selections (HLT trigger bits), potentially some of the higher level quantities calculated during HLT processing.	Input to Tier-0 reconstruction. Primary archive of events at CERN.		0.70-0.75
RECO	Reconstructed objects (tracks, vertices, jets, electrons, muons, etc.) and reconstructed hits/clusters	Output of Tier-0 reconstruction and subsequent rereconstruction passes. Supports re-finding of tracks, etc.	RECO & AOD	1.3-1.4
AOD	Subset of RECO. Reconstructed objects (tracks, vertices, jets, electrons, muons, etc.). Possible small quantities of very localised hit information.	Physics analysis, limited refitting of tracks and clusters	RECO & AOD	0.05
TAG	Run/event number, high-level physics objects, e.g. used to index events.	Rapid identification of events for further study (event directory).		0.01
FEVT		multiple		1.75

	Full Event: Term used to refer to RAW+RECO together (not a distinct format).			
GEN	Generated Monte Carlo event	-		-
SIM	Energy depositions of MC particles in detector (sim hits).	-		-
DIGI	Sim hits converted into detector response. Basically the same as the RAW output of the detector.	-		1.5

The *Data Type Ref* column entries point to the CMSSW Reference Manual, which is not complete.

Data Tiers: Reconstructed (RECO) Data and Analysis Object Data (AOD)

RECO data contains objects from all stages of reconstruction. AOD are derived from the RECO information to provide data for physics analyses in a convenient, compact format. Typically, physics analyses don't require you to rerun the reconstruction process on the data. Most physics analyses can run on AOD data.



RECO

RECO is the name of the data-tier which contains objects created by the event reconstruction program. It is derived from RAW data and provides access to reconstructed physics objects for physics analysis in a convenient format. Event reconstruction is structured in several hierarchical steps:

1. Detector-specific processing: Starting from detector data unpacking and decoding, detector calibration constants are applied and cluster or hit objects are reconstructed.
2. Tracking: Hits in the silicon and muon detectors are used to reconstruct global tracks. Pattern recognition in the tracker is the most CPU-intensive task.
3. Vertexing: Reconstructs primary and secondary vertex candidates.
4. Particle identification: Produces the objects most associated with physics analyses. Using a wide variety of sophisticated algorithms, standard physics object candidates are created (electrons, photons, muons, missing transverse energy and jets; heavy-quarks, tau decay).

The normal completion of the reconstruction task will result in a full set of these reconstructed objects usable by CMS physicists in their analyses. You would only need to rerun these algorithms if your analysis requires you to take account of such things as trial calibrations, novel algorithms etc.

Reconstruction is expensive in terms of CPU and is dominated by tracking. The RECO data-tier will provide compact information for analysis to avoid the necessity to access the RAW data for most analysis. Following the hierarchy of event reconstruction, RECO will contain objects from all stages of reconstruction. At the lowest level it will be reconstructed hits, clusters and segments. Based on these objects reconstructed tracks and vertices are stored. At the highest level reconstructed jets, muons, electrons, b-jets, etc. are stored. A direct reference from high-level objects to low-level objects will be possible, to avoid duplication of information. In addition the RECO format will preserve links to the RAW information.

The RECO data includes quantities required for typical analysis usage patterns such as: track re-finding, calorimeter reclustering, and jet energy calibration. The RECO event content is documented in the Offline Guide at RECO Data Format Table.

AOD

AOD are derived from the RECO information to provide data for physics analysis in a convenient, compact format. AOD data are usable directly by physics analyses. AOD data will be produced by the same, or subsequent, processing steps as produce the RECO data; and AOD data will be made easily available at multiple sites to CMS members. The AOD will contain enough information about the event to support all the typical usage patterns of a physics analysis. Thus, it will contain a copy of all the high-level physics objects (such as muons, electrons, taus, etc.), plus a summary of the RECO information sufficient to support typical analysis actions such as track refitting with improved alignment or kinematic constraints, re-evaluation of energy and/or position of ECAL clusters based on analysis-specific corrections. The AOD, because of the limited size that will not allow it to contain all the hits, will typically not support the application of novel pattern recognition techniques, nor the application of new calibration constants, which would typically require the use of RECO or RAW information.

The AOD data tier will contain physics objects: tracks with associated Hits, calorimetric clusters with associated Hits, vertices, jets and high-level physics objects (electrons, muons, Z boson candidates, and so on).

Because the AOD data tier is relatively compact, all Tier-1 computing centres are able to keep a full copy of the AOD, while they will hold only a subset of the RAW and RECO data tiers. The AOD event content is documented in the Offline Guide at AOD Data Format Table.

Reference Documentation for RECO and AOD Data Format Packages

The reference documentation is provided in the Offline Guide sections:

- RECO Data Formats
- AOD Data Formats

These links provide a list of all packages related to the RECO and AOD data formats within the CMSSW repository. Links there point to the class documentation of each data object. A short instructions how to access the object are given.

More about the different data inside an event can be found in the "ADVANCED TOPICS" part of the WorkBook where the objects and their creation are explained in detail. In the "ESSENTIALS" part, the section about Particle Candidates gives an overview of the candidate model.

RECO Class and Method Naming Conventions

The RECO classes are designed to provide a simple and uniform interface. For instance, the same method-naming convention is used to acccess the same quantities throughout the various classes:

- everywhere: pt(), eta(), phi(), energy()
- not inconsistent conventions: E(), Energy(), getE(), getEnergy(), etc.

For example, let's look at the RECO data format `TrackReco`. In the CMSSW Cross-Reference (LXR), I search for file name `TrackReco`. We get lots of returned file names:

```
Calibration/HcalIsolatedTrackReco/data/isolPixelTrackProdSeq.cff --
Calibration/HcalIsolatedTrackReco/data/isolPixelTrackProd.cfi --
Calibration/HcalIsolatedTrackReco/doc/html/overview.html --
Calibration/HcalIsolatedTrackReco/doc/html/index.html --
Calibration/HcalIsolatedTrackReco/interface/IsolatedPixelTrackCandidateProducer.h --
Calibration/HcalIsolatedTrackReco/src/SealModule.cc --
Calibration/HcalIsolatedTrackReco/src/IsolatedPixelTrackCandidateProducer.cc --
Calibration/HcalIsolatedTrackReco/CMS.BuildFile --
DataFormats/EgammaTrackReco/doc/html/overview.html --
DataFormats/EgammaTrackReco/doc/html/index.html --
DataFormats/EgammaTrackReco/interface/TrackCandidateSuperClusterAssociation.h --

...

DataFormats/TrackReco/interface/Track.h --
DataFormats/TrackReco/interface/TrackBase.h --
DataFormats/TrackReco/interface/TrackExtra.h --

...
```

Clicking on `/CMSSW/src/DataFormats/TrackReco/interface/Track.h`, we find several functions (methods) defined with clear function names, e.g.:

```
...
    /// x coordinate of momentum vector at the outermost hit position
    double outerPx()     const { return extra_->outerPx(); }
    /// y coordinate of momentum vector at the outermost hit position
    double outerPy()     const { return extra_->outerPy(); }
    /// z coordinate of momentum vector at the outermost hit position
    double outerPz()     const { return extra_->outerPz(); }
    /// x coordinate of the outermost hit position
...
```

Another way to learn about the interface of the data inside an event is described in FWLite in Python.

Information Sources

Computing TDR, section 2.5.

Oliver Gutsche's "Make an Analysis" tutorial from May 06.

Review status

Reviewer/Editor and Date (copy from screen)	**Comments**

JennyWilliams - 24 Oct 2007	review and minor editing
BenediktHegner - 20 Dec 2006	Changes for the print version.
AnneHeavey - 09 Jun 2006	Edit after review by Kati, some questions answered by Luca; talk with Oliver G (and take some content from his "make an analysis")

Responsible: BenediktHegner
Last reviewed by: PetarMaksimovic - 15 Feb 2008

This topic: CMSPublic > WorkBookDataFormats
Topic revision: r31 - 2016-06-27 - SudhirMalik