Muon Event Data Model & Dataflow Overview

Edward Moyse
• Raw ‘bytestream’ from the detector is fed into convertors, which then produce object representations of this data, known as ‘Raw Data Objects’ (RDOs).

• Next the ‘RDOs’ are converted into ‘Prepared Raw Data’ (PRDs). For both these steps the Cabling services are needed, to convert the online identifiers to their offline equivalents. Additionally the RDO to PRD step makes use of calibration services (for example to provide the drift radius of the MDT drift tubes, and the detector description (to assign the data to the correct detector elements). Clustering is also done at this stage.

• Finally, the reconstruction turns this raw data into tracks, segments, ntuples etc, for which it uses a plethora of services and common tools (including of course the magnetic field)
Documentation

• Before we start, this will just be an overview - for more in depth info, look here:

• Tracking EDM is documented with ATLAS note: [http://cdsweb.cern.ch/search?sysno=002696474CER](http://cdsweb.cern.ch/search?sysno=002696474CER)
  
  • Explains a lot of design principles, and defines meaning of track parameters etc.

• Recommended way of browsing Muon EDM classes is to use Doxygen
  
  
  • ... and type the name of the class, package (e.g. MuonPrepRawData), or method you want to find out about.

• Twiki pages:
  
  • [https://twiki.cern.ch/twiki/bin/view/Atlas/MuonEventDataModel](https://twiki.cern.ch/twiki/bin/view/Atlas/MuonEventDataModel)
  
  • [https://twiki.cern.ch/twiki/bin/view/Atlas/CommonTrackingSoftware](https://twiki.cern.ch/twiki/bin/view/Atlas/CommonTrackingSoftware)
RDO

- Detailed, detector specific data, primarily for experts. e.g.

- CscRawData:
  - `rpulD()` : Sparsifier Processing Unit - identifier in which this strip is.
  - `identify`() : online identifier of the collection containing this strip
  - `width`() : number of consecutive strips forming the on-line cluster
  - `isTimeComputed`() : false, if failed to compute the time in the ROD
  - `samples`() : vector of amps
  - `address()` : online identifier of the first strip in the ROD cluster
  - `hashId()` : offline hash identifier of the first strip in the online cluster

PRD

- Starts to get more generic.
- Inherit from Trk::PrepRawData, so all PRDs provide the following information:
  - `identify()`: the offline identifier of the detecting element (strip, drift tube ...) which made this measurement
  - `localPosition()`: the position on the detector element. For MDTs this is 1-d and is the drift radius, for CSCs it's the position of the CSC wire in plane of the module etc
  - `rdoList()`: the list of offline identifiers of the strips which made this 'cluster' i.e. for MDTs it's just the same as `identify()` but for CSCs it could be several adjacent wires.
  - `localErrorMatrix()`: the error matrix corresponding to this measurement (1-d for MDTs!)
  - `detectorElement()`: the detector element which took this measurement.

- Further extensions for specific technologies:
  - `MdtPrepData`: tdc, adc + status flag
    - Status flag is e.g. whether MDT masked
  - `CscPrepData`: charge, time + status flag
  - `RpcPrepData`: time + trigger info (to be removed soon, and put in separate class)
Segments

- MuonSegments are an intermediate stage of track finding, but are also useful for validation.
- They are (by convention) confined to one station layer (i.e. inner, middle or outer)

_MuonSegment:_

- `globalPosition()` : returns the global position of the segment (there is also `globalDirection()`)
- `associatedSurface()` : returns the Trk::Surface used to define this segment (can be a detector element, but is more usually an ‘free’ surface, owned by the segment)
- `localDirection()` : the direction wrt the surface returned by the above
- `containedROTs()` : returns the vector of Trk::RIO_OnTrack objects (i.e. the measurements used to make this segment)
- `localParameters()` & `localErrorMatrix`: returns the ‘parameters’ (i.e. position) wrt to segment surface, and the associated errors.
- `fitQuality()` : returns the Trk::FitQuality (typically chi-squared per degrees of freedom) of the segment fit
- `author()` : returns the algorithm used to create this surface
Tracks

- Tracks are necessarily complex & flexible objects, since they are used in a wide variety of different ways.

- Fundamentally `Trk::Tracks` consist of a ‘fit quality’, author, track parameters, measurements, material affects.

- In practice they consist of an arbitrary number of `TrackStatesOnSurface` (TSOS)

- Each TSOS can have any/all of:
  - `TrackParameters`
  - `Measurements`
  - `FitQuality`
  - `Material interactions`

Parameters at various points through the detector

(MDT) measurements

Material interaction

Pixel measurement
Tracks (2)

- A track may also provide a ‘TrackSummary’
- This object returns e.g. number of holes per track, number of hits etc etc
- Note that it is not guaranteed to exist on the track, nor is everything it can contain be guaranteed to be there i.e. it will not have information on holes, if a hole search was not run
- More information: link
Track Parameters

- As with other Tracking EDM classes there is a common base class (TrackParameters) which provides e.g.
  - Local position (i.e. position on the associated surface)
  - Momentum
  - Global position (in tracking frame)
  - Charge
  - A pointer to the associated surface.

- There are then specific concrete implementation for each of the 5 surface types. These return track parameters in the relevant frames

- Surfaces:
  - Cylinder, Disc, Plane, StraightLine, Perigee

- The parameters can be "bare" parameters (e.g. Perigee), or they can be from an extrapolated measurement and therefore possess an error matrix as well (e.g. MeasuredPerigee).
Measurements

• Measurements can be e.g. Segments, calibrated measurements (RIO_OnTracks) etc, all of which can exist on a track

• All measurements inherit from Trk::MeasurementBase, and provide:
  • LocalParameters (i.e. the local position), the associated error, the surface on which they were measured,

• Segments were covered before, but RIO_OnTracks (more calibrated PRDs e.g. with wire-sag corrections) provide:
  • a pointer to the PRD used to create it, the detector element the measurement took place on, the Identifier of the detecting element used to make the measurement

• Some Muon RIO_OnTracks extend this still further:
  • MdtDriftCircleOnTrack: the side on which the drift radius is wrt to the track, the status of the drift radius calibration, the drift time, whether sagged line corrections were used.
  • CscClusterOnTrack: status (e.g. Unspoilt, Simple, etc ) Link
Summary

• This has been a very brief skim through the Muon EDM and dataflow.

• The tracking EDM is necessarily quite complex - to understand more (for example about the track particle) I encourage you to read the ATLAS note and to browse Doxygen

• Read the documentation we have provided on the Twiki

• Look at code which uses the EDM

• And of course, ask the tracking hypernews forum:
  • atlas-sw-tracking@cern.ch

• Good luck!