ND Software Data Model

ND Software Integration Meeting

July 30, 2020

Andy Mastbaum
Rutgers University
mastbaum@physics.rutgers.edu

Data Model

Overview/Reminder

- **Goal:** Define a robust data model for simulation & analysis
  - Ensure consistency/compatibility as ND SW development continues
  - Ensure consistency/compatibility with FD where possible
  - Support joint analyses, e.g. LAr + GAr matching, ND + FD
- **Data Model**
  - "Data Model" mainly refers to information stored ("data products")
  - Also considering "external" data that will live in databases, etc.
  - Related question of the file format implementation (ROOT `TTrees`, HDF5, etc.) is left flexible
- **Status**
  - A first iteration, defining and documenting the expected data
  - Built from input from LAr, ND-GAr, and SAND groups (thank you!)
  - A *"living document"* — Updates expected as definitions evolve with continued detector & software development

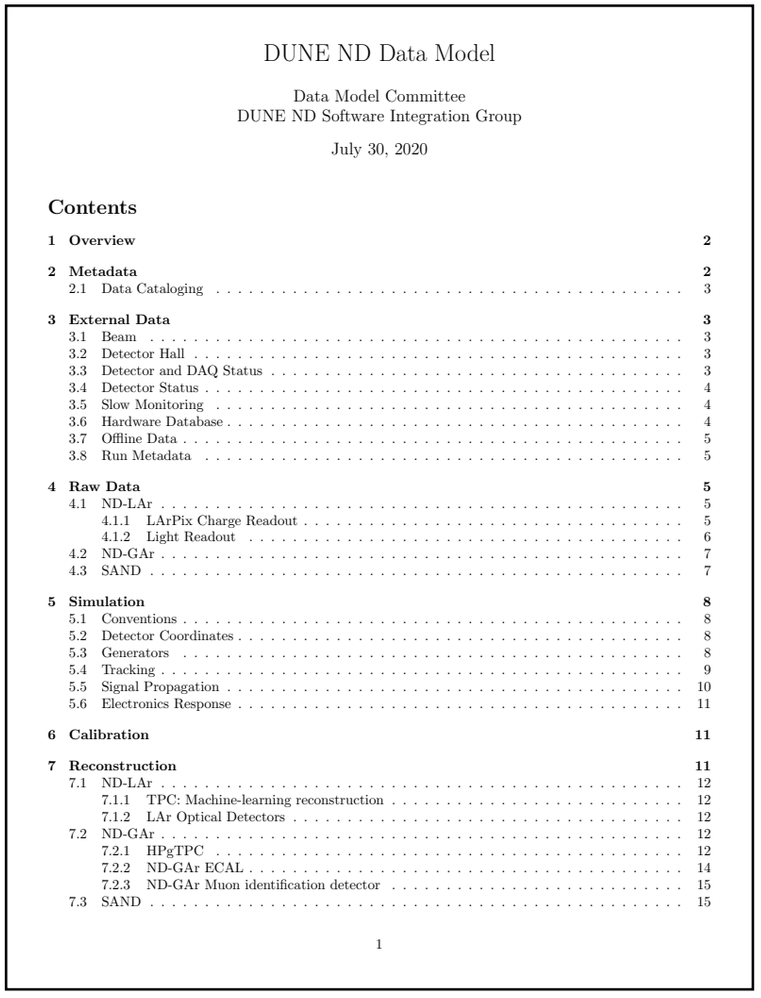
View link: https://www.overleaf.com/read/ypvkqbfmpmtx

Data Model Document

Contents

Data Model Document

Discussion



DUNE ND Data Model

Data Model Committee
DUNE ND Software Integration Group

July 30, 2020

Contents

1

View link: https://www.overleaf.com/read/ypvkqbfmpmtx

- Thank you to everyone for the contributions & input, and Mathew and Roberto for helping expand the narrative text.
- Currently the draft *includes*
  - Brief summaries of detector components and processing stages
  - Data formats for each data/simulation stage for each detector, as they are currently envisioned, with common definitions for similar data
  - Discussion of external data and the metadata needed to access it
- Discussion points
  - Looking forward to feedback from ND groups!
  - Scope — Is everything included that should be included?
  - Accuracy & completeness — up to date with current plans?

DUNE ND Data Model

Data Model Committee
DUNE ND Software Integration Group

July 30, 2020

Contents

1 Overview

This document describes the data model for the DUNE ND simulation and analysis software. This encompasses all ND components, using common data formats across the ND wherever possible to enable compatibility of analysis software across the ND detector groups. The main goal is to centralize the model definition, and promote consistency across the ND detectors and their subdetectors, as well as the DUNE Far Detector, wherever possible. This extends to the data definitions ("data products" in LArSoft terms), the relationships between objects such as those supporting truth-matching, system of units, and coordinate conventions. This consistency will ease the sharing of analysis code and algorithms, joint analysis tasks such as multi-detector matching and FD/ND studies, and reduce the learning curve for developers.

This model is intended as a "living document" that will necessarily evolve as ND development continues. It is meant to be updated so that it can serve as a common reference for data content and facilitate using common definitions for similar data across detectors — for example, a shared definition of a hit in a pixel-readout LArTPC and GArTPC.

The *Data Model* here refers to information stored, without explicit reference to files or formats, i.e. ROOT TTrees, HDF5, etc. Rather, we define a common set of variables such that ND simulation and analysis software may develop independently and employ a variety of formats that change with new technologies, while remaining compatible via a thin translation layer.

The document begins with a discussion of data is expected to be stored external to the data (e.g. in databases) (Section 3) and the metadata required to index and access that data (Section 2). Next, each step in the data processing and simulation is covered, beginning with the raw data formats in Section 4. This is followed by simulation data, broken down by stages (Section 5). Post-calibration data is described in Section 6. Finally, the reconstruction stage is covered in Section 7.

2 Metadata

Metadata refers to information included in the offline files to provide additional information about the context or conditions for the data/simulation events. Depending on the specific needs, this may be on the file, run, subrun, or event level. We consider two types of metadata: (1) information stored directly in the file alongside the events, and (2) references to external data such as a conditions database.

The following metadata fields are expected to be stored in files:

Geometry ID *[file]* — Sufficient information to uniquely identify the detector geometry used in the simulation and/or analysis. Rather than storing a full representation of the GDML, files will store a name and version identifier, referring to a version-controlled GDML geometry.

Cluster Information *[file]* — Basic information about the computer system that produced the file, including at minimum the date, hostname, OS version, and compiler version.

Software Information *[file]* — Information about the software packages run to produce the file, including at minimum the names and version numbers or Git SHA IDs.

Conditions Database Index *[subrun]* — A tuple of run and subrun IDs corresponding to keys in an external database of run conditions. This is meant to enable access to a broad range of ND and detector-specific run condition details, which are enumerated in Section 3. — this is event-level. run/subrun or time, and DB table version ID.

2.1 Data Cataloging

Due to the huge numbers of expected data files associated with the near detector a file cataloging and handling system will be required to define multi-file datasets, archive deprecated data, provide data locations, and transfer request files to areas for analysis on-demand. An example of such a system is the Sequential Access via Metadata (SAM) system at Fermilab, which is being used to catalog the existing DUNE datasets. Metadata describing the files and allowing dataset definitions must be provided to the cataloging system. As defined by the DUNE Data Management Group cataloging metadata includes:

Provenance information Record of files used to produce the file

Data format Reference to conditions in which the file was produced (e.g. run number, trigger type, creation time, software version)

Dataset Relationship between files

Information about file contents This may include event numbers, run number, and other identifiers required to adequately distinguish files.

3 External Data

External data refers to information stored outside of the files containing event data that provides necessary input or context for simulation and analysis. It is crucial that for any event, one can find the required corresponding external data with either the event data (timestamps, i.e.) or provided *metadata* as described in Section 2.

It is anticipated that the required external data will be stored across multiple external databases; for example, a beam conditions database with horn current polarity and a channel database with a list of disabled readout channels and calibration values. Unique identifying keys or validity identifiers must stored in the event file to reference the needed data in each database appropriately. External data used for analysis/simulation should be version controlled with version information available to record for future reference. Below is a list of data which is expected to be accessed as external. This list is meant to be illustrative and will evolve as analysis use cases are further developed.

3.1 Beam

Conditions related to the beam include include:

Horn current polarity – Whether the beam is operating in forward (FHC or "neutrino") or reverse (RHC or "antineutrino") mode. This is expected to vary only on long time scales relative to a run, and may be indexed by run ID.

Target mode – Whether the beam is operating in an 'on-target' or 'off-target' mode.

3.2 Detector Hall

Conditions related to the detector hall include:

Detector Positions *[time-based]* – With the the DUNE-PRISM capability, the ND-LAr and ND-GAr will be movable several meters transverse to the beam direction. We require the *positions* and *position uncertainties* of the detectors to be monitored and recorded.

3.3 Detector and DAQ Status

Conditions related to the detectors include:

Online status – Whether the detector is online or offline (e.g. undergoing maintenance)

Channel Conditions – Configuration for each detector channel

- Channel status – Whether the channel is enabled
- Channel settings – Parameters used to configure the channel

Magnet Conditions – Conditions for the magnets (ND-GAr and SAND)

- Magnet status – Whether the magnet is enabled
- Magnet settings – Parameters used to configure the magnet

DAQ Conditions – State of the DAQ including software and hardware status.

3.4 Detector Status

Conditions related to the detectors include:

Online status – Whether the detector is online or offline (e.g. undergoing maintenance)

Channel Conditions – Configuration for each detector channel

- Channel status – Whether the channel is enabled
- Channel settings – Parameters used to configure the channel

Magnet Conditions – Conditions for the magnets (ND-GAr and SAND)

- Magnet status – Whether the magnet is enabled
- Magnet settings – Parameters used to configure the magnet

3.5 Slow Monitoring

Slow monitoring encompasses the measured conditions of detector hardware systems, usually read as a continuous stream of time-stamped measurements. Since some conditions may be directly useful in simulation and analysis, software tools should be able to query this database and files/events must have sufficient metadata. Examples of Slow Monitoring data include, but are not limited to:

Cryogenic and Gas Systems — Fluid temperature, pressures, flow rates, circulation status, filter status

Low-Voltage Power — Front-end electronics power supply status, voltage and currents

High-Voltage Systems — Drift high voltage, anode plane wire bias status, photon detectors, etc., voltage, and current measurements at multiple readout points

DAQ and Computer Status — Status and performance metrics for the DAQ software and hardware

3.6 Hardware Database

A record of hardware installation history across the near detector suite.

Hardware serial number — Identifying information of hardware.

Hardware installation location — Where was this hardware installed.

Time Validity range – When was this hardware installed at the given location.

3.7 Offline Data

It is expected that certain external data used in analysis will be determined using the offline software, stored in a database, and then applied in subsequent analysis stages. Such information includes:

Calibrations — Calibration information for each component of the ND complex, such as constants used to convert raw measurements into physical units. This includes, for example:

- Channel data — Gain factors and baseline offsets for each channel
- High-level parameters — Measured model parameters for high-level data/MC corrections
- Calibration constants — Factors to convert e.g. raw slow monitoring data into physical units

Field Maps — Electric & magnetic field maps. This must provide sufficient information for Geant4 simulation and event reconstruction with an accurate field model.

- Electric field — A measured electric field map for TPCs, and/or parameters for a model
- Magnetic field — A measured magnetic field map for each magnet, and/or model parameters

Data Quality — Derived metrics for assessing run/subrun and channel quality for analysis .

3.8 Run Metadata

Externally-managed data associated with the (sub-)run may include:

- Run ID
- Subrun ID
- Start/stop time
- Start/stop trigger IDs
- Indices into relevant conditions databases

4 Raw Data

This section describes the formats of the raw data, defined as the set of data fields available in the final output from the detector data acquisition (DAQ) system, which would be passed to the "offline" software for calibration, reconstruction, and analysis.

4.1 ND-LAr

The ND-LAr includes mutltiple detector subsystems producing raw data output, including the LArPix charge readout and the ArCLight and LCM optical detectors. With an ongoing prototype R&D program, the raw data formats are continuing to evolve but already quite well-defined.

4.1.1 LArPix Charge Readout

The LArPix raw data is persisted in self-documenting HDF5-format files. The format is detailed in the `larpix-control` documentation[1], from which this information is drawn.

The raw data present in LArPix readout packets includes:

- **io_group** *u1/unsigned byte* — an id associated with the high-level io group associated with this packet
- **io_channel** *u1/unsigned byte* — the id associated with the mid-level io channel associated with this packet

[1] `https://larpix-control.readthedocs.io/en/stable/api/format/hdf5format.html`

- **packet_type** *u1/unsigned byte* — the packet type code, which can be interpreted according to the map stored in the `packets` attribute `packet_types`
- **chip_id** *u1/unsigned byte* — the LArPix chip id
- **parity** *u1/unsigned byte* — the packet parity bit (0 or 1)
- **valid_parity** *u1/unsigned byte* — 1 if the packet parity is valid (odd), 0 if it is invalid
- **downstream_marker** *u1/unsigned byte* — a marker to indicate the hydra io network direction for this packet
- **channel_id** *u1/unsigned byte* — the ASIC channel
- **timestamp** *u8/unsigned 8-byte long int* — the timestamp associated with the packet. Caution: this field does "many-duty" as both the ASIC timestamp in data packets (type 0), as the global timestamp in timestamp packets (type 4), as the message timestamp in message packets (type 5), as the timestamp in sync packets (type 6), and the timestamp in trigger packets (type 7).
- **first_packet** *u1/unsigned byte* indicates if this is the packet received in a trigger burst (v2.1 or newer only)
- **dataword** *u1/unsigned byte* — the ADC data word Caution: as of v2.2, this field does double duty as both the LArPix ADC value in data packets (type 0) and the clk source value in sync packets (type 6).
- **trigger_type** *u1/unsigned byte* — the trigger type associated with this packet. Caution: as of v2.2, this field does triple duty as both the LArPix packet trigger type in data packets (type 0), the sync type in sync packets (type 6), and the trigger type in trigger packets (type 7).
- **local_fifo** *u1/unsigned byte* — 1 if the channel FIFO is >50% full, 3 if the channel FIFO is 100% full
- **shared_fifo** *u1/unsigned byte* — 1 if the chip FIFO is >50% full, 3 if the channel FIFO is 100% full
- **register_address** *u1/unsigned byte* — the configuration register index
- **register_data** *u1/unsigned byte* — the configuration register value
- **direction** *u1/unsigned byte* — 0 if packet was sent to ASICs, 1 if packet was received from ASICs.
- **local_fifo_events** *u1/unsigned byte* — number of packets in the channel FIFO (only valid if FIFO diagnostics are enabled)
- **shared_fifo_events** *u2/unsigned byte* — number of packets in the chip FIFO (only valid if FIFO diagnostics are enabled)
- **counter** *u4/unsigned 4-byte int* — the message index (only valid for message type packets)
- **fifo_diagnostics_enabled** *u1/unsigned byte* — flag for when fifo diagnostics are enabled (1 if enabled, 0 if not)

4.1.2 Light Readout

ArCLight and LCM data formats will include:

- **type** *enum OpDetType* — Detector type (ArCLight/LCM)
- **channelID** *unsigned long* — Channel ID number
- **adc** *unsigned* — Digitized charge
- **timestamp** *unsigned long* — Trigger timestamp

4.2 ND-GAr

The ND-GAr reference design consists of an HPgTPC chamber, a pressure vessel, an electromagnetic calorimeter (ECAL), a magnet system (Helmholtz-coils or solenoid) and a muon ID detector. The different raw data types are listed below.

- HPgTPC — The design of the TPC in the ND-GAr is based on the ALICE readout chamber design with a full radius of 2.6 m and a length of 5 m. It consists of 18 chambers. The inner part (up to a radius of 83 cm), not covered by the ALICE chambers, is filled with a readout pixel plane (LArPIX-style). Waveform snippets or hits or LArPIX-style charge integral times. Data format is self-describing. Compression algorithm and version, waveform lengths, pedestal and RMS as needed. DAQ format is not yet specified, but we will need to convert it into the offline format when it becomes known. Most data will be zero-suppressed, but we also want continuous waveform digitization for diagnostic purposes.
- ECAL — The ECAL in the ND-GAr is a copper-scintillator sampling calorimeter. The barrel is octagonal and is around the HPgTPC and Pressure Vessel, inside the magnet (Helmholtz or SPY solenoid). Two endcaps are in addition to provide hermiticity. The barrel consists of 5 modules in the B field direction, trapezoidal in cross-section and are 1.4 m long and 44 cm thick with bases of 2.04 m and 2.93 m. In total each module fits 60 layers, the first 8 layers consists of tiles of 2.5x2.5 cm^2 readout by SiPMs, the last 52 layers consists of cross-strips spanning the full module length readout on both sides by SiPMs. The endcap design is not yet finalized. They would consists of 60 layers with the first 6 with tiles and the last 54 with cross-strips. The front end electronics would be integrated on the layer directly. The data would be zero-suppressed. No continuous waveform from the SiPM, only digitized data from the front end electronics. DAQ format is not yet specified but would be close to the CALICE AHCAL format. Data should be self describing. Several running mode may be necessary (beam, calibration (SiPM gain monitoring, temperature), pedestal).
- Muon identification detector — May be similar as the ECAL or could be MINERvA-like; design not finalized.
- Photon detection — May be included depending on future R&D. This would provide additional T0 to match ECAL energy deposition and tracks.

4.3 SAND

The SAND reference design consists of a magnet system, an electromagnetic calorimeter (ECAL), the 3D projection scintillator tracker (3DST) and a low-density tracker system, which could be a time projection chamber (TPC) or a straw tube tracker (STT). Potentially, there are multiple raw data types. They are listed below.

- ECAL The ECAL in SAND is a lead-scintillating fiber sampling calorimeter. The barrel calorimeter is cylindrical and is located inside the KLOE magnet, close to the coil cryostat. Two additional calorimeters (endcaps) ensure hermeticity along the magnet endcaps. The barrel consists of 24 modules, each of which is 4.3 m long, 23 cm thick and trapezoidal in cross-section, with bases of 52 and 59 cm. Each end-cap consists of 32 vertical modules that are 0.7–3.9 m long and 23 cm thick. For each module, The read-out subdivides the calorimeter into five planes in depth. The first four planes are 4.4 cm deep and the last plane is 5.2 cm deep. Each plane is subdivided in the transverse direction into 4.4 cm wide elements, except at the edges of the trapezoidal modules.The light guides matching the module end-faces to the photo-tube windows begin with a mixing section and terminate with a Winston cone providing an area concentration factor of about 4.
- 3DST The 3DST consists of a large number of cubes (2400 x 2400 x 200 as the reference design). Each cube has three fibers passing through along three directions. The number of readout channel/fiber is $240 \times 240 + 240 \times 200 + 240 \times 200 = 145600$. Each fiber connects to a MPPC at one end and the signal goes through the front end board consisting ASIC board, ADC and FPGA, and being transferred to the PC through the DAQ system. Each of the ASIC chip can read out 32 channels and each front end board can host three ASIC chips in the beam test performed at CERN and LANL. The raw data will be in the binary format in the DAQ machine and should be quickly transferred to some remote servers.

- TPC The design of the TPCs for SAND is based on three rectangular chambers in the downstream and upper/lower side of the 3DST, filling the low-density gas tracker volume. The readout planes are instrumented with bulk Micromegas modules, providing a signal-over-noise ratio of 100, with pixeled readout anode with pads of 7×10 mm for the vertical-TPCs. The Micromegas modules of the ND280 vertical-TPCs are rectangular with 34×36 cm2 size, hosting 1728 pad each. Each 15 module is read by 6 Front-End Card (FEC)s, each of which is instrumented with 4 AFTER ASICs. The data from each module are further processed by a Front-End Mezzanine (FEM) card which sends them through optical fibers to Data-Concentrator cards (DCC) placed outside the magnet (2 for each read-out plane). A similar architecture can be envisaged for SAND: in the baseline hypothesis of 70000 channels, this would correspond to a production of about 1000 ASICs, 250 FECs, 40 FEM and about 10 DCC.

- STT The base tracker technology is provided by low-mass straws (5 mm diameter, 12 μm walls, 20 μm wire Au plated W, operated with Xe/CO2 70/30 gas at 1.9 atm) similar to the ones used in many modern experiments for precision physics or the search for rare processes. The front-end electronics readout is based upon the VMM3a custom Application Specific Integrated Circuit (ASIC). Each ASIC chip offers the readout for 64 individual straws, allowing the use of compact FE electronic boards fully integrated within the frames of the STT modules.

5 Simulation

5.1 Conventions

System of Units For consistency with the Far Detector simulation, the ND data model universally adopts the same units as LArSoft:

Energy — GeV

Distance — centimeters, cm

Time — nanoseconds, ns

Related quantities are defined accordingly, for example momenta in GeV/c. In case of ambiguity, the LArSoft conventions must be followed. We note that in some cases, the LArSoft units are not well documented, for example in class members containing a momentum value. Wherever code implementations of this data model exist, the units should always be explicitly defined in a comment.

We define one important exception to this rule: in the case of data models defined externally to this document, the rules of that model take precedence. For example, files containing a copy of a GENIE event record should not modify the units (such as meters) native to that object, as this will enable downstream use in e.g. GENIE reweighting tools. Such cases should, however, be explicitly documented wherever applied.

5.2 Detector Coordinates

Following the convention chosen for the far detector coordinate axes, the y axis points vertically upwards, opposite to the local gravity direction at the near site. The z axis points approximately along the beam direction, and perpendicular to gravity. The x axis is also horizontal, and points approximately South. The origin of coordinates is along the beam axis on the upstream wall (where the beam enters the ND Hall). When individual detectors' geometries are used in isolation, they are free to choose the origin of coordinates that is most convenient.

5.3 Generators

At the vertex generation level, the ND data model must support both GENIE and other event generator formats, including minimal four-vector input. While GENIE is treated as a default, many analyses including long-baseline oscillations and BSM searches may require alternative generators.

Additionally, the generator-level information stored for ND events must support translation into the corresponding *art* data products, `simb::MCTruth`, `simb::MCFlux`, and `simb::GTruth`. To the extent that

generic generator output can be cast into the GENIE `GHepRecord` format, tools within `larsim` exist for mapping to/from *art* objects. In the case that only four-vectors are provided as input, these may be trivially converted to `sim::MCParticle`s in a `simb::MCTruth` object.

Generator-level information includes:

- **eventID** *struct EventID* — Event ID (run, subrun, spill)
- Truth interaction ID
- Generator truth — Replicas of LArSoft's `simb::GTruth` and `simb::MCTruth`, which contain GENIE truth information and a minimal subset required for GENIE event reweighting.
- Flux truth — Replicas of LArSoft's `bsim::Dk2Nu` and/or `simb::MCTruth`, depending on the input flux.
- Spill record — Spill-level beam measurements, including intensity and on/off target

The *Event ID* here is used throughout this document, and is defined as a tuple uniquely identifying a beam spill:

- **run** *unsigned* — Run ID number
- **subrun** *unsigned* — Subun ID number
- **spill** *unsigned* — Spill ID number

5.4 Tracking

The next stage in the simulation is final-state particle tracking, i.e. the Geant4-level simulation of particle tracks through the detector geometry. This stage is nominally performed using the `edep-sim` package, and we adopt that data structure here. A module to convert the `edep-sim` format into the corresponding *art* data products has been written by E. Brianne and is included as part of GArSoft.

Data at the tracking stage includes:

- **eventID** *struct EventID* — Event ID (run, subrun, spill)
- **GeometryID** *struct GeoID* — Geometry information
  - **name** *string* — Geometry filename or string identifier
  - **version** *string* — Version control tag or hash ID
- **Primaries** *TG4PrimaryVertexContainer* — Primary vertex container, per `edep-sim`
- **Trajectories** *TG4TrajectoryContainer* — Trajectory container, per `edep-sim`
  - **Position** *TLorentzVector* — The position of the trajectory point
  - **Momentum** *TVector3* — The momentum as the trajectory leaves the point.
  - **Process** *unsigned long* — The process type which created this point. These are defined by Geant4.
  - **Subprocess** *unsigned long* — The process type which created this point. These are defined by Geant4.
- **SegmentDetectors** *TG4HitSegmentDetectors* — Container with volume-by-volume collections of truth-level energy depositions (Geant4 track segments), per `edep-sim`
  - **Contrib** *unsigned long [N]* — A vector of track identifiers which contributed to this hit segment. There is almost always only one contributor, but for certain run settings there may be several particles associated with one segment (this is a very unusual situation). This has type `TG4HitSegment::Contributors` and is equivalent to a vector of integers.
  - **PrimaryId** *unsigned long* — The track identifier for the primary particle creating this hit.

- **EnergyDeposit** *float* — The total energy deposited over the length of this track. The energy should be assumed to have been uniformly deposited along the segment (not at the beginning or the end). This is the total dE/dx of the track between the start and stop position of the segment.
- **SecondaryDeposit** *float* — The "secondary" energy deposited over the length of the segment. This is generally used to help simulate the energy emitted as scintillation light vs the total dE/dx of the track. In other words, when the secondary deposit simulation is turned on, This is the energy deposited as optical photons. The remaining energy is usually deposited as ionization. For example, in liquid argon, the mean number of quanta created will be $N_q = (E_{dep}/W_{\mathrm{Ar}})$, where W_{Ar} is the work function for argon (typically 19.5 eV). The number of optical photons is $N_\gamma = N_q \times E_{secondDep}/E_{dep}$, and the number of ionization electrons is $N_e = N_q - N_\gamma$.
- **TrackLength** *float* — The total track length between the start and stop points (as estimated by Geant4).
- **Start** *TLorentzVector* — The starting point of the segment.
- **Stop** *TLorentzVector* — The stopping point of the segment.

5.5 Signal Propagation

Detector response simulation, mapping true G4 energy depositions into e.g. true simulated hits in detector subsystems. Per Tom, include some mechanism like LArSoft for grouping subsets of hits, e.g. for reducing need to store non-primary MCParticles esp. for showers.

- **eventID** *struct EventID* — Event ID (run, subrun, spill)
- **hitCollection** *struct* — Charge hit collection (TPC and ECAL hits)
  - **channelID** *unsigned long* — Channel or cell ID number
  - **trackID** *unsigned long* — Geant4 track ID
  - **trackStepID** *unsigned long* — Geant4 track step ID. Should refer to an edep-sim hit segment, which may be multiple steps, for now.
  - **energy** *float* — Energy loss in hit
  - **time** *float* — Time of hit
  - **start** *float[3]* — Spatial start coordinates
  - **stop** *float[3]* — Spatial stop coordinates
  - **nElectrons** *unsigned long* — Number of electrons
  - **landauFactor** *float* (optional) — Landau fluctuations applied
  - **recombinationFactor** *float* (optional)— Recombination factor applied
  - **diffusionFactorL** *float* (optional) — Longitudinal diffusion applied (D_L)
  - **diffusionFactorT** *float* (optional) — Transverse diffusion applied (D_T)
- **opHitCollection** *struct* — Optical hit collection
  - **type** *enum OpDetType* — Detector type (ArCLight/LCM)
  - **channelID** *unsigned long* — Channel or cell ID number
  - **trackID** *unsigned long* — Geant4 track ID
  - **trackStepID** *unsigned long* — Geant4 track step ID. Should refer to an edep-sim hit segment, which may be multiple steps, for now.
  - **nScintillationPhotons** *float* — Number of scintillation photons
  - **nCherenkovPhotons** *float* — Number of Cherenkov photons
  - **time** *float* — Time of optical hit

5.6 Electronics Response

Simulation of the electronics response to true signal inputs, resulting in an output format equivalent to data, and which can be processed using the same downstream calibration and analysis tools, but imbued with additional truth-matching capabilities. **Note:** In SAND, Sections 5.5 and 5.6 are typically combined into a single step.

- Event ID (run, subrun, spill)
- LArPix charge data *Raw data fields listed in Section 4.1.1*
  - **truthHits** *unsigned[N]* — List of contributing truth charge hit IDs
- Calo data
  - **cellID** *unsigned* — Cell ID
  - **adc** *unsigned* — Measured ADC charge
  - **time** *float[2]* — Hit time measurement
  - **position** *float[3]* — Position
- Optical data *Raw data fields listed in Section 4.1.2*
  - **truthHits** *unsigned[N]* — List of contributing truth optical hit IDs

6 Calibration

The Calibration stage converts the raw data format, coming either from data (Section 4) or simulation (Section 5.6), to physical units while preserving available truth-matching information. The formats are by nature similar to the Signal Propagation information (Section 5.5), with true charges before the application of the electronics response.

- **eventID** *struct EventID* — Event ID (run, subrun, spill)
- **calHitCollection** *struct* — Charge hit collection (TPC and ECAL hits)
  - **channelID** *unsigned long* — Channel or cell ID number
  - **energy** *float* — Energy loss in hit
  - **time** *float* — Time of hit
  - **position** *float[3]* (ECAL) — Position
  - **truthHits** *unsigned[N]* (MC only) — List of contributing truth hit IDs
- **calOpHitCollection** *struct* — Optical hit collection
  - **type** *enum OpDetType* — Detector type (ArCLight/LCM)
  - **channelID** *unsigned long* — Channel or cell ID number
  - **nphotons** *float* — Number of photons
  - **time** *float* — Time of optical hit
  - **truthHits** *unsigned[N]* (MC only) — List of contributing truth optical hit IDs

7 Reconstruction

The Reconstruction stage refers to the analysis of (calibrated) event data/simulation to create the objects used in downstream event selection and analysis. This includes several degrees of closeness to the "truth" information, from track or cluster formation (reconstructing the Geant4 truth) to particle identification and particle flow hierarchy (reconstructing the generator truth).

7.1 ND-LAr

7.1.1 TPC: Machine-learning reconstruction

- **meta** *struct* — Image meta data
  - **eventID** *struct EventID* — Event ID (run, subrun, spill)
  - **boundingBox** *float32[6]* — Volume boundary XYZ min/max
  - **pixelSize** *float32[3]* — Pixel size
  - **version** *unsigned* — Reco software version
- **instanceID** *float32[N]* — Pixel-level particle instance ID for each pixel
- **interactions** *struct[n_interactions]* — Interaction information
  - **id** *int16* — Interaction ID
  - **type** *int8* — Interaction type
  - **vertex** *float32[3]* — Interaction vertex position
  - **particles** *int16[n_particles]* — List of particle instances for each particle
- **particles** *struct[n_particles]* — Particle information
  - **id** *int16* — Particle ID
  - **interactionID** *int16* — Interaction ID
  - **interactionScore** *float32[n_interactions]* — Interaction ID score
  - **type** *int32* — Particle type
  - **typeScore** *float32* — Particle type score
  - **semanticType** *int32* — Particle semantic type
  - **start** *float32[6]* — Particle start position and direction
  - **end** *float32[3]* — Particle end position and direction
  - **momentum** *float32[2]* — Particle momentum and energy
  - **parentParticleID** *int16* — Parent particle ID
  - **childrenParticleIDs** *int16[n_particles]* — List of children particle ID
  - **rankID** *int16* — Rank ID in particle tree

7.1.2 LAr Optical Detectors

- **eventID** *struct EventID* — Event ID (run, subrun, spill)
- *To be determined*

7.2 ND-GAr

7.2.1 HPgTPC

All reconstruction data products in the HPgTPC have a unique integer ID number inside them in order to assist with identification and testing for equality. If a data product's data has been copied out of the *art* event store, we still want to test for equality of data and not of memory location.

- **hits** *Hit* — TPC Hit
  - **fChannel** *unsigned* — DAQ channel – hits are localized to one readout pad

  - **fPosition** *float[3]* — Position (relative to the readout window start. Time is not yet known as tracks have to be associated to the ECAL or cross the cathode or be linked to another track that does to get the time)
  - **fSignal** *float* — Charge (ADC integral)
  - **fStartTime** *float* — Start Time
  - **fEndTime** *float* — End Time
  - **fTime** *float* — Best-fit or average time
  - **fRMS** *float* — Hit width (RMS) in 1D

- **clusters** *TPCCluster* — TPC Clusters
  - **fIDnumero** *gar::rec::IDNumber* — DAQ channel
  - **fPosition** *float[3]* — Position (relative to the readout window start. Time is not yet known as tracks have to be associated to the ECAL or cross the cathode or be linked to another track that does to get the time)
  - **fCovMat** *float[6]* — A compressed representation of a covariance matrix indicating the spatial extent of the charge contributing to this cluster.
  - **fStartTime** *float* — Start Time
  - **fEndTime** *float* — End Time
  - **fTime** *float* — Best-fit or average time
  - **fRMS** *float* — RMS in 1D
  - *Associations to contributing hits*

- **vectorHits** *Vector Hit* — Vectors hits, an intermediate step in pattern recognition.
  - **fPosition** *float[3]* — Position (relative to the readout window start. Time is not yet known as tracks have to be associated to the ECAL or cross the cathode or be linked to another track that does to get the time)
  - **fDirection** *float[3]* — Direction
  - **fLength** *float* — Length, centered on **fPosition**
  - **fNclusters** *unsigned* — Number of contributing TPC Clusters
  - *Associations to contributing TPC Clusters.*

- **vertices** *Vertex* — TPC vertex
  - **fIDnumero** *gar::rec::IDNumber* — DAQ channel
  - **fPosition** *float[3]* — Position
  - **fCovMat** *float[3][3]* — Position fit uncertainty covariance matrix
  - **fTime** *double* — Timestamp

- **vees** *Vee* — A K/Λ decay-like "vee" vertex
  - **fIDnumero** *gar::rec::IDNumber* — DAQ channel
  - **fPosition** *float[3]* — Position
  - **fCovMat** *float[3][3]* — Position fit uncertainty covariance matrix
  - **fTime** *double* — Timestamp
  - **fChisq** *float* — Fit χ^2
  - **fFourMomentum** float[4][3] — Vee four-momentum for three hypotheses (one K^0_s and two Λ^0 particle-assignment hypotheses)

- **tracks** *Track* — TPC Track
  - **fLengthforwards** *float* — Length from forward fit
  - **fLengthbackwards** *float* — Length from backwards fit
  - **fChisqForward** *float* — χ^2 from forward fit
  - **fChisqBackward** *float* — χ^2 from backwards fit
  - **fMomentum_beg** *float* — Momentum at end
  - **fMomentum_end** *float* — Momentum at end
  - **fVertex** *float[3]* — Start position
  - **fEnd** *float[3]* — End position
  - **fVtxDir** *float[3]* — Start direction
  - **fEndDir** *float[3]* — End direction
  - **fNHits** *size_t* — Number of hits
  - **fTime** *double* — Trigger time
  - **fTrackParBeg** *float[5]* — Start fit parameters
  - **fTrackParEnd** *float[5]* — End fit parameters
  - **fCovMatBeg** *float[15]* — Start fit covariance matrix
  - **fCovMatEnd** *float[15]* — End fit covariance matrix
- **trackioniz** *TrackIoniz* — Track ionization data
  - **fFWD_dSigdXs** *std::vector<std::pair<float, float>>* — Ionization dE/dx values along track.
  - **fBAK_dSigdXs** *std::vector<std::pair<float, float>>* — Ionization dE/dx values along track.
  - *Association to corresponding track.*
- **trackTrajectories** *TrackTrajectory* – Fitted trajectory points along a track. To be used with the event display. Some tracks deviate significantly from helices. Even though helical track parameters work at any point along a track, energy loss and scattering require a fuller description of the trajectory
  - **fFWDTrajectory** *float[3]* — Segment values ordered by forward fit
  - **fBAKTrajectory** float[3] — Segment values ordered by backward fit
  - *Association to corresponding track.*
- **particleID** — Particle ID, not yet designed. Discriminant output scores from particle ID algorithms using input from the TPC, ECAL, and muon catcher. We may need a separate data product for each kind of particle we want to detect: "Muon", "Electron", "Proton", "Conversion" are examples of objects that depend on information from several subdetectors and will require links to the contributing hits, clusters, tracks, and stubs.

7.2.2 ND-GAr ECAL

- **caloHits** *CaloHit* — ECAL Hit
  - **fIDnumero** *gar::rec::IDNumber* — DAQ channel
  - **fEnergy** *float* — Energy [GeV]
  - **fPosition** *float[3]* — Position
  - **fTime** *std::pair<float,float>* — Hit time [ns]
  - **fCellID** *raw::CellID* — ECAL cell ID
  - **fLayer** *unsigned* — ECAL layer

- **ecalCluster** *ECAL_Cluster* — ECAL clusters
  - **fEnergy** *float* — Energy [GeV]
  - **fTime** *float* — Time [ns]
  - **fTimeDiffFirstLast** *float* — Time difference
  - **fPosition** *float[3]* — Position of cluster [cm]
  - **fShape** *float[6]* — Shape parameters (6 floats)
  - **fTheta** *float* — Theta
  - **fPhi** *float* — Phi
  - **fEigenVector** *float[9]* — PCA eigenvectors (9 floats)
  - **fParticleID** *int* — Particle ID flag
  - **fTracks** *std::vector<Track>* — Associated tracks
  - **fHits** *std::vector<CaloHit>* — Hit contributions
  - **fWeights** *std::vector<float>* — Weights
- **ecalPFParticle** *ECAL_PFParticle* — ECAL Reconstructed Pandora Object
  - **fType** *int* — Type of PFParticle
  - **fEnergy** *float* — Energy [GeV]
  - **fPos** *float[3]* — Position [cm]
  - **fMom** *float[3]* — 3-Momentum components [GeV]
  - **fMass** *float* — Mass [GeV]
  - **fCharge** *float* — Charge
  - **fPdg** *int* — PID of the PFParticle
  - **fGoodness** *float* — Goodness of the PID
  - **fParent** *size_t* — Parent ID of the PFParticle
  - **fDaughters** *std::vector<size_t>* — Daughters ID of the PFParticle

7.2.3 ND-GAr Muon identification detector

Likely like the ECAL.

- **caloHits** *CaloHit*

7.3 SAND

7.3.1 3DST

- Pre-selection
- View matching/voxelization
- Charge assignment
- Clustering
- Vertex finding
- Pattern recognition
- PID

7.3.2 TPC

7.3.3 ECAL

7.3.4 STT