REST-for-Physics Framework

10/05/2022

Luis Antonio Obis Aparicio

(lobis@unizar.es)

Centro de Astropartículas y Física de Altas Energías

Universidad Zaragoza

About myself

- **Physicist**, **PhD student** at University of Zaragoza (**CAPA**)
- Member of the **IAXO** collaboration
- Working on rare event searches (axions), specifically on simulation of background, detector response and analysis with **REST**
- Developer and user of the REST-for-Physics framework (**REST**)

Overview

- REST-for-Physics framework overview
- GitHub repositories and documentation links
- REST core library
- Versioning strategy
- REST libraries
- restG4 simulation package
- REST python interface

The REST-for-Physics framework

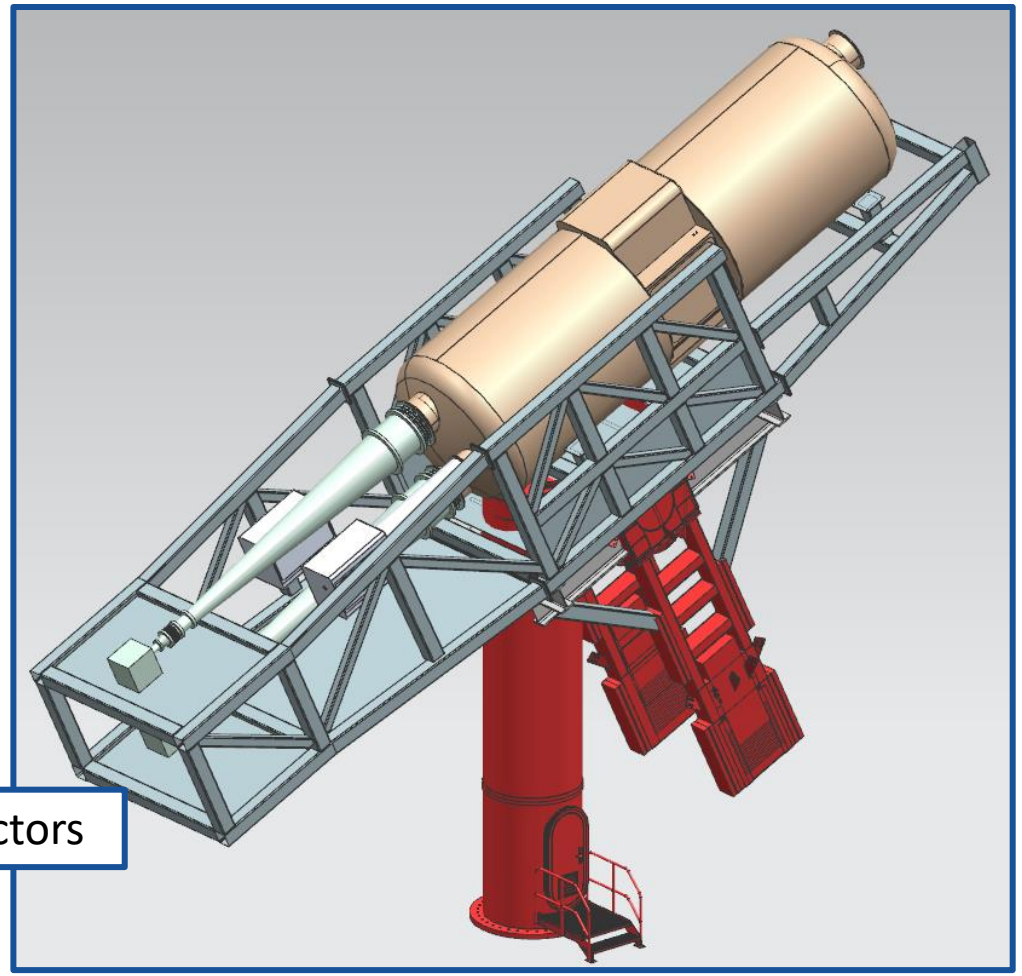
- REST (Rare Event Searches Toolkit) is a **ROOT** based event-oriented data processing and analysis framework
- Created in University of Zaragoza from an effort to unify our experimental and analysis activities in a common environment. First “established” version from (~2014-2015)
- Used and contributed to by users from different institutions such as University of Zaragoza, University of Barcelona (Spain), University of Shanghai (China) or CEA Saclay (France)
- Made for physicists by physicists, in an academic environment, **not by software engineers** (disclaimer!)
- Composed by different modular libraries and packages
- Can be used for simulation (Geant4) of background and signal, detector response, analysis…
- Unified event format for experimental and MC data, (mostly) same analysis chain
- REST provides ready to use examples. This is especially useful for (undergraduate) students

The REST-for-Physics framework

- REST official publication: https://doi.org/10.1016/j.cpc.2021.108281 (arxiv)
- Initially developed for rare event searches with TPC (time projection chamber), but not limited to this!
- Used in multiple experiments:
  - PandaX-III: 0νββ of 136Xe using high pressure gaseous TPC
  - TrexDM: Low mass WIMPs using high pressure gaseous TPC (Micromegas readout)
  - IAXO: Proposed solar axion detection platform
- Used extensively in research and teaching for undergraduate / graduate theses

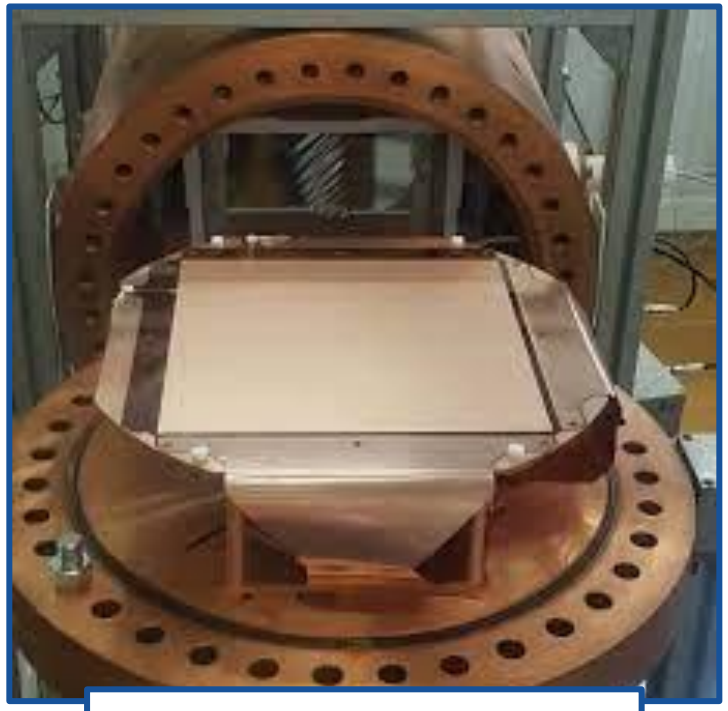
The REST-for-Physics framework

BabyIAXO CAD model



Detectors

TREX-DM Micromegas pixelated readout and time projection chamber (TPC)



Canfranc Underground Laboratory (Spain)

The REST-for-Physics framework

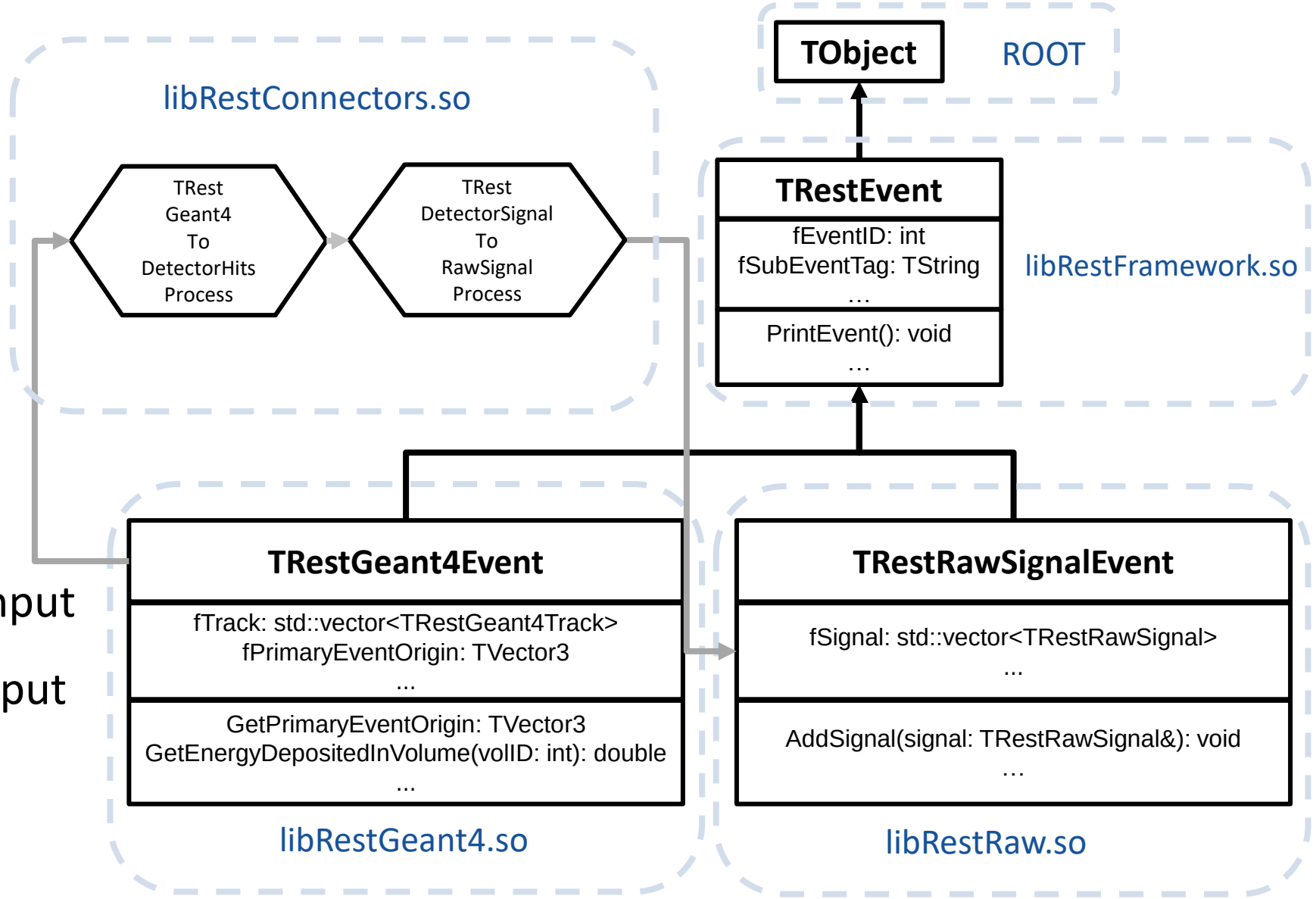
- REST official forum: https://rest-forum.unizar.es/
- GitHub page: https://github.com/rest-for-physics
- People: https://github.com/orgs/rest-for-physics/people
- Documentation (WIP): https://rest-for-physics.github.io/
- Feel free to write an **issue** / **PR** in the core repository or in any library / package repository! **Always looking for new contributors!**

REST Project Structure

- **GitHub: https://github.com/rest-for-physics**
- **Core**: Metadata, Run, Analysis Tree, etc. classes
- Libraries:
  - **Geant4Lib**: Produce REST compatible data in Geant4 simulations (does not need Geant4)
  - **DetectorLib**: Detector response, readouts…
  - **RawLib**: Raw signal (electronics) analysis
  - **TrackLib**: Algorithms related to tracks
  - **ConnectorsLib**: Communication between different libraries
- Packages:
  - **restG4**: Produce REST structured simulated data using the Geant4 software

The REST-for-Physics framework

- **REST** is **event oriented**.
- Many kinds of events:
  - ***TRestGeant4Event***
  - ***TRestDetectorSignalEvent***
  - ***TRestRawSignalEvent***
  - ***…***
- ***Event Processes*** take an input event and produce an output event



REST Core Library

- *libRestFramework.so*: https://github.com/rest-for-physics/framework
- Most important classes:
  - ***TRestRun***: stores run information (1 run = 1 root file), handles I/O (via ROOT), provides convenient access to some objects
  - ***TRestMetadata***: serialization/deserialization (XML), handles user configurations and persistence. Every user configurable object inherits from this class
  - **TRestEvent**: run data is stored in a **TTree** with different types of **TRestEvent** as branches
  - **TRestEventProcess**: base for all analysis processes: input event -> output event
  - **TRestAnalysisTree**: TTree derived class, hosts event level observables produced by different processes, which can later be used in cuts…

Structure of a REST file

EventTree (TTree) for event storage

TRestRun object for run management

```
root [0] .ls
TFile**
 TFile*
  OBJ: TTree    EventTree        TRestTrackEventTree : 0 at: 0x55cfeccd0f80
  OBJ: TRestAnalysisTree         AnalysisTree    REST Process Analysis Tree : 0 at: 0x55cfecf8f500
  KEY: TRestRun IAXOD0-2021;2    IAXOD0 2021 data taking
  KEY: TRestProcessRunner        Signals;1       Signal to track analysis
  KEY: TRestDetectorSignalToHitsProcess signalToHits;1  A Signal To Hits reconstruction template.
  KEY: TRestDetectorHitsAnalysisProcess hitsAna;1       Hits analysis template
  KEY: TRestDetectorHitsGaussAnalysisProcess    hitsAnaGauss;1  defaultTitle
  KEY: TRestDetectorHitsToTrackProcess  hitsToTrack;1
  KEY: TRestTrackAnalysisProcess        tckAna;1        Track analysis template
  KEY: TTree    EventTree;1      TRestTrackEventTree
  KEY: TRestAnalysisTree         AnalysisTree;1  REST Process Analysis Tree
  KEY: TRestProcessRunner        RawSignals;1    Raw processing and analysis
  KEY: TRestRawMultiFEMINOSToSignalProcess       virtualDAQ;1    defaultTitle
  KEY: TRestRawVetoAnalysisProcess      veto;1  defaultTitle
  KEY: TRestRawSignalAnalysisProcess    sAna;1
  KEY: TRestRawZeroSuppresionProcess    zS;1
  KEY: TRestDetectorSignalChannelActivityProcess        chActivity;1    Channel activity process
  KEY: TRestDetectorReadout      iaxo_readout;2  IAXO-D0 readout 0.5 mm-Pitch 120+120 channels
root [1]
```

filename

AnalysisTree (TRestAnalysisTree) for analysis observables

Analysis process metadata for traceability

TRestMetadata

- **TRestMetadata** class implements serialization via XML files (custom .rml extension). Most classes inherit from **TRestMetadata**
- Example file: https://github.com/rest-for-physics/framework/blob/0153c04a702bfd37c78e8a02f2844ac9443c5413/examples/g4ToHits.rml

TRestRun user configuration for analisis process

```
<TRestRun name="Run metadata" title="REST Metadata run info
(template)" verboseLevel="info">
    <parameter name="experimentName" value="Test"/>
    <parameter name="runType" value="simulation"/>
    <parameter name="runNumber" value="-1"/>
    <parameter name="runTag" value="preserve"/>
    <parameter name="runDescription" value="We launch just 100
events to check everything is ok."/>
    <parameter name="user" value="${USER}"/>
    <parameter name="verboseLevel" value="2"/>
    <parameter name="outputFile"
value="Run[fRunNumber]_[fRunTag]_g4ToHits.root" />
</TRestRun>
```

Analysis processes with configurable parameters

```
<TRestProcessRunner name="TemplateEventProcess"
verboseLevel="info">
    <parameter name="eventsToProcess" value="0" />
    <parameter name="threadNumber" value="2"/>

    <parameter name="inputAnalysisStorage" value="on"/>
    <parameter name="inputEventStorage" value="on"/>
    <parameter name="outputEventStorage" value="on"/>

    <addProcess type="TRestGeant4ToDetectorHitsProcess"
name="g4ToHits" value="ON">
    </addProcess>

    <addProcess type="TRestDetectorHitsAnalysisProcess"
name="hitsAna" value="ON">
        <parameter name="observable" value="all" />
    </addProcess>

</TRestProcessRunner>
```

Process name

Process parameters

TRestAnalysisTree

- Every REST file has an analysis tree (inherits from TTree)
- All observables defined in the analysis RML file appear here

```
*.....................................................................*
*Br    5 :subEventTag : TString                                       *
*Entries :     4297 : Total  Size=      22190 bytes  File Size  =     *
*Baskets :        2 : Basket Size=      32000 bytes  Compression=  3.4*
*.....................................................................*
*Br    6 :veto_PeakTime : map<int,double> (map<int,double>)           *
*Entries :     4297 : Total  Size=     511148 bytes  File Size  =     165198 *
*Baskets :       17 : Basket Size=      32000 bytes  Compression=   3.09     *
*.....................................................................*
*Br    7 :veto_MaxPeakAmplitude : map<int,double> (map<int,double>)   *
*Entries :     4297 : Total  Size=     511308 bytes  File Size  =     224702 *
*Baskets :       17 : Basket Size=      32000 bytes  Compression=   2.27     *
*.....................................................................*
*Br    8 :veto_VetoAboveThreshold : Int_t (int)                       *
*Entries :     4297 : Total  Size=      17828 bytes  File Size  =       1745 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   9.91     *
*.....................................................................*
*Br    9 :veto_NvetoAboveThreshold : Int_t (int)                      *
*Entries :     4297 : Total  Size=      17832 bytes  File Size  =       3357 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   5.15     *
*.....................................................................*
*Br   10 :veto_VetoInTimeWindow : Int_t (int)                         *
*Entries :     4297 : Total  Size=      17820 bytes  File Size  =        766 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  22.56     *
*.....................................................................*
*Br   11 :veto_NVetoInTimeWindow : Int_t (int)                        *
*Entries :     4297 : Total  Size=      17824 bytes  File Size  =       3645 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   4.74     *
*.....................................................................*
*Br   12 :sAna_pointsoverthres_map : map<int,int> (map<int,int>)      *
*Entries :     4297 : Total  Size=     538864 bytes  File Size  =     234765 *
*Baskets :       18 : Basket Size=      32000 bytes  Compression=   2.29     *
*.....................................................................*
*Br   13 :sAna_risetime_map : map<int,int> (map<int,int>)             *
*Entries :     4297 : Total  Size=     538717 bytes  File Size  =     221989 *
*Baskets :       18 : Basket Size=      32000 bytes  Compression=   2.42     *
*.....................................................................*
```

Process observable

Process name

REST versioning strategy

- All REST available releases: https://github.com/rest-for-physics/framework/releases
- Each library / package is a git submodule with independent versioning
- Each framework release defines a version for each submodule
- All REST classes have **ROOT dictionaries** associated
- This enables **compatibility with older REST versions**

v2.3.12 Latest

Library, packages and project updates (Check their respective repositories to know about changes there):

- detectorlib updated to version 1.7.
- rawlib updated to version 1.6.
- connectorslib updated to version 1.2.
- tracklib updated to version 1.3.

What's Changed

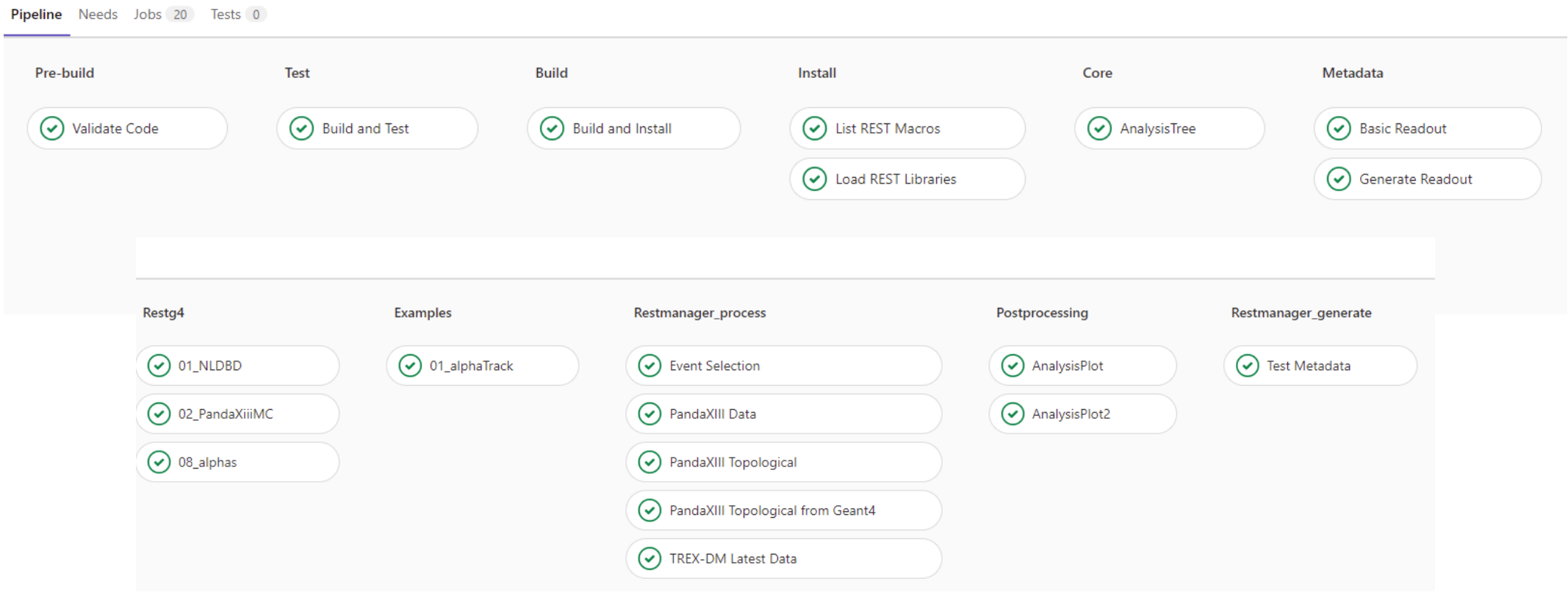
- Garfield compilation fix by @jgalan in #158
- TRestEventProcess::Begin/EndOfEventProcess are now virtual by @jgalan in #170
- Support for testing framework by @lobis in #129
- TRestRun::TRestRun(string) constructor supports both root file and rml file initialization. by @nkx111 in #165

REST versioning strategy

- One of the main goals of REST is to provide **traceability** and **reproducibility**
- **Metadata** objects have members to guarantee this:
  - **fVersion**: A string containing the human version number.
  - **fCommit**: The latest commit hash value when the compilation took place.
  - **fLibraryVersion**: The human readable library version. It is fixed by CMakeLists at the library submodules.
  - **fOfficialRelease**: It will be true if the commit was tagged at the repository
  - **fCleanState**: It will be true if there are no local modifications (including submodules)

REST validation strategy

- REST has a growing amount of testing implemented via the GoogleTest framework and GitLab CI pipelines



REST Geant4 Library

- *libRestGeant4.so*: https://github.com/rest-for-physics/geant4lib
- Store and Analyze Geant4 simulation data without Geant4 installed.
- Package ***restG4*** links to *libRestGeant4.so and Geant4, but we don't need Geant4 to analyze data.*
- Most of the Geant4 simulation data is stored thanks to this library, we can then produce smaller events (such as **TRestDetectorEvent**) for smaller file sizes.
- *Core classes:*
  - ***TRestGeant4Event***: simulation event information, useful methods to compute energy in each volume, etc.
  - ***TRestGeant4Metadata***: stores the configuration of the Geant4 simulation (which runs via restG4), such as primary generator, number of events, sensitive detectors…
  - **TRestGeant4Track/TRestGeant4Hits**: store similar information as Geant4 track and steps.

REST Detector Library

- *libRestDetector.so*: https://github.com/rest-for-physics/detectorlib
- Define detector response, detector properties (gas), event reconstruction / production from simulations
- Optionally links to Garfield++ for gas properties / processes
- REST originally designed for gaseous topological detectors (TPC), but not limited to this
- Important classes:
  - **TRestDetectorGas**: gas properties generated using Garfield++, does not need Garfield++ to work, only to produce “gas files”
  - **TRestDetectorSignalEvent**: detector signals, usually multiple channels.
  - **TRestDetectorHitsEvent**: positional and energy information about hits, either reconstructed from a TRestDetectorSignalEvent or produced via simulation. A readout definition is required for reconstruction
  - **TRestDetectorReadout**: Complete information about detector readout, mapping between channel ids and pixels etc.

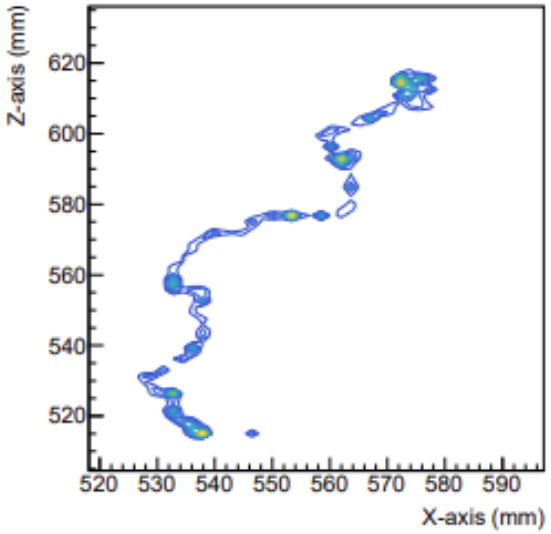
REST Detector Library

- Example: From Geant4 data to Detector data

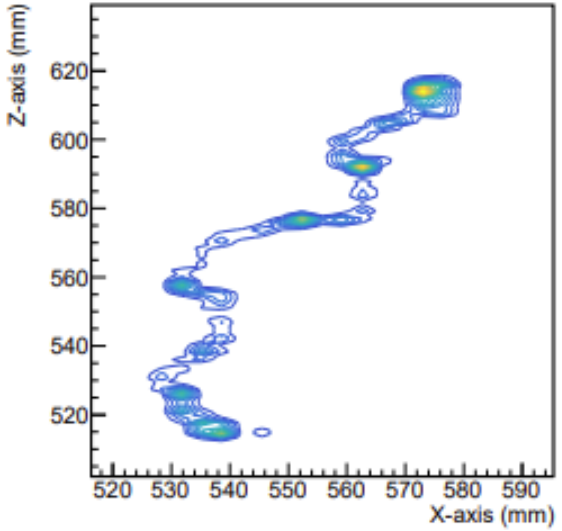
TRestGeant4Event → TRestGeant4To DetectorHitsProcess → **TRestDetectorHitsEvent**

Store a set of {position, energy} hits, remove Geant4 track information, step processes names…
Ignore hits outside detector volume

TRestDetectorHitsEvent → TRestDetector ElectronDiffusionProcess → **TRestDetectorHitsEvent**

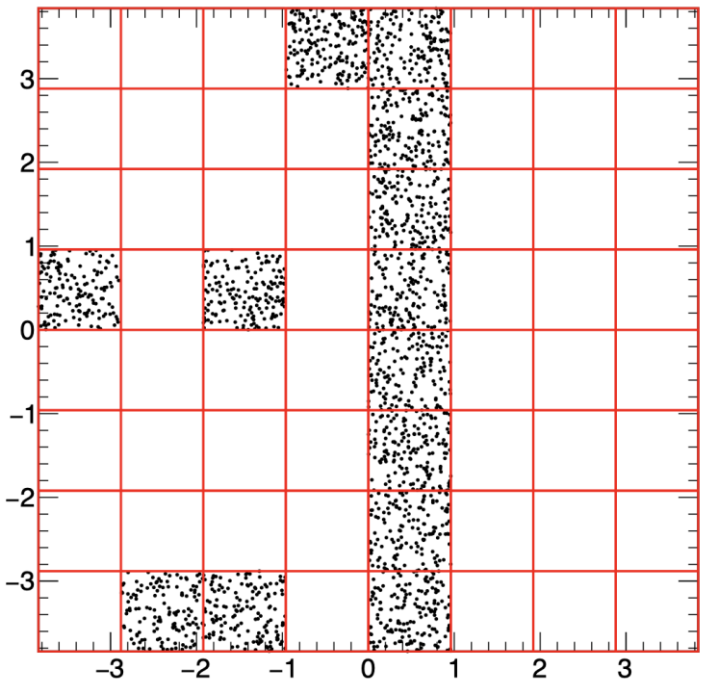


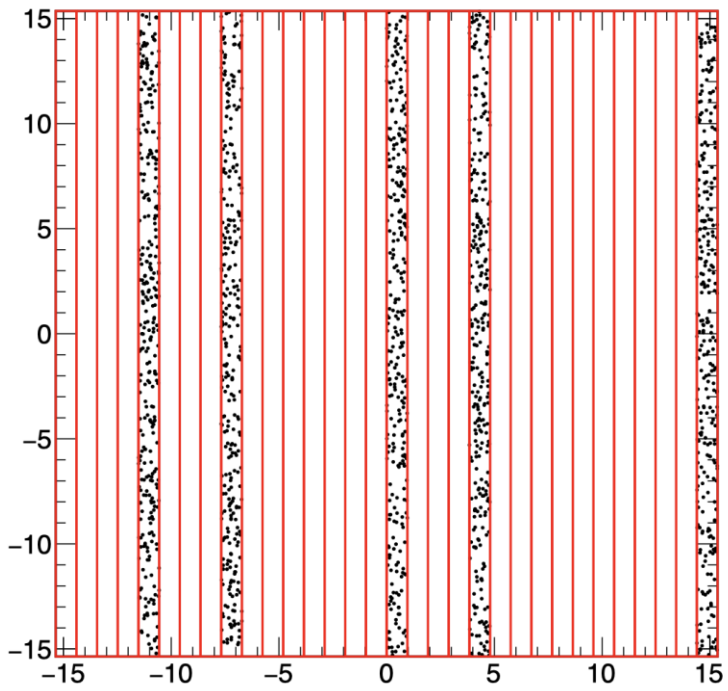
Geant4 simulation produces hits in gas (sensitive volume)
This process simulates the drift of electrons and diffusion, using detector properties such as drift electric field, gas properties and readout position

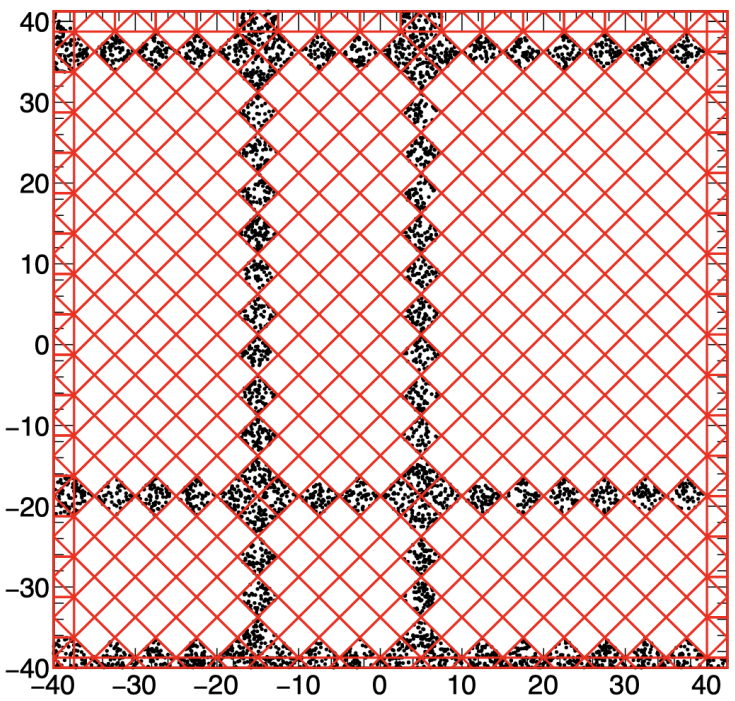


REST Detector Library

- *REST Detector Readouts example:* https://github.com/rest-for-physics/basic-readouts
- A readout is a mapping between electronic channels and the physical pixels of the detector
- Needed for event reconstruction / production via simulations





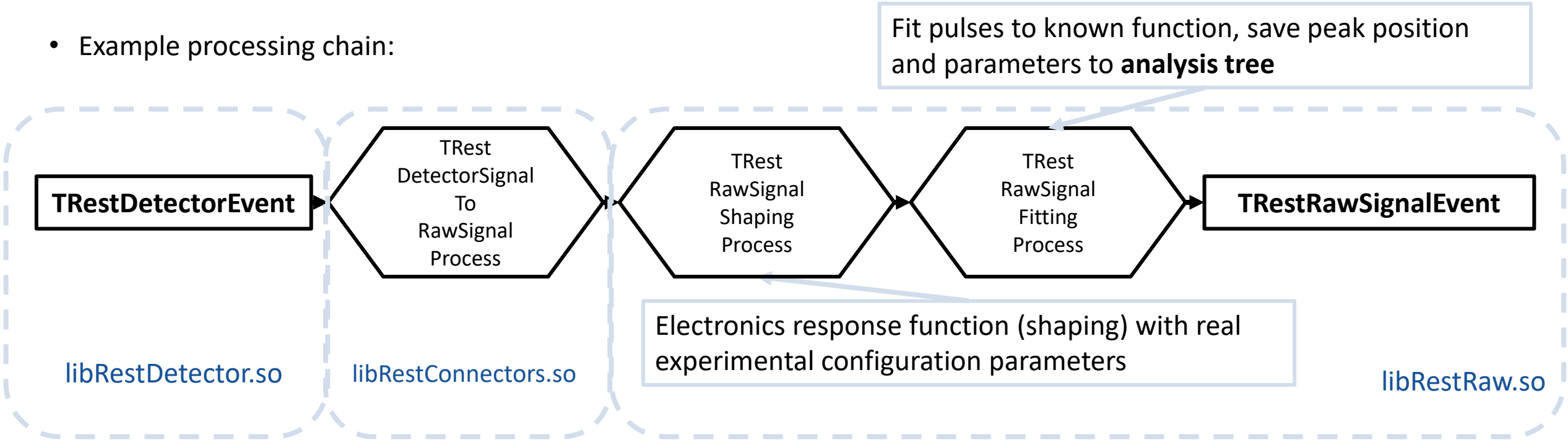


REST Connectors Library

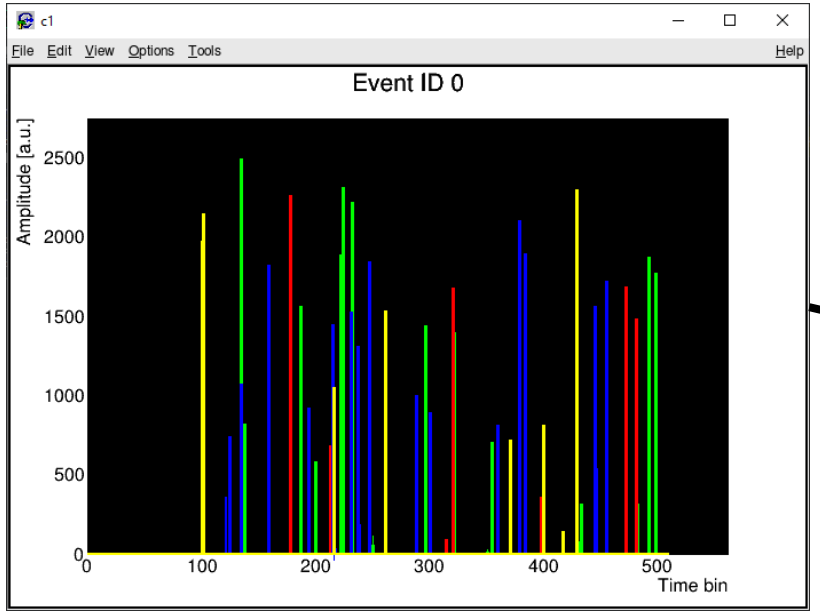
- *libRestConnectors.so*: https://github.com/rest-for-physics/connectorslib
- Library needed to connect events from different libraries

REST Raw Library

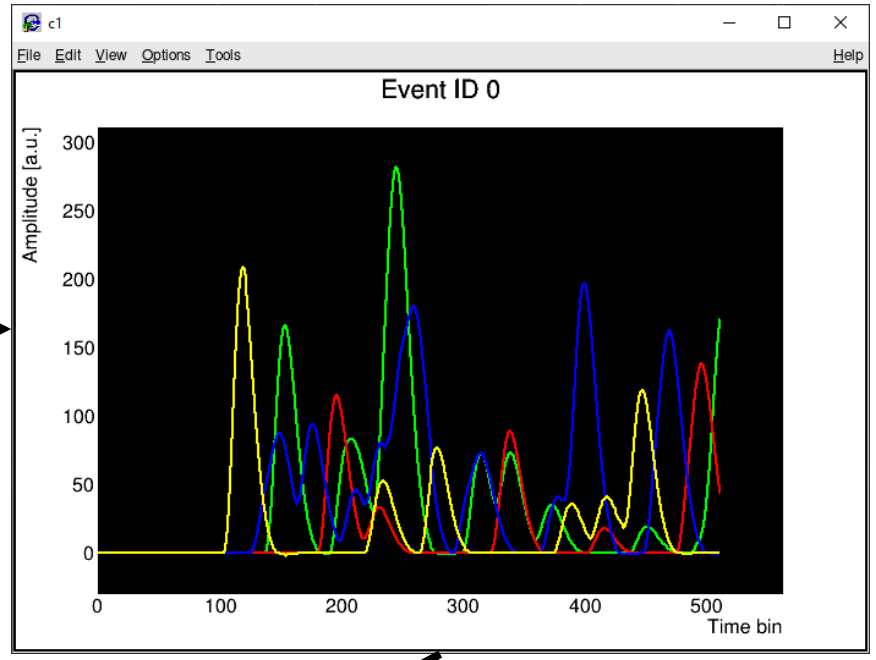
- *libRestRaw.so*: https://github.com/rest-for-physics/rawlib
- Store and process raw detector data from DAQ
- **TRestRawSignalEvent**: collection of signals, one for each channel (**TRestRawSignal**) with data as *std::vector<Short_t>*
- Example processing chain:



REST Raw Library

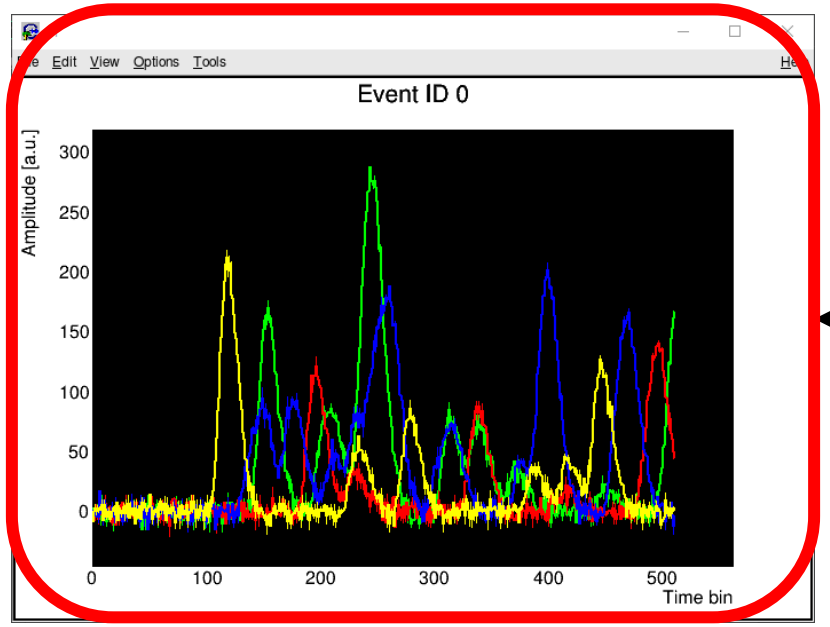


TRest
RawSignal
Shaping
Process



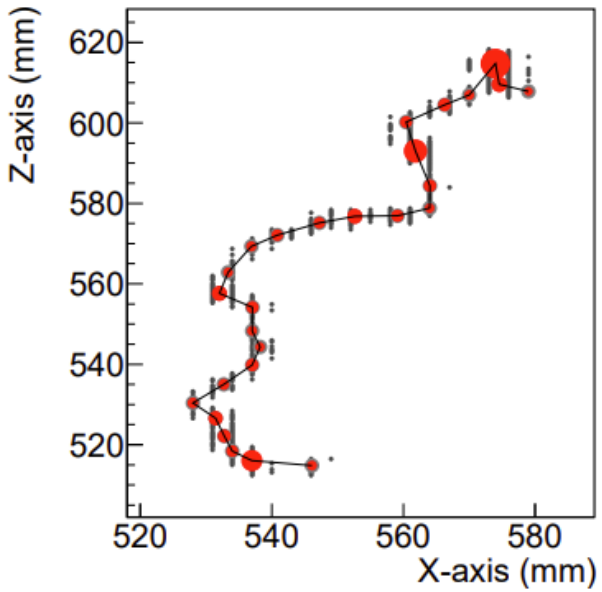
TRest
RawSignal
AddNoise
Process

Add random noise to emulate real data

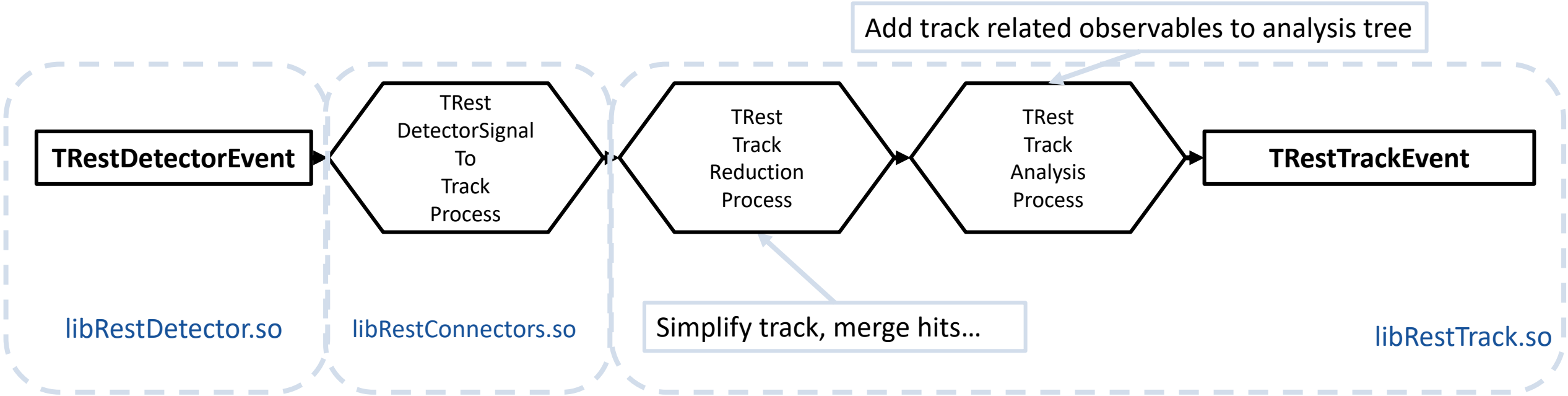


REST Track Library

- *libRestTrack.so*: https://github.com/rest-for-physics/tracklib
- Handles track reconstruction and processing
- **TRestTrackEvent** contains tracks in a **std::vector<TRestTrack>**



- Example processing chain:



REST *restG4* package

- *restG4*: https://github.com/rest-for-physics/restG4
- User fully configurable Geant4 simulation via RML (XML) files, allowing user to generate simulated data without writing a single line of C++ / Geant4 code
  - Define geometry using GDML files
  - Select sensitive detectors, active volumes in RML file
  - Primary generation using Geant4-like generators
  - Automatic handling of sub-event splitting (activation decays etc.)

```
<restG4>
```

simulation.rml example file:

Defines run information

```
<TRestRun name="DemoRun" title="A Demo Run">
    <parameter name="experimentName" value="DemoExperiment"/>
    <parameter name="runNumber" value="auto"/>
    <parameter name="runDescription" value="Geant4 Simulation"/>
    <parameter name="verboseLevel" value="1"/>
    <parameter name="outputFileName"
value="[fRunType]_[fRunTag]_run[fRunNumber].root"/>
</TRestRun>
```

Simulation parameters

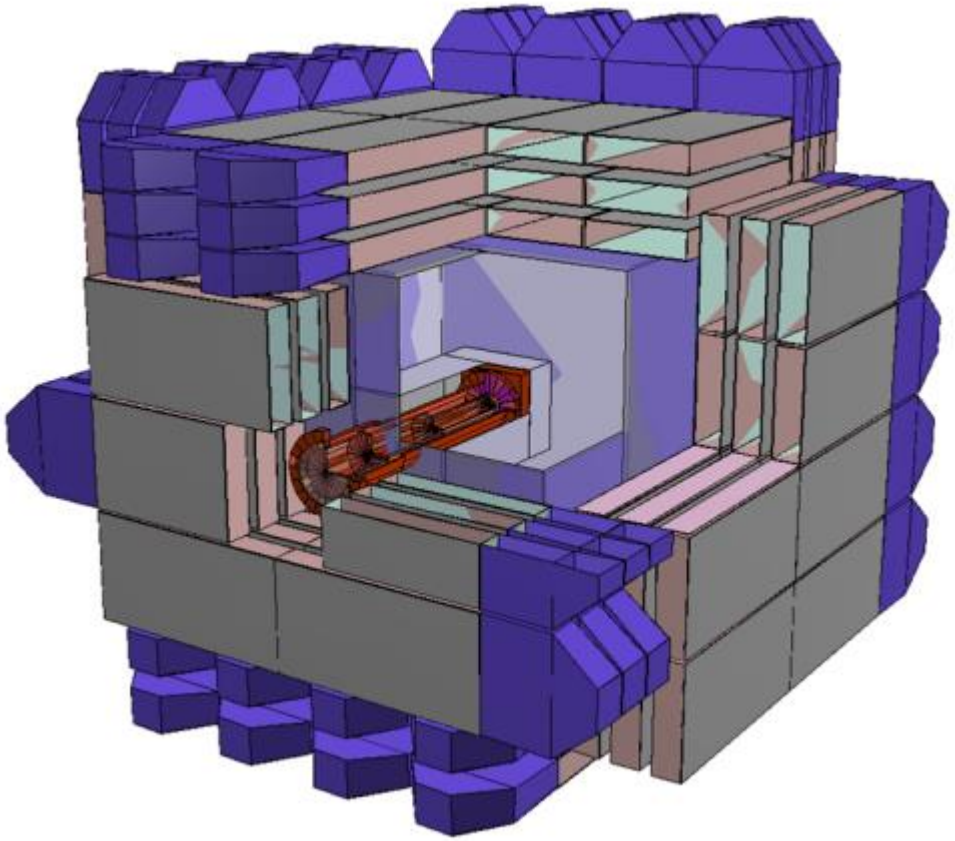
```
<TRestGeant4Metadata name="restG4 run" title="Cosmic Muons">
    <parameter name="Nevents" value="1000" />
    <parameter name="gdml_file" value="geometries/layered-
cylinder/geometry.gdml" />
    <parameter name="subEventTimeDelay" value="100us" />
    <parameter name="saveAllEvents" value="false" />
    <parameter name="printProgress" value="true" />

    <generator type="surface" shape="wall" position="(0,0,50)cm"
size="(30,30,0)cm" rotationAngle="0" rotationAxis="(1,0,0)">
        <source particle="mu-" excitedLevel="0.0" fullChain="on">
            <energyDist type="TH1D" file="Muons.root" spctName="cosmicmuon" />
            <angularDist type="TH1D" file="CosmicAngles.root" spctName="Theta2"
direction="(0,0,-1)" />
        </source>
    </generator>

    <storage sensitiveVolume="detector">
        <parameter name="energyRange" value="(0,10)" units="GeV" />
    </storage>
</TRestGeant4Metadata>
```

Geant4 PhysicsLists

```
<TRestGeant4PhysicsLists name="default" title="Physics List implementation."
verboseLevel="warning">
    <parameter name="cutForGamma" value="1" units="um" />
    <parameter name="cutForElectron" value="1" units="um" />
    <parameter name="cutForPositron" value="1" units="um" />

    <parameter name="cutForMuon" value="1" units="mm" />
    <parameter name="cutForNeutron" value="1" units="mm" />
    <parameter name="minEnergyRangeProductionCuts" value="1" units="keV" />
    <parameter name="maxEnergyRangeProductionCuts" value="1" units="GeV" />

    <!-- EM Physics lists -->
    <physicsList name="G4EmLivermorePhysics"></physicsList>

    <!-- Decay physics lists -->
    <physicsList name="G4DecayPhysics"></physicsList>
    <physicsList name="G4RadioactiveDecayPhysics"></physicsList>
    <physicsList name="G4RadioactiveDecay">
        <option name="ICM" value="true" />
        <option name="ARM" value="true" />
    </physicsList>

    <!-- Hadron physics lists -->
    <physicsList name="G4HadronElasticPhysicsHP"></physicsList>
    <physicsList name="G4IonBinaryCascadePhysics"></physicsList>
    <physicsList name="G4HadronPhysicsQGSP_BIC_HP"></physicsList>
    <physicsList name="G4EmExtraPhysics"></physicsList>

</TRestGeant4PhysicsLists>
```

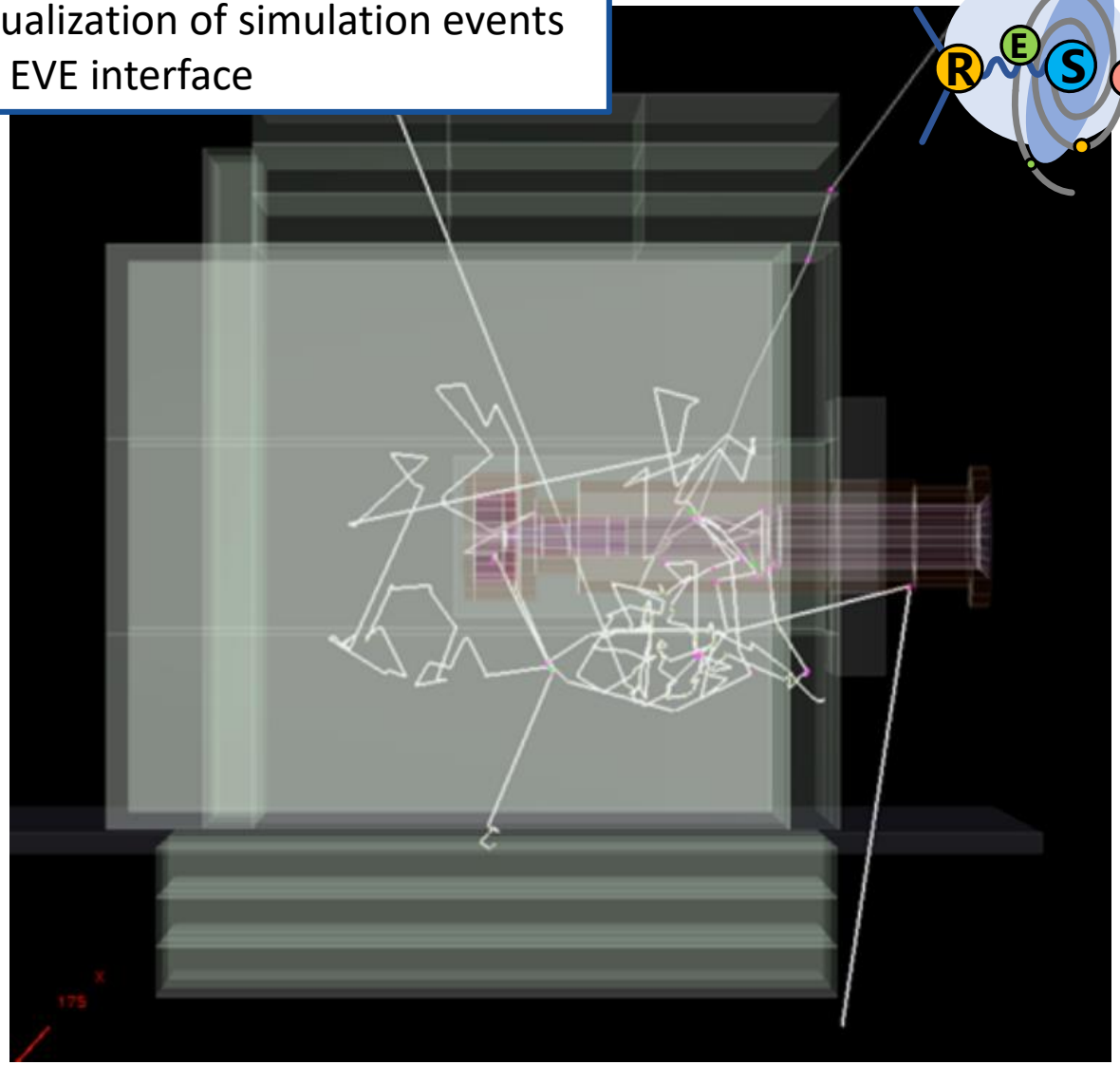
```
</restG4>
```

REST *restG4* package



TGeoManager geometry stored in simulation file

Visualization of simulation events via EVE interface



REST Installation instructions

- Detailed instructions at https://rest-for-physics.github.io/installation
- ROOT is main requirement (only compilation with C++17 is officially supported)
- Build and installation via CMake

tag version or commit

Pulling libraries / packages source code
User is free to checkout another revision
(as long as they are compatible)

Selecting which libraries / packages to install

```
git clone https://github.com/rest-for-physics/framework.git
cd framework
git reset --hard v2.3.11
python3 pull-submodules.py --clean
mkdir -p build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/your/install/directory -DCMAKE_CXX_STANDARD=17 \
    -DRESTLIB_GEANT4=ON -DRESTLIB_DETECTOR=ON -DRESTLIB_RAW=ON -DRESTLIB_TRACK=ON
-DREST_G4=ON -DREST_GARFIELD=OFF
make -j$(nproc) install
source /your/install/directory/thisREST.sh
```

REST usage example

- Possibility to follow demo using docker:
  - docker run –it --rm ghcr.io/lobis/rest-for-physics-demo:latest /bin/bash
- ***restRoot*** opens a root prompt with REST libraries loaded,

useful to discover all classes and user interfaces using tab-completion

```
root@6f270000814c:/# restRoot
REST WARNING!! Lacking system env "USER"!
Setting user name to : "defaultUser"

= REST Version: v2.3.10
= Loading libraries ...
 - libRestDetector.so
 - libRestFramework.so
 - libRestTrack.so
 - libRestRaw.so
 - libRestGeant4.so
 - libRestConnectors.so

root [0] TRestRun run;
root [1] run.
```

- Thanks to PyROOT, we can use REST from Python / Jupyter notebooks!
  - DISCLAIMER: REST wasn't originally designed for this, so API may be lacking (improvement suggestions are welcome!)
  - docker run -it --rm -p 8888:8888 ghcr.io/lobis/rest-for-physics-demo:latest jupyter-lab --ip=0.0.0.0 --port=8888
  - Demo also available on Binder (but not 100% availability)

REST usage example (Python)

Load REST libraries in PyROOT

```python
import ROOT

libraries = [
    "libRestFramework.so",
    "libRestRaw.so",
    "libRestGeant4.so",
    "libRestConnectors.so",
    "libRestTrack.so",
    "libRestDetector.so",
]

for library in libraries:
    ROOT.gSystem.Load(library)
```

Check access to REST classes

```python
event = ROOT.TRestGeant4Event()
event.PrintEvent()
```

```
*******************************************************
 EVENT ID : 0
 TIME : Thu, 01 Jan 1970 00:00:00 +0000 (GMT) +        0 nsec
 SUB-EVENT ID : 0
 Status : OK
*******************************************************
Total energy : 0 keV
Sensitive volume energy : 0 keV
Source origin : (0,0,0) mm
Number of active volumes : 0
-------------------------------------------------------------
Total number of tracks : 0
```

REST usage example (restG4)

(1) You can also call REST packages without Python bindings (using !)

```
!restG4 --help

restG4 requires at least one parameter, the rml configuration file (-c is optional)

example: restG4 example.rml

there are other convenient optional parameters that override the ones in the rml file:
        -h or --help | show usage (this text)
        -c example.rml | specify RML file (same as calling restG4 example.rml)
        -g geometry.gdml | specify geometry file
        -i | set interactive mode (default=false)
        -s | set serial mode (no multithreading) (default=true)
        -t nThreads | set the number of threads, also enables multithreading
```

(2) Let's run a simulation with restG4!

```
!restG4 simulations/simulation.rml
```

(3) You can see config file contents via console or

```
!cat simulations/simulation.rml
```

(4) To see ROOT file contents:

```
filename = "restG4_CosmicMuons_run00001.root"

file = ROOT.TFile(filename)

file.ls()

TFile**         restG4_CosmicMuons_run00001.root
 TFile*         restG4_CosmicMuons_run00001.root
  KEY: TRestAnalysisTree        AnalysisTree;3  AnalysisTree
  KEY: TTree    EventTree;3     TRestGeant4EventTree
  KEY: TRestRun DemoRun;3       A Demo Run
  KEY: TRestGeant4Metadata      restG4 run;2    Cosmic Muons
  KEY: TRestGeant4PhysicsLists  default;2       Physics List implementation.
```

(5) To access simulation event information:

```
run = ROOT.TRestRun(filename)

run.Print()

print(f"This run has {run.GetEntries()} entries")

event = ROOT.TRestGeant4Event()

run.SetInputEvent(event)

run.GetEntry(0)

event.PrintEvent()
```

Conclusions

- REST-for-Physics framework is a ROOT based software for processing and analysis of experimental or Monte Carlo event data
- The main data structure of REST is the event
- Processes are user configurable via RML (XML) files and store information in the form of Metadata which allows for traceability
- REST provides a package (restG4) for production of Monte Carlo data via Geant4
- REST is being used as the principal software tool in experiments such as IAXO
- REST is being actively developed and open to constructive criticism

Thanks for your attention!

Any questions?