eurizon
European network
for developing new horizons for RIs

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

CERN

WP5: Task 5.3: Development of software for the design of an SCT HEP detector

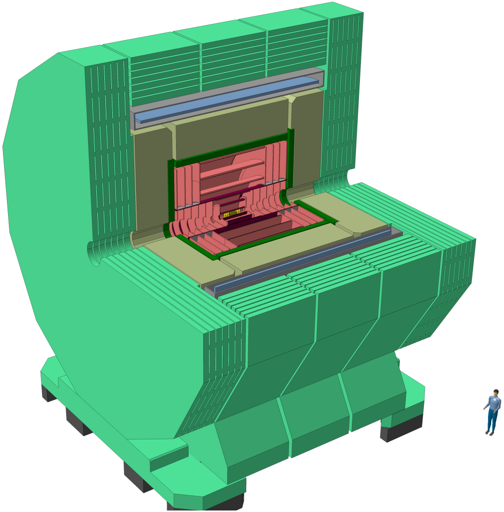
André Sailer

CERN

Eurizon Annual Meeting
GSI, Darmstadt
Feburary 9, 2023

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Table of Contents

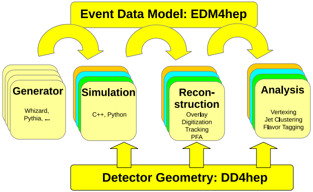


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Key4hep: Turnkey Software Stack

Create a software stack that connects and extends individual packages towards a complete data processing framework for detector studies with fast or full simulation, reconstruction, and for analysis

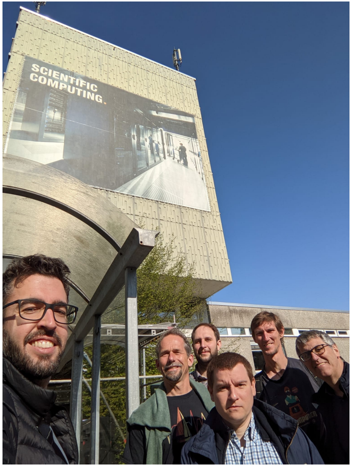
- Major ingredients: Event Data Model (EDM), Geometry Information, Processing Framework
- Sharing common components reduces overhead for all users
- Should be easy to use for librarians, developers, users
  - **easy to deploy, extend, set up**
- Full of functionality: plenty of examples for simulation and reconstruction of detectors
- Preserve and adapt existing functionality into the stack, e.g., from iLCSoft, FCCSW, CEPCSW



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

International Community

- Contributors/Interest from China, Germany, Italy, Americas, CERN; CEPC, CLIC, EIC, FCC, ILC, MuonCol
  - Big overlap with EIC software (DD4hep, podio/EDM4hep, ACTS, spack)
- CERN EP RnD work package 7.1 **https://ep-rnd.web.cern.ch/topic/software/turnkey-software-stack**
- Video Meetings:
  - Tuesday, 9:00 AM CET, weekly alternating EDM4hep/Key4hep **https://indico.cern.ch/category/11461/** (once per month: 3 PM CET for collaborators in Americas (EIC))
- GitHub Project: **http://github.com/key4hep**
- Documentation: **http://cern.ch/key4hep**



eurizon
European network
for developing new horizons for RIs

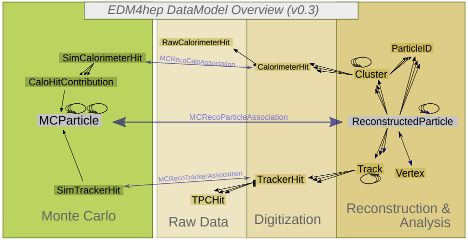
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

CERN

The Key4hep EDM: EDM4hep

For a high degree of interoperability, EDM4hep provides a common event data model

- Using podio to manage the EDM (described by `yaml`) and easily change the persistency layer (ROOT, SIO, . . .)
- EDM4hep data model based on LCIO and FCC-edm
- **http://github.com/key4hep/edm4hep**
- Recent developments for podio or EDM4hep
  - EDM4hep: additional types, associations
  - podio: event, run, collection metadata; UserDataCollection, Subset Collections, Frame
- A number of issues still need to be resolved
  - "Wrapper" for using different hit types transparently
  - multi-threading (the *frame* was recently added to simplify this)
  - schema evolution



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Framework: Gaudi

- Data processing frameworks are the skeleton on which HEP applications are built
- Gaudi was chosen as the framework, based on considerations for
  - portability to various computing resources, architectures and accelerators
  - support for task-oriented concurrency
  - adoption and developer community size; is used by LHCb, ATLAS
- Contribute developments were we see a need

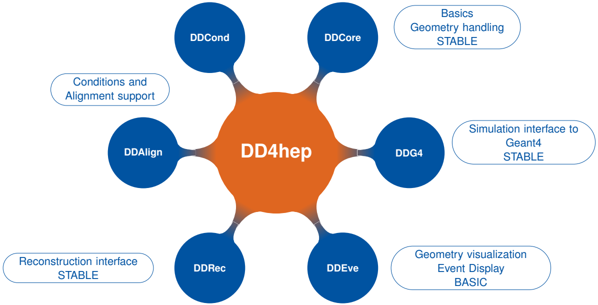
k4FWCore

- Basic IO functionality: podio data service
- Reproducible random number seeding

eurizon
European network
for developing new horizons for RIs
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Geometry Information: DD4hep

- Complete Detector Description
  - Providing geometry, materials, visualization, readout, alignment, calibration. . .
- Single source of information → consistent description
  - Use in simulation, reconstruction, analysis
- Supports full experiment life cycle
  - Detector concept development, detector optimization, construction, operation
  - Facile transition from one stage to the next

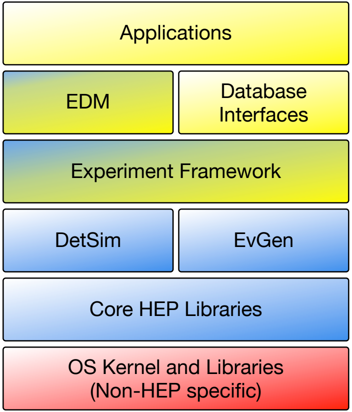


- DD4hep already in use by ILC, CLIC, FCC, and many more

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Packaging: Spack

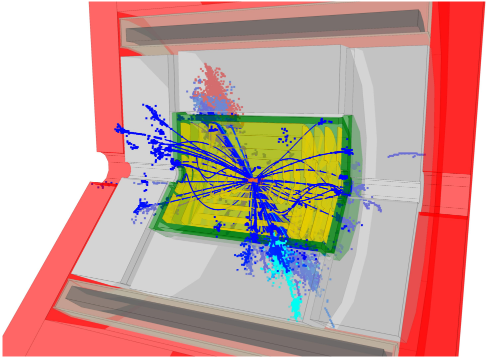
- Need to build a large number of packages to run our applications
- Adopted **spack** as the package manager
- Go beyond sharing of *build results* to sharing of *build recipes*
  - Many packages have build recipes provided by the spack community
  - Separate repository for Key4hep specific recipes
- Can build any and all pieces of the stack with minimum effort
  - `spack install key4hep-stack`
  - `spack dev-build conformaltracking@master`
- Used for nightly builds and releases of the stack
  - `source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh`



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

CLIC Reco Evolution: Adiabatic Changes

- Full CLIC reconstruction implemented in iLCSoft
- While transitioning to Key4hep, need to be able to keep running the CLIC reconstruction
- Switch components one by one, validate changes
  - Geometry provided by DD4hep, no changes needed
  - Move framework from Marlin to Gaudi: wrap existing processors
  - Move from `LCIO` to EDM4hep
  - Replace wrapped processors with native Gaudi algorithms, where necessary
- Incidentally will make iLCSoft functionality available to other users of the stack



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Marlin & Gaudi

Apart from some naming conventions, very similar ideas in the two frameworks*

	Marlin	Gaudi
language	`c++`	`c++`
working unit	Processor	Algorithm
configuration language	XML	Python
set up function	`init`	`initialize`
working function	`processEvent`	`execute`
wrap up function	`end`	`finalize`
Transient data format	`LCIO`	anything
Executable	Marlin	k4run

- To start using Gaudi: use a generic wrapper around the processors.
- Implementation: **https://github.com/key4hep/k4MarlinWrapper**
- Read LCIO files and pass the `LCIO::Event` to our processors

*Of course subtle differences emerge

eurizon
European network
for developing new horizons for RIs

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

CERN

Wrapper Configuration

- Translate the XML to python, using a stand alone python script: convertMarlinSteeringToGaudi.py
- Pass arbitrary number, types, and names of parameters to the processor

Marlin/XML

```xml
<processor name="VXDBarrelDigitiser" type="DDPlanarDigiProcessor">
  <parameter name="SubDetectorName" type="string">Vertex </parameter>
  <parameter name="IsStrip" type="bool">false </parameter>
  <parameter name="ResolutionU" type="float"> 0.003 0.003 0.003 0.003 0.003 0.003 </parameter>
  <parameter name="ResolutionV" type="float"> 0.003 0.003 0.003 0.003 0.003 0.003  </parameter>
  <parameter name="SimTrackHitCollectionName" type="string" lcioInType="SimTrackerHit">
        VertexBarrelCollection </parameter>
  <parameter name="SimTrkHitRelCollection" type="string" lcioOutType="LCRelation">
        VXDTrackerHitRelations </parameter>
  <parameter name="TrackerHitCollectionName" type="string" lcioOutType="TrackerHitPlane">
        VXDTrackerHits </parameter>
  <parameter name="Verbosity" type="string">WARNING </parameter>
</processor>
```

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No. 871072

Wrapper Configuration

- Translate the `XML` to `python`, using a stand alone python script: `convertMarlinSteeringToGaudi.py`
- Pass arbitrary number, types, and names of parameters to the processor

Gaudi/Python

```
VXDBarrelDigitiser = MarlinProcessorWrapper("VXDBarrelDigitiser")
VXDBarrelDigitiser.OutputLevel = WARNING
VXDBarrelDigitiser.ProcessorType = "DDPlanarDigiProcessor"
VXDBarrelDigitiser.Parameters = {
                                "IsStrip": ["false"],
                                "ResolutionU": ["0.003"] * 6,
                                "ResolutionV": ["0.003"] * 6,
                                "SimTrackHitCollectionName": ["VertexBarrelCollection"],
                                "SimTrkHitRelCollection": ["VXDTrackerHitRelations"],
                                "SubDetectorName": ["Vertex"],
                                "TrackerHitCollectionName": ["VXDTrackerHits"],
                                }
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Configuration: Control flow

- XML `execute` section translated to a python list

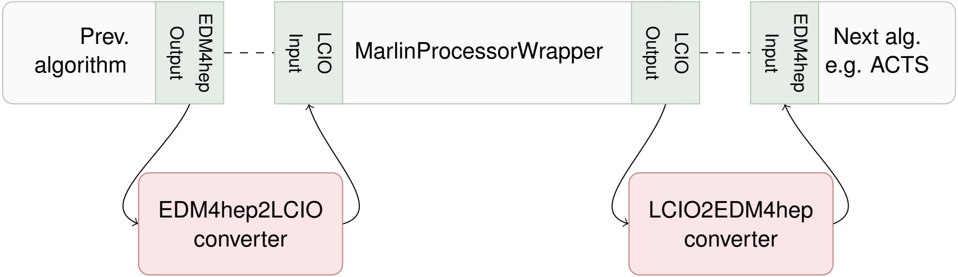
```xml
<execute>
  <processor name="MyAIDAProcessor"/>
  <processor name="EventNumber" />
  <processor name="InitDD4hep"/>
  <processor name="Config" />
  <!-- ... -->
</execute>
```

```python
algList = []
algList.append(lcioReader)
algList.append(MyAIDAProcessor)
algList.append(EventNumber)
algList.append(InitDD4hep)
algList.append(OverlayFalse)
algList.append(VXDBarrelDigitiser)
#...
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Event Data Model Conversion in Memory

- To use EDM4hep files as primary input, have to convert EDM4hep to LCIO and back at run time
- Integrate iLCSoft processors with Gaudi-based processors
- Configurable which collections to convert for which processor



eurizon
European network
for developing new horizons for RIs

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

CERN

Event Data Model Conversion in Memory

- To use EDM4hep files as primary input, have to convert EDM4hep to LCIO and back at run time
- Integrate iLCSoft processors with Gaudi-based processors
- Configurable which collections to convert for which processor

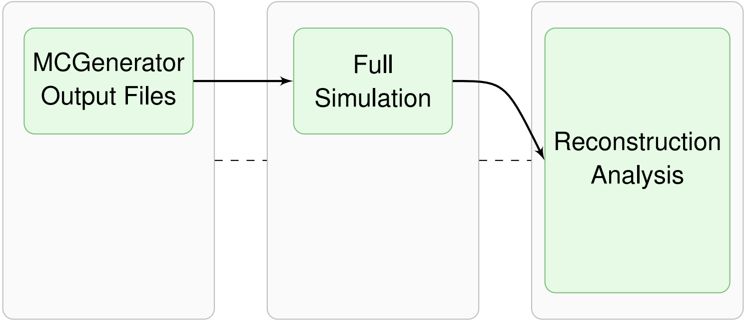
```
# EDM4hep to LCIO
edmConvTool = EDM4hep2LcioTool("VXDBarrelEDM4hep2lcio")
edmConvTool.Parameters = [
  "VertexBarrelCollection", "VertexBarrelCollection",
  ]
edmConvTool.OutputLevel = DEBUG
VXDBarrelDigitiser.EDM4hep2LcioTool = edmConvTool
```

```
# LCIO to EDM4hep
VXDBarrelDigitiserLCIOConv =
   Lcio2EDM4hepTool("VXDBarrelDigitiserLCIOConv")
VXDBarrelDigitiserLCIOConv.Parameters = [
  "VXDTrackerHits", "VXDTrackerHits",
  "VXDTrackerHitRelations", "VXDTrackerHitRelations"
  ]
VXDBarrelDigitiserLCIOConv.OutputLevel = DEBUG
VXDBarrelDigitiser.Lcio2EDM4hepTool =
    VXDBarrelDigitiserLCIOConv
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Simulation Integration in the Software Stack

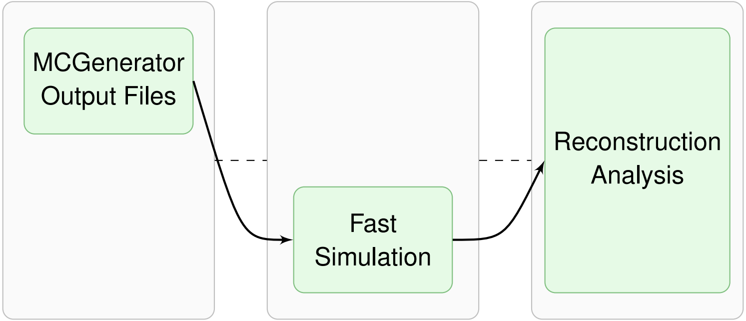
- The simulations can be run in a stand-alone mode using the output from a Generator as input
- Create its own input via “particle gun”, at least for full simulation
- Run as part of a chain inside a framework, where k4Gen calls a MC Generator, or reads an input file
- In all cases, the following step of (high level) reconstruction or analysis should be usable in the same way



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Simulation Integration in the Software Stack

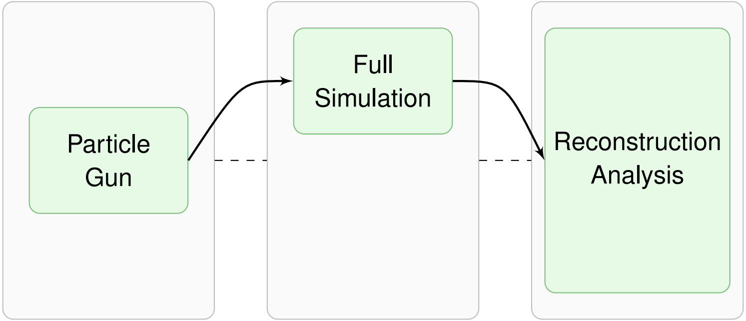
- The simulations can be run in a stand-alone mode using the output from a Generator as input
- Create its own input via "particle gun", at least for full simulation
- Run as part of a chain inside a framework, where k4Gen calls a MC Generator, or reads an input file
- In all cases, the following step of (high level) reconstruction or analysis should be usable in the same way



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Simulation Integration in the Software Stack

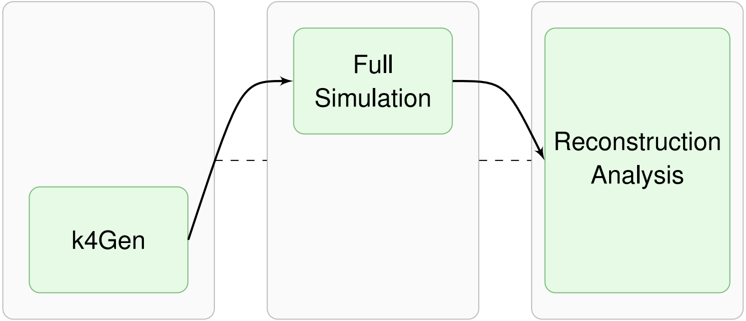
- The simulations can be run in a stand-alone mode using the output from a Generator as input
- Create its own input via “particle gun”, at least for full simulation
- Run as part of a chain inside a framework, where k4Gen calls a MC Generator, or reads an input file
- In all cases, the following step of (high level) reconstruction or analysis should be usable in the same way



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Simulation Integration in the Software Stack

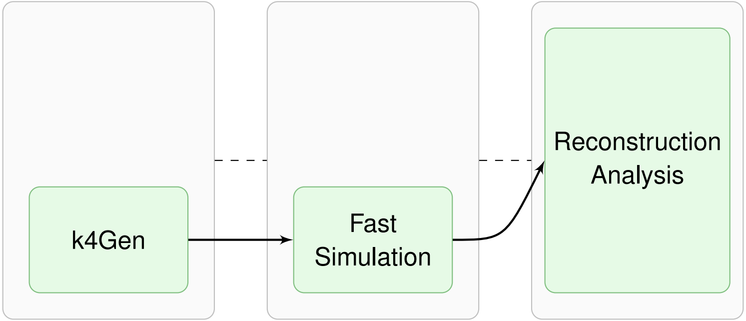
- The simulations can be run in a stand-alone mode using the output from a Generator as input
- Create its own input via "particle gun", at least for full simulation
- Run as part of a chain inside a framework, where k4Gen calls a MC Generator, or reads an input file
- In all cases, the following step of (high level) reconstruction or analysis should be usable in the same way



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Simulation Integration in the Software Stack

- The simulations can be run in a stand-alone mode using the output from a Generator as input
- Create its own input via “particle gun”, at least for full simulation
- Run as part of a chain inside a framework, where k4Gen calls a MC Generator, or reads an input file
- In all cases, the following step of (high level) reconstruction or analysis should be usable in the same way



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Geant4 Full Simulation Interfaces

- `ddsim` standalone program and `k4SimGeant4` framework integrated solution
  - Also looking at integrating with LHCb's `Gaussino` (Using DD4hep geometry, gaudi based processing)
- Both approaches have to provide the same functionality: sensitive detectors, MC History, particle guns, physics list construction and configuration,
  - Ideally by the same implementation, but we are not there yet

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No. 871072

ddsim: Full Simulation Example

- Only change needed to go from **LCIO** to EDM4hep output is the output file name

https://key4hep.github.io/key4hep-doc/examples/clic.html

```
source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh
git clone https://github.com/iLCSoft/CLICPerformance
ddsim --compactFile $LCGEO/CLIC/compact/CLIC_o3_v14/CLIC_o3_v14.xml \
      --outputFile ttbar.slcio \
      --steeringFile clic_steer.py \
      --inputFiles ../Tests/yyxyev_000.stdhep \
      --numberOfEvents 3
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

ddsim: Full Simulation Example

- Only change needed to go from LCIO to **EDM4hep** output is the output file name

https://key4hep.github.io/key4hep-doc/examples/clic.html

```
source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh
git clone https://github.com/iLCSoft/CLICPerformance
ddsim --compactFile $LCGEO/CLIC/compact/CLIC_o3_v14/CLIC_o3_v14.xml \
      --outputFile ttbar_edm4hep.root \
      --steeringFile clic_steer.py \
      --inputFiles ../Tests/yyxyev_000.stdhep \
      --numberOfEvents 3
```

eurizon
European network
for developing new horizons for RIs
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Configuring and Running k4SimGeant4

```
from Configurables import (SimG4Alg, SimG4SaveTrackerHits, SimG4UserLimitPhysicsList, GeoSvc, SimG4Svc,
                           SimG4FullSimActions)
# parse the given xml file
geoservice = GeoSvc("GeoSvc")
geoservice.detectors = [os.path.join(path_to_detectors, 'Detector/DetFCCeeIDEA/compact/FCCee_DectMaster.xml')]
# configure sensitive detector
savetrackertool_DCH = SimG4SaveTrackerHits("saveTrackerHits_DCH")
savetrackertool_DCH.readoutNames = ["DriftChamberCollection"]
savetrackertool_DCH.SimTrackHits.Path = "positionedHits_DCH"
SimG4Alg("SimG4Alg").outputs += [savetrackertool_DCH]
# Setup for physicslist
physicslisttool = SimG4UserLimitPhysicsList("Physics");     physicslisttool.fullphysics = "SimG4FtfpBert"
# enable MC history
actions = SimG4FullSimActions();      actions.enableHistory=True
# configure geant4
geantservice = SimG4Svc("SimG4Svc")
geantservice.detector = 'SimG4DD4hepDetector'
geantservice.physicslist = physicslisttool;      geantservice.actions = actions
geantservice.magneticField = field
geantservice.g4PostInitCommands +=["/process/eLoss/minKinEnergy 1 MeV", "/tracking/storeTrajectory 1"]
```

- Execute with `k4run simGeant4.py`, some steering files available for **FCC detectors**.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Delphes Fast Simulation

- "Delphes is a modular framework that simulates the response of a multipurpose detector in a parameterised fashion"
- Delphes integration to Key4hep framework: **key4hep/k4SimDelphes** and its **documentation**
- To integrate Delphes to Key4hep, we need to obtain EDM4hep output from it

eurizon
European network
for developing new horizons for RIs

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

CERN

Using Delphes Fast Simulation: Standalone

- Pick the Delphes card of your chosen detector
- Configuration for EDM4hep output: `edm4hep_output_config.tcl` which collections to store in the output file
- Pythia8 configuration: `p8_noBES_ee_ZH_ecm240.cmd`
- output file name: `delphes_events_edm4hep.root`

```
DelphesPythia8_EDM4HEP ${DELPHES_DIR}/cards/delphes_card_IDEA.tcl \
                       ${K4SIMDELPHES}/edm4hep_output_config.tcl \
                       p8_noBES_ee_ZH_ecm240.cmd \
                       delphes_events_edm4hep.root
```

There are other standalone programs in Key4hep to run for different input sources:

- `DelphesPythia8EvtGen_EDM4HEP`
- `DelphesROOT_EDM4HEP`
- `DelphesSTDHEP_EDM4HEP`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Using Delphes Fast Simulation: Framework Integrated

- Configure Delphes 'Algorithm' with similar arguments as standalone

```
#...
from Configurables import k4SimDelphesAlg
delphesalg = k4SimDelphesAlg()
delphesalg.DelphesCard = "delphes_card_IDEA.tcl"
delphesalg.DelphesOutputSettings = "edm4hep_output_config.tcl"
delphesalg.GenParticles.Path = "GenParticles"
#...
```

- Execute complete steering file: `k4run simDelphes.py`
  - Example **steering file**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

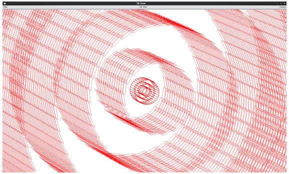
k4Clue

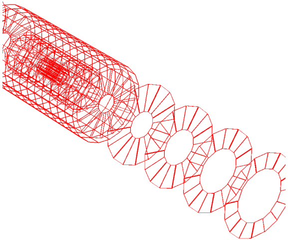
- Investigating use of the GPU friendly algorithm **CLUE** (CLUstering of Energy) as part of particle flow reconstruction
- CLUE Gaudi algorithm created: **k4Clue** and run as part of the CLIC reconstruction chain
- Validation and use of the clusters pending

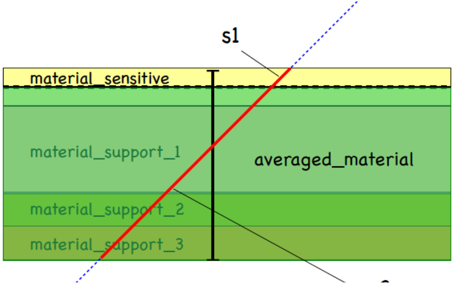
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

k4ActsTracking

- Started work towards integration of the ACTS tracking toolkit with Key4hep:
  - Planning to create thin Gaudi Algorithm(s) converting necessary information for ACTS and tracks back to EDM4hep
- Try to use information provided by `dd4hep::rec::Surface` class to ACTS
  - Surfaces can be added after the fact to the geometry instantiation





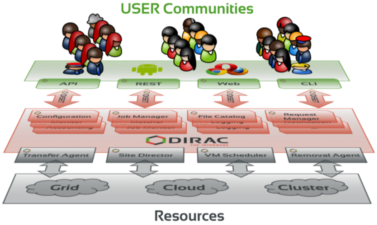


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Dirac in a Nutshell

iLCDirac is based on the DIRAC interware originally developed for LHCb

- Dirac (Distributed Infrastructure with Remote Agent Control): High level interface between users and distributed resources
- Distributed Workload Management: one interface to execute anywhere: batch farms, grid computing elements, HPCs
- Data Management (file transfers, meta data augmented file catalog)
- High degree of automation
- Web interface for controlling jobs

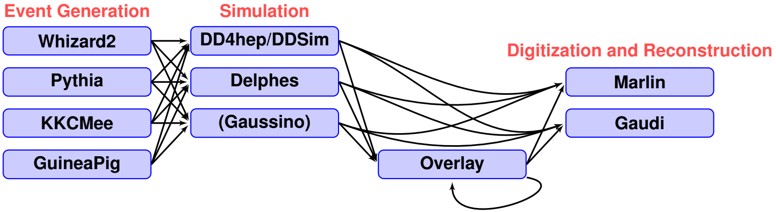


diracgrid.org

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

iLCDirac Use Cases

- The iLCDirac extension of Dirac is set up for the ILC, Calice, and FCC Virtual Organisations
- Centralized MC Production (Event Generation, Geant4 Simulation, Reconstruction)
- User jobs (Generation, Simulation, Reconstruction, Analyses)
- iLCDirac uses almost all functionality provided by DIRAC
- Specific features in iLCDirac
  - Workflow Modules for Key4hep Software (see later)



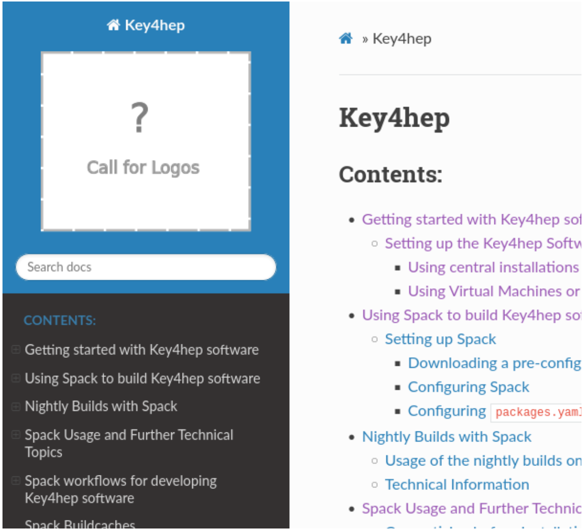
  - Overlay System for adding beam background files to MC jobs efficiently and effectively
  - Pandora Particle Flow calibration service (Marlin based)

Source Code: **https://gitlab.cern.ch/CLICdp/ILCDIRAC**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

Documentation

- Main documentation page **key4hep.github.io** based on GitHub pages **https://github.com/key4hep/key4hep-doc**
- Test the examples in the documentation via `notedown`
- Doxygen, e.g., EDM4hep**https://edm4hep.web.cern.ch/**
- CLIC simulation and reconstruction example
- Restructuring of documentation in the works
  - Separate *User*, *Developer*, *Librarian* content



This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No. 871072

Conclusions

- Developments for Key4hep software stack for future collider detector studies ongoing
- Fast and Full simulation, Reconstruction for a number of different detectors can be done with the same software installation
  - Developments of k4MarlinWrapper done by CERN Fellow partially funded EURIZON(f.k.a. CP)
- Inclusion of further detector models (e.g., IDEA, also part of Eurizon WP5 effort) on going
- Inclusion of further simulation and reconstruction tools on going
- Extension of iLCDirac for FCCee (or other Key4hep tool users) on going with technical student paid through Eurizon funds since this month