# Key4hep — TURNKEY SOFTWARE FOR FUTURE COLLIDERS*

Placido Fernandez Declara, Gerardo Ganis
Benedikt Hegner, Clement Helsens, Marko Petric
Andre Sailer, Valentin Volkl

CERN, Geneva, Switzerland

Frank Gaede, Thomas Madlener

DESY, Hamburg, Germany

Wenxing Fang, Weidong Li, Tao Lin, Xiaomei Zhang
Jiaheng Zou

IHEP, Bejing, China

Xingtao Huang, Teng Li

Shandong University, Qindao, Shandong, China

Sang Hyun Ko

Seoul National University, Seoul, Republic of Korea

Joseph Wang

Bitquant Digital Services, Hong Kong

*(Received March 31, 2021; accepted May 4, 2021)*

Future collider experiments rely on a well-maintained software for their physics performance studies and detector optimisation. The Key4hep project aims to design and provide a common set of software tools that can be used by future or even present-day high-energy physics projects. It unites the communities of all current future collider projects. These proceedings give

* Presented at XXVII Cracow Epiphany Conference on *Future of Particle Physics*, Cracow, Poland, January 7–10, 2021.

an overview of the goals of the Key4hep project and briefly describe the main components that are currently under development: the common event data model, EDM4hep, the detector description toolkit, DD4hep, the interfaces to Delphes, a brief description of the core framework and, finally, the infrastructure to deploy and build the whole software stack using spack. They also include some details about how different communities plan to adapt their software stacks to the Key4hep project. Overall, they show that the Key4hep software stack can be used already for first physics studies and highlight its potential as a baseline for future high-energy physics experiments.

DOI:10.5506/APhysPolB.52.1031

## 1. Introduction

High-energy physics (HEP) experiments rely on a well-maintained software for their successful operation. Providing this software and its integration with the experiment comes with significant challenges. Some of these challenges are related to the long lifetimes of HEP experiments, which can span decades. Others are related to the inherent complexity and size of the software stack of an experiment that is comprised of many different software packages and components. The Key4hep project aims at providing a complete, so-called, *turnkey software stack* for future collider projects. To achieve this goal, the project tries to connect and extend already available packages, and to develop a new software where it was necessary to provide a complete data processing framework from event generation and simulation through event reconstruction. While this framework provides the basic functionality, it should be easily extendable and adaptable with other software in the Key4hep software stack to support the different use cases of the different experiments.

The Key4hep project is community driven and unites collaborators from the different future collider projects that are currently under consideration: ILC, CLIC, FCC-ee, FCC-hh and CEPC. All software that is developed as a part of the Key4hep project is open source and hosted on GitHub [1]. Additionally, documentation on how to start using the available software is available [2] and aimed at helping users getting started quickly. It includes instructions for setting up existing environments, building the entire software stack as well as some examples for running simulation and reconstruction steps. There is a set of alternating bi-weekly meetings to discuss the progress as well as current issues of the Key4hep project and its event data model (EDM), EDM4hep, which we will also discuss briefly here in Section 2.

We continue with a brief description of the DD4hep toolkit for detector description in Section 3, before we briefly discuss the currently available possibilities to simulate first physics events in Section 4. The core framework

and the available software infrastructure are described in Sections 5 and 6 respectively. Finally, we describe the usage of the spack package manager that is used to build the Key4hep stack (Section 7), before we conclude with a summary and outlook in Section 8.

## 2. EDM4hep

At the core of every HEP experiment, the software framework is the event data model (EDM). It defines the interface and communication channels of the different components of the framework and, even more importantly, also the language that physicists use to express their ideas. The common EDM for the Key4hep project, EDM4hep [3], is based on experience that has been gathered with the LCIO EDM [4], which has been very successfully shared by the linear collider community for almost two decades, and FCC-edm [5]. EDM4hep is implemented using the podio EDM toolkit [6–8], which has already been used for FCC-edm. It aims at providing an efficient and easy-to-use implementation of different EDMs starting from a high-level description of the data types.

A schematic view is shown in Fig. 1. It features the typical data types that are used in HEP experiments to represent simulated as well as measurement data at different levels. It also allows to define relations among these types to build up object hierarchies, like the *ReconstructedParticle* that can comprise low-level measurements but can also be used to express short-lived decaying particles. Similar means are available for simulated data with a strict separation to the reconstruction side. To connect the two worlds, dedicated *association* data types are provided.

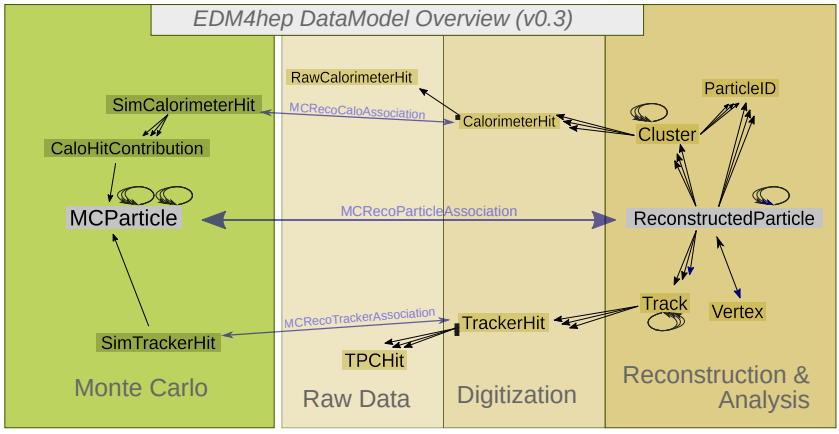


Fig. 1. Schematic view of the contents of EDM4hep and the relations among the different data types that it defines.

The current definition of EDM4hep is inspired by the requirements for precision physics at future lepton colliders, following a Particle Flow approach, where the aim is to reconstruct every individual particle emerging from the collision. Since Key4hep is also aimed at future hadron collider projects, EDM4hep also has to support the much more convoluted hadron collision environments. So far, it seems to be up to this challenge, but more testing is still necessary.

## 3. DD4hep

A second core part of every experiment framework is the description of the detector. For this purpose, Key4hep uses the DD4hep [9, 10] toolkit. It has originally been developed for the linear collider community, but had the whole HEP community in mind from the beginning. Full integration of Geant4 [11] full detector simulations using the DD4hep detector descriptions is ongoing. Nevertheless, it is now already possible to simulate the detector response with the *ddsim* executable [12] and the EDM4hep output plugin that has been developed for Key4hep. This allows for a seamless integration with other tools that use EDM4hep information.

## 4. k4SimDelphes

A slightly different approach that allows for first physics studies within Key4hep is the integration of the Delphes fast simulation framework [13]. It can be used to simulate the parameterized response of the detector consisting of a tracking system, embedded in a magnetic field, calorimeters, and a muon system. The framework offers executables to read different standard file formats (*e.g.* Les Houches Event Files or HepMC) and produces different output types such as isolated leptons, jets or missing transverse energy, which can be used for dedicated analyses. The k4SimDelphes package [14] wraps the Delphes simulation and converts its output into EDM4hep format.

At its current stage, k4SimDelphes offers the same executables as Delphes, and adds an additional one that also allows to use the EvtGen [15] plugin with PYTHIA 8. The contents of k4SimDelphes are configurable at run-time via a configuration file, which steers which Delphes branches should be converted and also how different branches should be combined into different EDM4hep collections. A default configuration that should cover the majority of use cases and can be used without changes with the default Delphes cards is shipped with k4SimDelphes.

The k4SimDelphes converter is also part of a general approach to simulation interfaces in Key4hep, where it should be possible to transparently replace different simulation modules that all produce more or less identical output. The full integration of k4SimDelphes into the core framework and an interface to different generators is currently under development.

## 5. Core framework

The core framework of Key4hep will be based on the Gaudi framework [16], which has been successfully used by the LHCb, ATLAS, the FCC software and smaller experiments. The Key4hep project contributes to the development of Gaudi where it is necessary in order to have all the required features. The k4FWCore [17] package provides the core functionality of the Key4hep framework, most importantly it has the functionality to read and write EDMs that have been generated via podio. Additionally, it also comes with tools for background overlaying and other fundamental functionality.

Other core functionality is currently being ported from FCCSW and is organized into different packages according to the different aspects that are covered by these packages, *e.g.*,

— k4Gen [18]: Framework components for interfacing to different generators and particle guns.

— k4SimGeant4 [19]: Framework components that deal with Geant4 simulations.

— k4SimDelphes: see Section 4.

— k4RecCalorimeter [20]: Framework components for calorimeter reconstruction.

Further components can be integrated in a similar fashion. The approach of separating the concerns of the different packages as well as possible should allow to pick and combine only the necessary components later on.

The linear collider community has been using the Marlin [21] event processing framework for more than 15 years. Hence, a lot of dedicated reconstruction and analysis software is already available for this framework. In order to ease the transition to the Gaudi based framework of Key4hep, a wrapper around these Marlin processors has been developed: k4MarlinWrapper [22]. It allows to run them using the Gaudi framework; it provides the necessary components and converters to interface between Marlin processors and other components of Key4hep. Further work that will ease the transition to the Key4hep software stack is ongoing.

For FCC, the transition is much easier since the experiment framework is already Gaudi-based in this case. Additionally, the FCC-edm is generated via podio so the major changes in the migration here are related to the slightly different definition of the data types in EDM4hep.

## 6. Software infrastructure

Building and deploying software for a HEP experiment is usually a time-consuming and challenging task. The main challenges are related to a large number of different systems and packages that have to be integrated, the fact that the developments of these components often happen on partially very different time scales and the fact that the whole stack should still be easily extendable to add new functionality. Sharing approaches and experiences as well as using common, community-wide installations and tooling can significantly increase the efficiency of these processes. Additional benefits can be reaped by using tools from the open-source community and from outside HEP.

To facilitate all of this, the Key4hep project tries to use modern software development practices. These include the usage of automated builds and continuous integration (CI) wherever possible to increase the probability of detecting problems before they are released to a wider audience. For the C++ parts, the project aims at using modern CMake [23] features, which allow for easier handling of dependent projects. To build the complete software stack, the spack package manager is used (see the next section). The distribution of the software stack is achieved via CVMFS, and the most recent release is available at `/cvmfs/sw.hsf.org/key4hep`. In order to gather as much feedback as possible at the early stages of the project, releases follow the "release early and release often" paradigm. This also allows to discover problems that slipped through automated tests and nightly builds to be discovered early.

## 7. Spack

The spack package manager [24], originally developed by the high-performance computing (HPC) community, makes it possible to deal with multiple different configurations of the same package in a software stack. It does not depend on the operating system and builds software packages from the source. It builds on top of the usual build tools like CMake or Autotools and integrating a new software package into the spack ecosystem can be done by defining an appropriate package definition using Python. Spack will then take care of resolving all the stated dependencies and making sure that they are built and available when a given package should be built. The Key4hep project offers a repository of package definitions for software packages that are not yet available via the central spack repository and which can be used to build these packages alongside the ones which are already available centrally. This makes it possible to use spack to build the central installation of the Key4hep software stack that is deployed via CVMFS.

## 8. Conclusions

The Key4hep project aims to provide a well-maintained and complete software stack for future collider experiment studies. All of the major groups — CEPC, CLIC, FCC and ILC — are actively contributing to Key4hep and have either already started the migration to the new software stacks or have developed migration strategies. Since its start in mid 2019 significant work has been put into defining a first version of the common EDM4hep and to provide some first tools for physics studies. Additionally, central pieces for deploying, building and testing the software have been put in place. The group is actively working on developing the Key4hep software ecosystem further and is welcoming new collaborators and users to this exciting project.

## REFERENCES

[1] GitHub Organization, https://github.com/key4hep

[2] V. Volkl *et al.*, «key4hep/key4hep-doc: Initial Release, Documentation for the Key4hep Turnkey Software», 2021, https://doi.org/10.5281/zenodo.4564650

[3] V. Volkl *et al.*, key4hep/EDM4hep: v00-03-02 GitHub Repository, https://doi.org/10.5281/zenodo.4785063

[4] F. Gaede, T. Behnke, N. Graf, T. Johnson, «LCIO — A Persistency framework for linear collider simulation studies», *eConf* **C0303241**, TUKT001 (2003), arXiv:physics/0306114.

[5] FCC-edm GitHub Repository, https://github.com/HEP-FCC/fcc-edm

[6] F. Gaede, B. Hegner, P. Mato, «podio: An Event-Data-Model Toolkit for High Energy Physics Experiments», *J. Phys.: Conf. Ser.* **898**, 072039 (2017).

[7] podio GitHub Repository, https://github.com/AIDASoft/podio

[8] F. Gaede, B. Hegner, G.A. Stewart, «podio: recent developments in the Plain Old Data EDM toolkit», *EPJ Web Conf.* **245**, 05024 (2020).

[9] M. Frank, F. Gaede, C. Grefe, P. Mato, «DD4hep: A Detector Description Toolkit for High Energy Physics Experiments», *J. Phys.: Conf. Ser.* **513**, 022010 (2013).

[10] M. Frank, F. Gaede, M. Petric, A. Sailer, Andre, «AIDASoft/DD4hep: v01-16-01», https://doi.org/10.5281/zenodo.592244

[11] S. Agostinelli *et al.*, «Geant4 — a simulation toolkit», *Nucl. Instrum. Methods Phys. Res A* **506**, 250 (2003).

[12] M. Petrić *et al.*, «Detector Simulations with DD4hep», *J. Phys.: Conf. Ser.* **898**, 042015 (2016).

[13] DELPHES 3 Collaboration (J. de Favereau *et al.*), «Delphes 3: a modular framework for fast simulation of a generic collider experiment», *J. High Energy Phys.* **1402**, 057 (2014), arXiv:1307.6346 [hep-ex].

[14] V. Volkl *et al.*, k4SimDelphes Delphes Integration in the Key4hep Framework, https://doi.org/10.5281/zenodo.4564683

[15] https://evtgen.hepforge.org/

[16] G. Barrand *et al.*, «Gaudi — A software architecture and framework for building HEP data processing applications», *Comput. Phys. Commun.* **140**, 45 (2001).

[17] V. Volkl *et al.*, «k4FWCore: Preliminary Initial Release: Core Components for the Gaudi-based Key4hep Framework», https://doi.org/10.5281/zenodo.4564605

[18] V. Volkl *et al.*, «k4Gen: Preliminary Initial Release», https://doi.org/10.5281/zenodo.4564609

[19] V. Volkl *et al.*, «k4SimGeant4: v0.1.0pre02 Initial Release; Gaudi Components for Geant4 Simulation in the Key4hep software framework.», https://doi.org/10.5281/zenodo.4564574

[20] V. Volkl *et al.*, «k4RecCalorimeter: Preliminary Initial Release», https://doi.org/10.5281/zenodo.4564669

[21] F. Gaede, «Marlin and LCCD — Software tools for the ILC», *Nucl. Instrum. Methods Phys. Res. A* **559**, 177 (2006).

[22] Key4hep GitHub Repository, «k4MarlinWrapper», https://github.com/key4hep/k4MarlinWrapper

[23] https://cmake.org/

[24] T. Gamblin *et al.*, «The Spack package manager: bringing order to HPC software chaos», in: «SC15: International Conference for High-Performance Computing, Networking, Storage and Analysis», *IEEE Computer Society*, Los Alamitos, CA, USA 2015, pp. 1–12.