The Key4hep turnkey software stack for future colliders

Placido Fernandez Declara,[a,*] Gerardo Ganis,[a] Benedikt Hegner,[a] Clement Helsens,[a] Marko Petric,[a] Andre Sailer,[a] Valentin Volkl,[a] Frank Gaede,[b] Thomas Madlener,[b] Wenxing Fang,[c] Weidong Li,[c] Tao Lin,[c] Xiaomei Zhang,[c] Jiaheng Zou,[c] Xingtao Huang,[d] Teng Li,[d] Sang Hyun Ko[e] and Joseph Wang[f]

[a] *CERN, Geneva, Switzerland*
[b] *DESY, Hamburg, Germany*
[c] *IHEP, Bejing, China*
[d] *Shandong University, Qindao, Shandong, China*
[e] *Seoul National University, Seoul, Republic of Korea*
[f] *Bitquant Digital Services, Hong Kong*

E-mail: placido.fernandez@cern.ch

High Energy Physics experiments for the future depend on reliable, proven, and flexible software components to perform detector optimisation and physics performance studies. Key4hep is a software stack designed to provide a set of software components for future and present High Energy Physics projects. In these proceedings the main components of Key4hep are described, comprising: EDM4hep for the common event data model, simulation interfaces for both Delphes and Geant4, k4MarlinWrapper to integrate with various iLCSoft components, and software tools to build, validate and integrate the different components of Key4hep seamlessly. It provides a software infrastructure for easy access and use of its components, where different future collider projects are adapting to use Key4hep, such as FCC, CEPC, ILC, CLIC and SCT. These adaptations showcase the feasibility of the software stack as a turnkey solution for nascent physics studies and as a base for future High Energy Physics experiments.

**** The European Physical Society Conference on High Energy Physics (EPS-HEP2021), ****
**** 26-30 July 2021 ****
**** Online conference, jointly organized by Universität Hamburg and the research center DESY ****

[*]Speaker

1. Introduction

Future High Energy Physics detector studies rely on well maintained software to perform proper studies of detector concepts, and test their physics reach and limitations. The Key4hep project is designed to create and provide a turnkey software stack for this purpose by providing a set of software components, documentation and the interfaces to connect them. It is designed to meet the needs of present and future experiments which have long lifetimes; it connects and extends existing software packages that range from event generation to simulation and reconstruction into a data processing framework. To facilitate creating a project with Key4hep, the software stack has to be fully functional and provide a variety of examples to allow users to adapt it to their specific use cases. These proceedings present the progress made since previous conferences [1–4].

Key4hep is a community driven project with contributions and adaptations from different future collider projects that are under consideration: CEPC, FCC-ee, FCC-hh, ILC, CLIC and SCT. Key4hep is open source, its packages are hosted on GitHub [5]. Documentation is provided [6] including instructions on how to use the existing installations locally, from a Virtual Machine or using Docker provided access to CVMFS, how to build the complete software stack using Spack [7] including guidelines to use Spack, and how to run simulation and reconstruction jobs with examples. Documentation also includes contribution guidelines to setup a project and write algorithms for it. Alternating bi-weekly meetings are held to discuss progress, issues and updates for Key4hep and the Event Data Model (EDM) EDM4hep, which is briefly discussed in Section 2.

The detector description toolkit DD4hep is discussed in Section 3, the common framework used for the project, Gaudi, is discussed in Section 4. A wrapper to use iLCSoft processors in an integrated manner is described in Section 5. Simulation components of Key4hep are discussed in Section 6 with k4SimDelphes. The core framework, Spack, and available infrastructure are discussed in Section 7. Finally the ongoing work and progress of various adaptations for future experiments is discussed in Section 8, before concluding with a summary and outlook in Section 9.

2. EDM4hep

The EDM describes the event data entities used in the different components of the framework, defines the interface to access and manipulate them, and provides the means to persist the event data. The common EDM for the Key4hep project, EDM4hep [8], is based on the experience that has been gathered with the battle-proven LCIO EDM [9], and FCC-edm [10]. LCIO has been used by the linear collider community for nearly 20 years with great success. EDM4hep is implemented with the PODIO EDM toolkit [4, 11, 12].

Figure 1 shows a schematic view of the main data types found in EDM4hep. It is organized to differentiate between different steps in a typical HEP procesing chain: from data types for simulation with *MCParticles* at the center, to reconstructed data types with the *ReconstructedParticles* at the end of the chain, with raw and digitized hits for the intermediate steps of the chain. The relations between these data types is featured to expose the hierarchy of the different types: higher level types always point back to their constituent object.

The present EDM4hep description attends to the needs of precision future lepton colliders, aiming to reconstruct all individual particles resulting from a collision as in a Particle Flow approach.

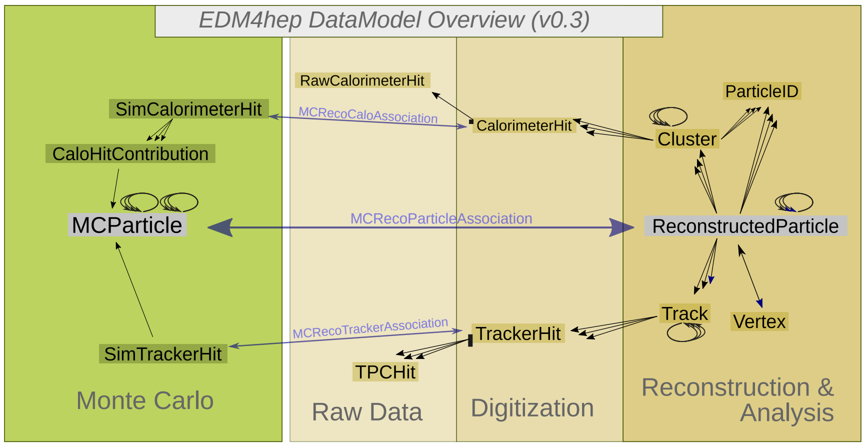

Figure 1: Schematic view of the contents of EDM4hep and the relations among the different data types that it defines.

Attending to the Key4hep principles, EDM4hep aims to support future hadron colliders with more complex resulting collisions, for which it should be equally prepared.

3. DD4hep

For the detector description Key4hep uses the DD4hep [13, 14] toolkit. It was designed to be a good fit for most HEP experiments; initially developed for the linear collider community. It provides the means to describe a complete detector including geometry, materials, visualization, readout, alignment, calibration and more. It provides these from a single source of information so it can be easily used from simulation, reconstruction and analysis with a common interface.

It is widely used in the HEP community including ILC, CLIC, FCC, CEPC or IEC, with LHCb and CMS integrating to use it for Run3 [15, 16]. It provides simulation of detector response with *ddsim* [17], including a plugin to produce output in EDM4hep format developed for Key4hep. DD4hep already includes integration with Geant4 [18] for full detector simulation, and there is ongoing work to integrate it with the rest of the framework.

4. Framework

Traditionally in High Energy Physics, experiment frameworks have been developed individually by each experiment, with little shared efforts towards getting a common software framework or shared pieces of the framework. Notable exceptions are: Marlin [19] which has been successfully developed and used by CLIC and ILC in the linear collider community; and Gaudi which was originally developed by the LHCb experiment, and it is now developed and used also by ATLAS. Gaudi is also used by FCCSW, HARP, Minerva and other experiments. Key4hep has adopted Gaudi [20] as its base common experiment framework after its wider adoption in the HEP community and successful use in some LHC experiments. Several Key4hep components have been now developed using Gaudi such as k4MarlinWrapper [21], k4FWCore [22], k4SimDelphes [23] or k4SimGeant4 [24], and the project is contributing to its development were necessary. To allow for iLCSoft algorithms to be used in Key4hep, a wrapper has been developed which enables to run Marlin processors within the Gaudi framework as described in Section 5.

The k4FWCore component provides the primary components needed to build a package in Key4hep, and to communicate and integrate with the other components. It contains functionality to manage input and output of EDMs generated via PODIO, such as EDM4hep. It also includes tools for background overlaying and other functionality.

5. k4MarlinWrapper

The k4MarlinWrapper package was developed by Key4hep to provide the means to run and integrate Marlin processors with the other Key4hep components: to run *Marlin* processors through Gaudi algorithms. By producing a wrapper, Marlin functionality and source code is kept in its original form, benefiting from the experience gained for more than 15 years in the linear collider community: it enables all the dedicated reconstruction and analysis software in Key4hep through the wrapper.

k4MarlinWrapper provides the components and converters to interface between Marlin and Gaudi. A converter between the different formats for the steering files was developed: from Marlin XML to Gaudi Python. It now includes features to support optional execution markers and constants parsing that are propagated through the file. It supports input and output in both LCIO and EDM4hep format to read and write events.

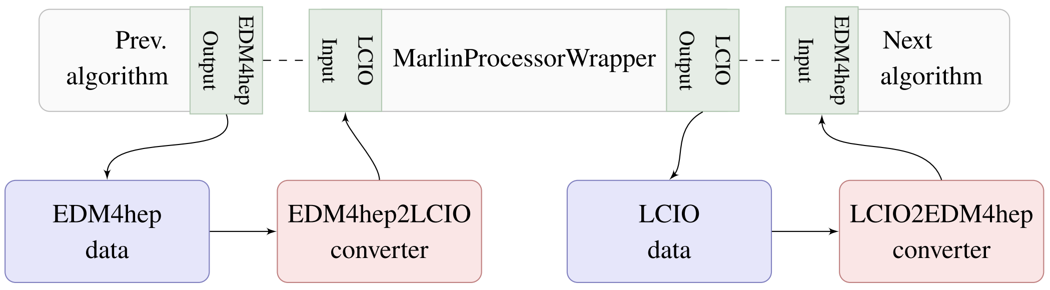

Figure 2: k4MarlinWrapper converters in between algorithms with different input and output EDM formats.

It includes in-memory on-the-fly converters for the supported EDMs: LCIO and EDM4hep, to enable running Marlin processors and integrate them with present and future algorithms that will use EDM4hep format. Marlin processors will only input and output LCIO format, so a converter from and to EDM4hep can be attached to any k4MarlinWrapper algorithm for proper interfacing as can be seen on Figure 2. Converters are implemented as Gaudi tools that can be attached to any wrapper indicating the collections to be converted by their names. It supports conversion for metadata in different collections.

k4MarlinWrapper runs the full CLIC reconstruction of processors and has been successfully used to integrate and run the LCFI+ algorithm [25] for vertex and jet finding, and flavour tagging by using the EDM converters in the sequence. There are undergoing efforts to integrate the k4MarlinWrapper with the SCT experiment into the *Aurora* framework.

6. k4SimDelphes

Delphes is a fast simulation framework [26] which is integrated within Key4hep through k4SimDelphes [23]. Delphes performs a fast multipurpose detector response simulation including tracking with magnetic field, and calorimeter and muon systems. For the input and output system, it is compatible with standard input file formats such as *Les Houches Event Files* or *HepMC* format. It can output a range of types including isolated leptons, jets or missing transverse energy, which can be used for dedicated analysis; the output is converted to EDM4hep format to be compatible and standardized in Key4hep through k4SimDelphes.

Currently k4SimDelphes offers standalone executables that cover all the input formats that Delphes also supports. Additionally an interface with EvtGen [27] has been developed.

As mentioned in Section 4, k4SimDelphes is undergoing an integration into the Gaudi framework to offer the same unified framework interface as in Key4hep. This eases the use of different simulation modules and interchange them while producing a similar output. k4SimDelphes aims to provide a coherent approach to generation and simulation.

7. Software infrastructure

To manage the compilation and deployment of the software stack in an integrated manner, Key4hep offers a series of tools, guidelines and templates. HEP experiments handle a large number of packages that need to be well integrated while being flexible in adding newly developed packages.

Key4hep offers a centralized location for its documentation which includes guidelines and best practices to produce a consistent and homogeneous software stack. It encourages the use of modern software tooling and practices: it uses automated builds with stable and nightly builds that include continuous integration tests to make deployments robust and catch issues early. Key4hep has two main programming languages: C++ and Python. For these it encourages the use of modern CMake [28] and common testing frameworks with Catch2 and Pytest respectively.

For the compilation and deployment of the software stack the Spack package manager is used. Spack was originally developed for the High Performance Computing (HPC) community, were dealing with multiple configurations of the same package is needed, and it is a good fit for scientific communities that rely on complex software. It is operating system and build system independent and uses Python to define the builds. Spack handles the dependencies between packages and allows Key4hep to have a streamlined deployment process that follows the paradigm of releasing early and often. The software stack is deployed through CVMFS after being built with Spack.

8. Key4hep adaptations in different communities

Different communities are undergoing or have already integrated with Key4hep or some of its components. CEPC started using iLCSoft components from the beginning but it adopted components like the Gaudi framework, EDM4hep for its EDM, and DD4hep for describing the detector. Adopting those components made it easier to integrate with Key4hep, and CEPC also contributed to Key4hep with the Geant4 based simulation and implemented the k4LCIOReader to be able to move from LCIO to EDM4hep. The resulting software stack, CEPCSW is fully integrated with Key4hep.

FCCSW is the experiment software for the FCC design studies. From its inception it used the Gaudi framework and DD4hep to build its software, and it adapted to use EDM4hep format from the FCC-EDM. Both the original FCC-EDM and EDM4hep use PODIO to generate their event data model. The structure of FCCSW has been changed to split into components that fit a generic purpose inside Key4hep: k4FWCore, k4Gen, k4SimDelphes, k4SimGeant4, k4RecCalorimeter, fccDetectors and dual-readout.

The iLCSoft software stack with the ILC and CLIC communities was developed with great success over the past 15 years. It used DD4hep to develop detailed detector models, and implemented reconstruction and digitization algorithms for the Marlin framework using the LCIO EDM. These communities need to keep using the components developed over the past years, which is enabled via the k4MarlinWrapper described in Section 5. Interfaces and converters between the different formats and EDMs enable a smooth transition for both communities while integrating and benefiting from other Key4hep components.

The SCT experiment is building its software stack around the Aurora framework. It uses a PODIO based EDM and Gaudi based framework to develop its algorithms. These similarities with the Key4hep project and integration with k4MarlinWrapper is undergoing to benefit from the already developed algorithms in the linear collider communities.

9. Conclusions

Key4hep aims to provide a complete software stack for future HEP experiments. All main future collider experiments, CEPC, CLIC, FCC and ILC, contribute to Key4hep and have started migrating or have migrations plans to integrate Key4hep into their software stacks. This includes key components such as the Event Data Model with EDM4hep, detector description with DD4hep, a common base framework with Gaudi, a wrapper to integrate with existing software from the linear collider community with k4MarlinWrapper, and an interface to use Delphes with converters for consistent EDM output with k4SimDelphes. Key4hep also developed the software infrastructure to build and deploy all software components in a uniform manner, and includes guidelines for homogeneous practices with respect to software tooling. Different communities have made progress in integrating with Key4hep and contributed back to it. Namely the CEPC community is fully integrated with Key4hep and has contributed several packages to it. FCCSW started its project with several components which are already shared with Key4hep and has undergone a restructure of its components to adapt and contribute to Key4hep. iLCSoft follows a different approach to integrate with it by adopting a set of interfaces and converters that allow to seamlessly use the more than 15 years of experience gained in the linear collider communities. Future experiments like SCT already share many similarities with Key4hep and are evaluating and adapting their components to benefit from the common approach followed by other future experiments.

The Key4hep group is actively developing and improving the turnkey software stack, supporting the future collider experiments that adopt it, and welcome current, future and new users and HEP experiments to collaborate with the project.

Acknowledgments

This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (https://cds.cern.ch/record/2649646/, CERN-OPEN-2018-006). This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101004761.

References

[1] P. Fernandez Declara et al., *Key4hep: Status and Plans*, *EPJ Web Conf.* **251** (2021) 03025.

[2] F. Gaede, G. Ganis, B. Hegner, C. Helsens, T. Madlener, A. Sailer et al., *EDM4hep and podio - The event data model of the Key4hep project and its implementation*, *EPJ Web Conf.* **251** (2021) 03026.

[3] A. Sailer, G. Ganis, P. Mato, M. Petrič and G.A. Stewart, *Towards a Turnkey Software Stack for HEP Experiments*, *EPJ Web Conf.* **245** (2020) 10002.

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

[5] "Key4hep github repository." `https://github.com/key4hep`, 2021.

[6] Key4hep Software Group et al., *key4hep-doc: Documentation for key4hep-software*, Feb., 2021. 10.5281/zenodo.4565650.

[7] V. Volkl, T. Madlener, T. Lin, J. Wang, D. Konstantinov, I. Razumov et al., *Building HEP Software with Spack: Experiences from Pilot Builds for Key4hep and Outlook for LCG Releases*, *EPJ Web Conf.* **251** (2021) 03056.

[8] "EDM4hep github repository." `https://github.com/key4hep/EDM4hep`, 2021.

[9] F. Gaede, T. Behnke, N. Graf and T. Johnson, *LCIO: A Persistency framework for linear collider simulation studies*, *eConf* **C0303241** (2003) TUKT001 [`physics/0306114`].

[10] "Fcc-edm github repository." `https://github.com/HEP-FCC/fcc-edm`.

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

[12] "PODIO github repository." `https://github.com/AIDASoft/podio`, 2021.

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

[14] M. Frank, F. Gaede, M. Petric and A. Sailer, *Aidasoft/dd4hep: v01-18*, Sept., 2021. 10.5281/zenodo.5494511.

[15] Ivanchenko, Vladimir, Banerjee, Sunanda, Hugo, Gabrielle, Lo Meo, Sergio, Osborne, Ianna, Pedro, Kevin et al., *Cms full simulation for run 3*, *EPJ Web Conf.* **251** (2021) 03016.

[16] B.G. Siddi and D. Müller, *Gaussino - a gaudi-based core simulation framework*, .

[17] M. Petric, M. Frank, F. Gaede, S. Lu, N. Nikiforou and A. Sailer, *Detector simulations with dd4hep*, *J. Phys. Conf. Ser.* **898** (2016) 042015.

[18] S. Agostinelli et al., *GEANT4 - A Simulation Toolkit*, .

[19] F. Gaede, *Marlin and LCCD: Software tools for the ILC*, *Nucl. Instrum. Meth. A* **559** (2006) 177.

[20] G. Barrand et al., *GAUDI - A software architecture and framework for building HEP data processing applications*, *Comput. Phys. Commun.* **140** (2001) 45.

[21] P.F. Declara, A. Sailer, M. Petric and V. Volkl, *key4hep/k4marlinwrapper: v00-03-01*, Apr., 2021. 10.5281/zenodo.4719245.

[22] V. Volkl, J. Pöttgen, hegner, zaborowska, J. Cervantes, C. Helsens et al., *key4hep/k4fwcore:*, July, 2021. 10.5281/zenodo.5109722.

[23] V. Volkl, T. Madlener, C. Helsens, M. Chrząszcz and F. Gaede, *key4hep/k4SimDelphes: v00-01-06: More Evtgen / Test Fixes*, May, 2021. 10.5281/zenodo.4748578.

[24] zaborowska, J. Pöttgen, V. Volkl, M. Selvaggi, Z. Drasal, faltovaj et al., *Hep-fcc/k4simgeant4:*, July, 2021. 10.5281/zenodo.5109724.

[25] T. Suehara and T. Tanabe, *LCFIPlus: A Framework for Jet Analysis in Linear Collider Studies*, *Nucl. Instrum. Meth. A* **808** (2016) 109 [`1506.08371`].

[26] DELPHES 3 collaboration, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, *JHEP* **02** (2014) 057 [`1307.6346`].

[27] “Evtgen.” `https://evtgen.hepforge.org/`, 2021.

[28] “Cmake.” `https://cmake.org/`, 2021.