Grant Agreement No: 101004761

AIDAinnova

Advancement and Innovation for Detectors at Accelerators

Horizon 2020 Research Infrastructures project AIDAINNOVA

MILESTONE REPORT

LC RECONSTRUCTION PROTOTYPE IN KEY4HEP

MILESTONE: MS47

Document identifier:	AIDAinnova-MS47
Due date of milestone:	End of Month 21 (December 2022)
Report release date:	19/01/2023
Work package:	WP12: Software for Future Detectors
Lead beneficiary:	DESY
Document status:	Final

Abstract:

The iLCSoft software stack, developed originally by the linear collider communities, is part of the Key4hep software stack. While iLCSoft comes with its own event processing framework in the form of Marlin, Key4hep has decided to adopt Gaudi as its main event processing framework. In order to make it possible to benefit from the almost two decades of development work that has already gone into functionality provided by iLCSoft, a Marlin wrapper has been developed that allows Marlin Processors to run inside the Gaudi based Key4hep framework unchanged. This milestone demonstrates the ability to run existing reconstruction and analysis chains from iLCSoft entirely inside the Gaudi ecosystem.

AIDAinnova Consortium, 2023
For more information on AIDAinnova, its partners and contributors please see http://aidainnova.web.cern.ch/

The Advancement and Innovation for Detectors at Accelerators (AIDAinnova) project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 101004761. AIDAinnova began in April 2021 and will run for 4 years.

Delivery Slip

	Name	Partner	Date
Authored by	T. Madlener [Task coordinator]	DESY	13/12/2022
Edited by	Paolo Giacomelli	INFN	20/12/2022
Reviewed by	F. Gaede [WP coordinator] G.Stewart [WP coordinator]	DESY CERN	15/12/2022
Approved by	Paolo Giacomelli [Scientific coordinator]		19/01/2023

TABLE OF CONTENTS

Executive summary

The possibility to run reconstruction software originally developed for linear collider studies in the Key4hep ecosystem is crucial. This milestone's main goal is to demonstrate interoperability of the involved components as seamlessly as possible.

The main developments to enable this interoperability are a Marlin processor wrapper that allows to run Marlin processors unchanged in the Gaudi based Key4hep framework, as well as necessary event data model converters that can convert between the two event data models, LCIO and EDM4hep, on the fly where necessary.

To validate that all of the newly developed components work correctly and as expected, they have been checked and validated by running existing standard reconstruction and analysis chains from ILD and CLIC using the Marlin framework as a baseline and comparing that to running the same workflows using the Marlin processor wrapper in the Gaudi based Key4hep framework. In both cases the results agree excellently, thus, confirming that all involved software works as intended. To keep this working state intact in light of future developments, the CLIC reconstruction is run as part of the Key4hep test suite that is run regularly during nightly builds.

1. INTRODUCTION

The Key4hep project aims to provide a common and shared software stack for all Future Collider projects[1]. It receives regular contributions from all of the involved collaborations, mainly the ILC, CLIC, FCC and CEPC communities. Among others, two key components of Key4hep are the common and shared event data model (EDM), EDM4hep[2], and a common event processing framework, where Key4hep has adopted Gaudi.

The linear collider communities have established their software stack, iLCSoft, over the last 15 years. Given the amount of work that has gone into this effort, and the fact that it brings full-fledged reconstruction and analysis chains for detailed detector studies to the table, iLCSoft has been included into the Key4hep ecosystem from the beginning. Since iLCSoft brings its own event data model and processing framework in the form of LCIO and Marlin, interoperability of the existing processors written for Marlin and the Gaudi based software components of Key4hep does not come for free.

The scope of this milestone is enabling this interoperability as seamlessly as possible for the involved communities. By doing so, this opens an avenue for gradually integrating existing software more tightly into the Key4hep ecosystem, but also to migrate from one world to the other. In order to qualify as seamless, no changes to the existing processors should be necessary, and all required conversions from one EDM into the other should be handled automatically. All of these points are addressed by the k4MarlinWrapper components that have been developed to achieve this milestone.

This report will first introduce the necessary software components in Section 2, before giving a brief summary of how these components were validated to ensure that physics results remain stable when switching from the iLCSoft to the Key4hep ecosystem.

2. K4MARLINWRAPPER

The main purpose of an event processing framework is to schedule the execution of dedicated steps in an execution chain, as well as marshalling data between these steps. Depending on the framework these steps are referred to either as *Processors* (Marlin) or *Algorithms* (Gaudi), and they usually perform a small piece of reconstruction, e.g., digitizing simulated hits, or doing pattern recognition for tracking, before handing off to a next step, where the outputs of previous steps are used. The order and configurable parameters of these Processors/Algorithms are set at runtime. Both Gaudi and Marlin use C++ as the language in which the Algorithms or Processors are implemented; however, while Gaudi uses Python as the runtime configuration language, Marlin uses XML. Probably the biggest difference between the Marlin and the Gaudi framework are the capabilities of the transient event store (TES). While Gaudi can effectively handle arbitrary data that can be passed between different Algorithms, Marlin allows for transport of data via the LCIO data format only. In the following we discuss how these similarities are leveraged and how the differences are overcome to make it possible to run Marlin Processors unchanged in the Gaudi framework.

2.1. MARLINPROCESSORWRAPPER ALGORITHM

The main components of the k4MarlinWrapper are two Gaudi Algorithms: one that reads full events from LCIO files and puts them into Gaudi's TES, and a second one that is able to run arbitrary Marlin Processors, feeding them with data in the right format from the TES. The latter is called *MarlinProcessorWrapper*, and apart from instantiating and executing the correct Marlin Processor it also has to take care of some additional duties, most importantly configuring the wrapped Processor correctly using the parameters that are provided by the user. Additionally, it also provides the necessary hooks for EDM converters (see Section 2.3), takes care of correctly handling the random seed settings, as well as setting up all the necessary logging facilities.

2.2. STEERING FILE CONVERSION SCRIPT

A typical Marlin steering file in XML format for a full reconstruction and analysis chain contains of the order of roughly 100 Processors that are configured within it, resulting in files having a few thousand lines of code. Since Gaudi uses Python as its runtime configuration language all of the existing chains have to be converted. To avoid an error prone and tedious manual process, a converter script has been developed, taking as input a Marlin steering file and producing as output the corresponding Gaudi python options file. This script takes care of creating the necessary MarlinProcessorWrapper algorithms, wrapping one original Processor each, as well as of converting the parameters specified in the XML file into a format suitable for the MarlinProcessorWrapper. The converter script also handles so called *constants*, a powerful feature of Marlin that allows the definition of configuration parameters that depend on other parameters, such that they remain easily configurable in the produced Python options file. At runtime, the dependencies between different constants are recursively resolved before they are used as parameters. Finally, the converter script also handles conditional execution of wrapped processors as far as possible.

2.3. EVENT DATA MODEL CONVERTERS

Simply running an existing chain does not, in principle, require any conversions between the two involved EDMs (LCIO and EDM4hep), as the whole chain can be executed using the LCIO format, regardless of whether the Marlin or the Gaudi framework is used to run it. However, in order to allow interoperability with newly developed reconstruction and analysis algorithms, which are working with EDM4hep, it is necessary to convert from LCIO to EDM4hep and vice versa. To achieve this, EDM converter tools have been developed, each taking care of one direction of the conversion. They

can be hooked into individual MarlinProcessorWrapper algorithms, such that it is possible for the user to configure, for each of the wrapped processors, which EDM4hep inputs should be converted to LCIO before the processor is run, and which LCIO outputs should be converted to EDM4hep after the processor is run. This setup makes it possible to convert between the two EDMs on the fly as necessary. An illustration of this process is shown in Figure 1.

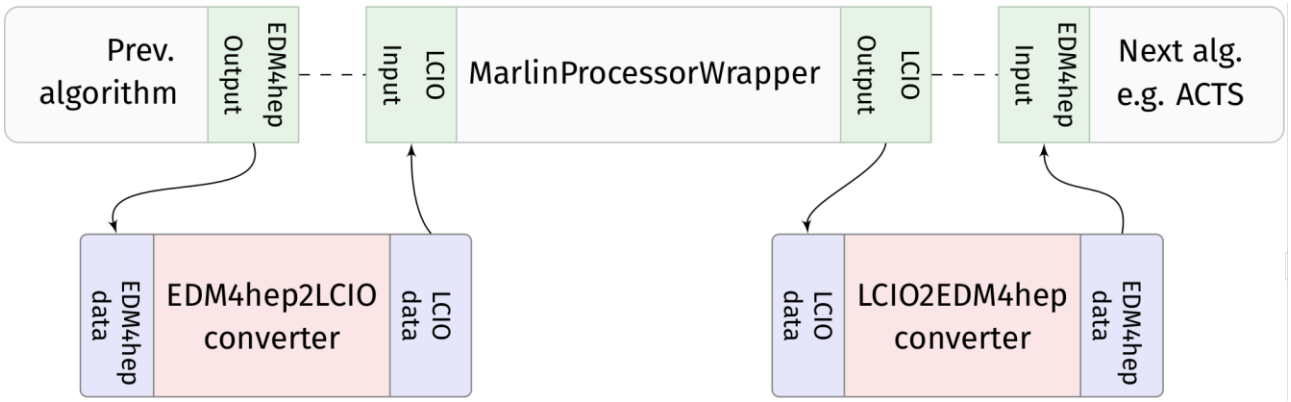

Fig. 1 Flow diagram showing how a wrapped Marlin Processor that expects LCIO input and produces LCIO output can be used in conjunction with a Gaudi Algorithm that produces and expects EDM4hep inputs and outputs with the help of the EDM converter tools.

3. RESULTS

To verify that all of the components described above are working as expected, several validation tests have been carried out. These will now be described in more detail. As discussed earlier, the main goal is to be able to run existing reconstruction and analysis chains without having to touch any of the existing code of the involved Processors. In general, the validation and testing process looks like this:

1. Run the existing chain with Marlin to get a baseline result to compare to.
2. Convert the Marlin XML steering file to a Gaudi Python options file using the converter script described above.
3. Run the Gaudi Python options file using the Key4hep framework to produce outputs.
4. Compare the outputs from step 1 with the ones from step 3.

3.1. ILD RECONSTRUCTION

The ILD reconstruction and analysis chain has been tested and validated using a small physics sample consisting of simulated Higgs-Strahlung events at a 250 GeV centre of mass energy, where the Higgs mass is reconstructed via the recoil against the di-muon decay of the Z boson. This test setup not only includes the complete reconstruction chain, but also executes various high level analysis processors, e.g., for identifying isolated muons. These latter processors are typically run by users rather than in central production workflows. Hence, this validation also covers existing software with potentially slightly less strict quality requirements than is typically found in the processors that are run centrally.

Figure 2 shows a comparison of the analysis results obtained via Marlin and the ones obtained using the Key4hep Gaudi framework and the k4MarlinWrapper components. As can be seen, there is excellent agreement between the resulting distributions of the Higgs recoil mass, as well as the Z mass obtained from combining the two muons. The residual differences have been traced back to

differences in random number generation. For this validation, no EDM conversions where used, as all inputs and outputs were in LCIO format.

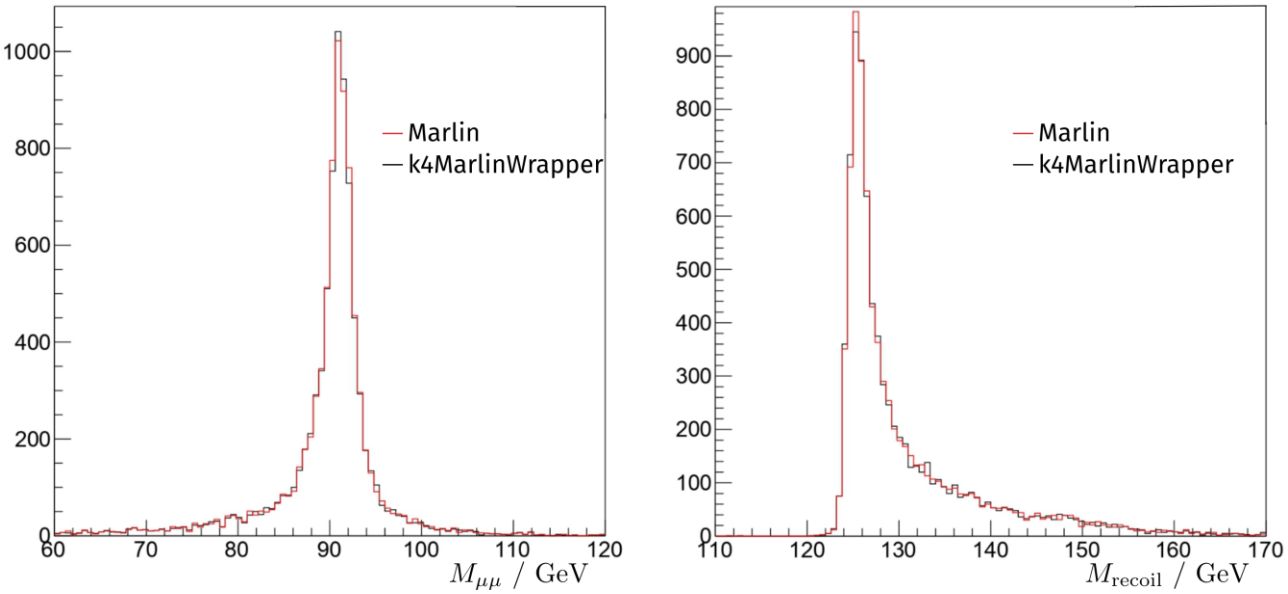

Fig. 2 Comparison of the reconstructed dimuon mass (left) and the Higgs recoil mass (right) when running the complete ILD reconstruction and analysis chain via Marlin and via the Key4hep framework using k4MarlinWrapper components.

3.2. CLIC RECONSTRUCTION

Although the CLIC and ILD reconstruction chains have quite some overlap from a software point of view, there are still significant differences, most notably the tracking. Hence, in order to ensure that all reconstruction processors that are in use work as expected, a similar exercise as for the ILD reconstruction described above has been done using the standard CLIC reconstruction chain. Also here excellent agreement has been found as is shown in Fig 3, which shows the total reconstructed energy from Z→qq (uds) events at different centre-of-mass energies. The reconstruction using the Marlin framework has been done using a CLIC iLCSoft release from a previous simulation and reconstruction campaign, while the reconstruction via the k4MarlinWrapper uses a current version. For the latter a proper calibration of the Pandora PFA has not yet been done, which explains the minor discrepancies that are visible. However, even without that it is evident that the Pandora PFA Clustering performance is very close to what has been achieved in the past.

In order to ensure that future developments on the k4MarlinWrapper components and the EDM converters do not introduce any regressions, the CLIC reconstruction is also run as part of the continuous integration test suite, that is run with every new commit to the k4MarlinWrapper repository. In this scenario simulated ttbar events are used, as these generally lead to rather busy events and in this way ensure that all processors are always run. Additionally, the inputs and outputs for this case use the EDM4hep format, thus making it necessary to run several EDM converters attached to the processors in the chain. Hence, the functionality of the linear collider reconstruction chain in the Key4hep ecosystem is continuously tested, at least on a technical level. Work to also run a basic physics validation in this setup is currently being planned.

Finally, the CLIC reconstruction chain is also used in studies for calorimeter clustering using GPUs in the context of k4CLUE[3]. Here it serves as the main platform for studying different clustering

schemes and how they affect the physics results. In fact this use case could already be described as "production" use of the LC reconstruction inside Key4hep.

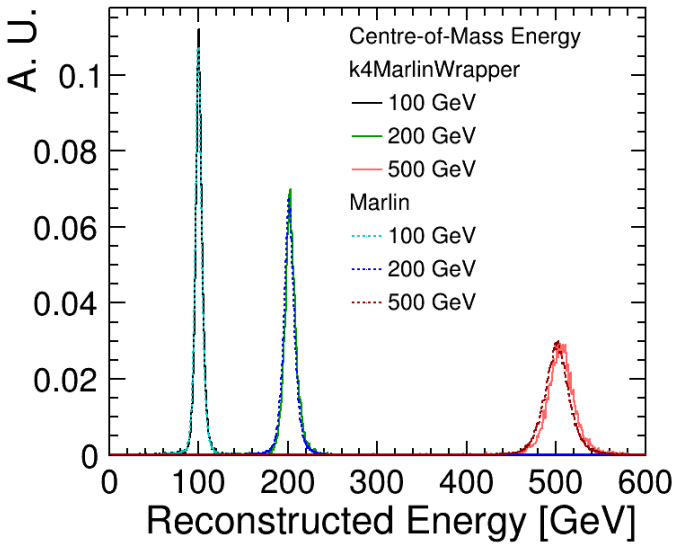

Fig 3 Comparison of the reconstructed total energy from Z->qq (uds) events at different centre-of-mass energies reconstructed once via Marlin (dashed lines) and once via the k4MarlinWrapper (solid lines).

4. CONCLUSIONS

The main goal of this milestone is to demonstrate the possibility to run existing reconstruction workflows from the linear collider communities within the Key4hep ecosystem without changing existing software. This has been achieved with the development of the k4MarlinWrapper components, with the MarlinProcessorWrapper and EDM conversion tools as central pieces. The correct functionality of the involved components has been validated using the ILD and CLIC reconstruction and analysis chains and comparing their results with results obtained from running the same chains using Marlin. In both cases excellent agreement has been found. The CLIC reconstruction is also routinely run as part of continuous integration tests for the k4MarlinWrapper repository, including several intermediate conversions between the two involved EDM formats. In conclusion: All necessary parts to run the LC reconstruction inside the Key4hep framework are present, including the necessary building blocks to integrate new developments seamlessly into existing workflows.

5. REFERENCES

CONFERENCE PAPERS

[1] Fernandez Declara, P., et al. (2022) The Key4hep turnkey software stack for future colliders. In *Proceedings of Science (PoS) Volume 398 – The European Physical Society Conference on High Energy Physics (EPS-HEP2021)*, 844 https://pos.sissa.it/398/844

[2] Gaede, F., et. al. EDM4hep – a common event data model for HEP experiments. In *Proceedings of Science (PoS) Volume 414 - 41st International Conference on High Energy physics (ICHEP2022)*, 1237 https://pos.sissa.it/414/1237

REPORTS

[3] Aglieri Rinella, G., et al. (2022) *Strategic R&D Programme on Technologies for Future Experiments – Annual Report 2021*, CERN, CERN-EP-RDET-2022-006, p. 111 - 113 https://cdsweb.cern.ch/record/2808204?

ANNEX: GLOSSARY

Acronym	Definition
EDM	Event Data Model
ILC	International Linear Collider
CLIC	Compact Linear Collider
FCC	Future Circular Collider
CEPC	Circular Electron-Positron Collider
LCIO	Linear Collider Input/Output
TES	Transient Event Store