EIC ATHENA Documentation

Release 0.00.02

Yulia Furletova, Dmitry Romanov

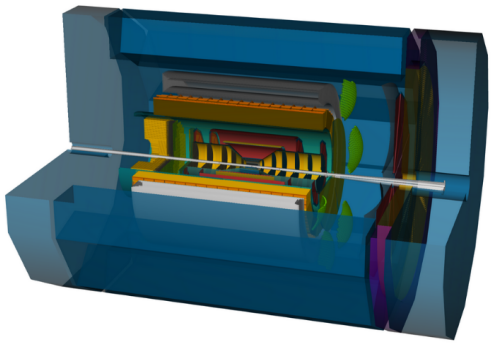
Jun 04, 2022

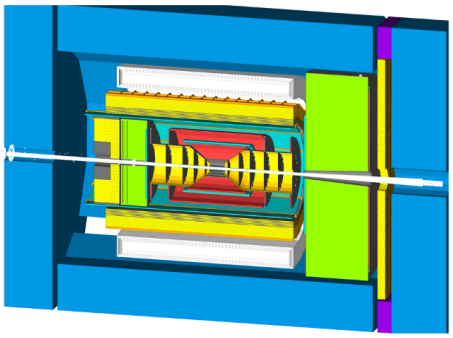
GETTING STARTED:

CHAPTER

ONE

ATHENA SOFTWARE PLAN





1.1 Philosophy

Let's prepare for our future at the EIC!

Software and computing will be critical to the success of any EIC experiment. Our strategy to acomplish this and ensure the long-term success is by building a forward-looking team of developers. We focus on modern scientific computing practices: modularity, orthogonal components designed for performance in heterogeneous computing environments. Our emphasis is on modern development practices built around continuous-integration, reproducible containerization and automated tests and benchmarks. At the same time, we try to choose mature, well-supported, and actively developed software components allowing us to focus our resources on those parts of the toolkit requiring custom development work.

Summarizing:

- Build forward-looking team of developers to ensure the long-term success of the EIC scientific program in software & computing.
- **Focus on modern scientific computing practices**
  - Strong emphasis on modular orthogonal tools
  - Integration with HTC/HPC, CI workflows, and enable use of data-science toolkits
- Avoid "not-invented-here" syndrome, and instead leverage cutting-edge CERN-supported software components where possible
- Build on top of mature, well-supported, and actively developed software stack.
- Externalize support burden where possibl
- Actively work with the EICUG SWG to help develop and integrate community tools for all collaborations.

CHAPTER

TWO

SOFTWARE OVERVIEW

2.1 Full Simulation

The software is currently being developed. Here is only the software short description with links. Please read the full desctiption next.

The main repository for ATHENA software is here

2.1.1 DD4Hep

DD4hep is a software framework for providing a complete solution for full detector description (geometry, materials, visualization, readout, alignment, calibration, etc.) for the full experiment life cycle (detector concept development, detector optimization, construction, operation). It offers a consistent description through a single source of detector information for simulation, reconstruction, analysis, etc.

- DD4Hep main repository
- EIC detector implementations

2.1.2 Juggler

Gaudi based digitization and reconstruction framework. Currently it supports ACTS based tracking, calorimetry reconstruction, some PID like RICH

- Project Juggler
- EICD - EIC event data model implemented in PODIO

2.2 Validation

On any commit to detector or reconstruction repository a bunch of validation benchmarks are being automatically run. Current benchmarks repository

- Detector benchmarks - Detector benchmarks are meant to test for regressions in individual detector subsystems without a complex reconstruction
- Reconstruction benchmarks - Reconstruction specific benchmarks (calorimetry, tracking, PID)
- Physics benchmarks - Check physical processes such as DIS, DVCS, etc.
- Cost estimation - (in develoment) A weighted evaluation of the detector performance versus estimated cost.

2.3 Infrastructure

- Main repository
- Rucio
- Farms

The resources page on the EICUG WG site

2.4 Fast simulation

2.4.1 Delphes

DELPHES provides the simulation of a multipurpose detector for phenomenological studies. The simulation includes a track propagation system embedded in a magnetic field, electromagnetic and hadron calorimeters, and PID. Physics objects that can be used for data analysis are then reconstructed from the simulated detector response. These include tracks and calorimeter deposits and high level objects such as isolated electrons, jets, taus, and missing energy. With features such as the particle-flow reconstruction, pile-up simulation and mitigation.

Contacts: Stephen Sekula, Miguel Arratia

Support line: ATHENA Slack

- Delphes web site
- EIC Reference Detector Delphes implementation

2.4.2 EicSmear

Eic-smear is a Monte Carlo analysis package originally developed by the BNL EIC task force. It contains classes and routines for building events in a C++ object and writing them to a ROOT file in a tree data structure, performing fast detector smearing on those Monte Carlo events. The smearing portion of the code provides a collection of classes and routines that allow simple parameterizations of detector performance, provided via a ROOT script, to be applied to any of the above Monte Carlo events. Detector acceptance can also be defined.

Contacts: Alexander Kiselev ayk@bnl.gov, Kolja Kauder kkauder@bnl.gov

- EIC-Smear main package
- EIC detector parametrizations

2.5 Legacy full simulation

2.5.1 ESCalate

Contacts: Dmitry Romanov, Yulia Furletova (g4e)

- escalate.readthedocs.io
- g4e.readthedocs.io

2.5.2 Fun4All

Contacts: Kolja Kauder (for ATHENA), Chris Pinkenburg and Jin Huang (developers)

- Fun4All general Tutorials https://github.com/eic/fun4all_tutorials
- Fun4All EIC specific Tutorials https://github.com/eic/fun4all_eic_tutorials
- Fun4All Build your own GEANT4 Detector https://github.com/eic/fun4all_g4exampledetector

CHAPTER

THREE

REPOSITORIES

3.1 GitLab

MAIN GITLAB SERVER LINK

- **Detectors - DD4Hep based detectors implementations:**
  - ATHENA detector The reference detector for ATHENA detector
  - IP6 beamline
- EICD - EIC event data model implemented in PODIO
- Project Juggler - Gaudi based digitization and reconstruction framework
- **NPDet - Nuclear physics detectors library. A set of ready to plugin detectors, examples and tools for DD4Hep for NP**
  - documentation
  - reference API
- Datasets - scripts for generating data for benchmarks
- **Benchmarks - detector, physics and other analyses that are run automatically by CI when changes to detectors or reconstruction are made.**
  - Detector benchmarks - Detector benchmarks are meant to test for regressions in individual detector subsystems without a complex reconstruction
  - Reconstruction benchmarks - Reconstruction specific benchmarks (calorimetry, tracking, PID)
  - Physics benchmarks - Check physical processes such as DIS, DVCS, etc.
  - Cost estimation - (in develoment) A weighted evaluation of the detector performance versus estimated cost.
  - Common benchmarks - Code common to all benchmarks
- Documentation (dir) - this website, tutorials
- Container - source of docker images
- MC Convert - Pythia6, Beagle and others -> hepmc2/3 converter

3.2 External repositories

- DD4hep
- PODIO
- Gaudi
- Delphes EIC

CHAPTER

FOUR

CONTAINERS

The containers are released both for docker and singularity. Singularity images are automatically propagated to CVMFS:

- Containers repository
- Docker Hub
- CVMFS path:

```
singularity run /cvmfs/singularity.opensciencegrid.org/eicweb/jug_xl:3.0-
↪stable
```

The versioning is:

- **nightly** - recreated every night, using master branches
- **testing** - recreated every night, using fixed versions
- **4.0-stable** - the latest stable version from 3.0 branch
- **unstable** (unstable-vX.x in future) - the latest build from triggered by CI from changed repo
- **4.0.0** - exact tagged version

Nightly and testing are different in that nightly uses the master branch of the software, while testing uses whatever version is given at the top of the *gitlab-ci.yml <https://eicweb.phy.anl.gov/containers/eic_container/-/blob/master/.gitlab-ci.yml#L68>*. So testing is the precursor to the next release version.

Images structure is:

- **debian_base** - is a container generic base container based on amd64/debian:testing
- **jug_dev** - have all major HENP packages such as ROOT, Geant4 and DD4HEP but without detector and reconstruction. The image is used for testing purposes and automation.
- **jug_xl** - intended to be used to run simulation and work on detectors for users. jug_dev + full simulation packages

4.1 Singularity

The below command creates the right working environment. It checks if there are CVMFS images available (which is true for JLab and BNL farms) and links them or automatically download images (which is a scenario for users laptops). It also creates eic_shell with the right environment setup, prepares the current dir to work with detector or etc.

```
curl https://eicweb.phy.anl.gov/containers/eic_container/-/raw/master/install.sh | bash
```

Please follow Use singularity how-to for full details.

4.2 Docker containers

Containers are available at eicweb namespace at the dockerhub

1. To load the container environment in your run scripts, you have to do nothing special. The environment is already setup with good defaults, so you can use all the programs in the container as usual and assume everything needed to run the included software is already setup.
2. If using this container as a basis for a new container, you can direction access the full container environment from a docker `RUN` shell command with no further action needed. For the most optimal experience, you can install your software to `/usr/local` to fully integrate with the existing environment. (Note that, internally, `/usr/local` is a symlink to `/opt/view`).

4.3 Packages & versions

- Included software:
  - gcc@10.2.0
  - cmake@3.18.4
  - fmt@7.1.2
  - spdlog@1.5.0
  - nlohmann-json
  - heppdt@3.04.01
  - clhep@2.4.1.3
  - eigen@3.3.8
  - python@3.7.8 with pip, numpy, pyyaml, pyafp, matplotlib, ipython, scipy
  - xrootd@4.12.3
  - root@6.22.06
  - pythia8@8303
  - hepmc3@3.2.2 +python +rootio
  - stow@2.3.1
  - podio@0.13
  - geant4@10.6.2
  - dd4hep@1.14.1

  - acts@1.00.0
  - gaudi@34.0
  - dawn@3.91a
  - dawncut@1.54a
  - opencascade
- The singularity build exports the following applications:
  - eic_shell: a development shell in the image
  - container_dev: same as EIC shell
  - ipython

CHAPTER

FIVE

CVMFS & SPACK

CHAPTER

SIX

SUPPORT

- Mail lists
- Quick help and chat is available at our Slack channel

People responsible for the software:

Core Group:

Whitney Armstrong (Argonne), Andrea Bressan (INFN), Sylvester Joosten (Argonne), Dmitry Romanov (Jefferson Lab)

Existing software support:

- **Fun4All** : Kolja Kauder
- **eic_smear** : Kolja Kauder
- **Delphes** : Miguel Arratia, Stephen Sekula
- **ESCalate** : Dmitry Romanov, Yulia Furletova (g4e), Nathan Brei (Tracking)
- **MC Samples** : Andrea Bressan

CHAPTER

SEVEN

SINGLE PARTICLES

Single particle productions are split by particle type, angular regions, and thrown energy. All productions are thrown uniformly in cos(theta) and fixed in energy.

All productions are relative to the following locations on S3 and XRootD: * https://dtn01.sdcc.bnl.gov:9000/minio/eictest/ATHENA/ (see s3_file_storage.rst for instructions) * root://sci-xrootd.jlab.org//osgpool/eic/ATHENA/ (see xrootd_file_storage.rst for instructions)

Links point to the S3 location; the conversion to XRootD URIs should be trivial. Mirroring from S3 to XRootD may take up to one day.

7.1 Lepton endcap (N), 130 to 177 degrees

- **geometry `master`: (1M each, last update: 2021-08-14)**
  - e-: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20

7.2 Barrel region (B), 45 to 135 degrees

- **geometry `master`: (1M each, last update: 2021-08-14)**
  - e-: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20
  - gamma: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20
  - kaon0L: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20
  - mu-: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20
  - neutron: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20
  - pi+: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20
  - pi0: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20

7.3 Hadron endcap (P), 3 to 50 degrees

- **geometry `master`: (1M each, last update: 2021-08-14)**
  - pi+: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20
  - kaon0L: E [GeV] = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20

CHAPTER

EIGHT

PHYSICS GENERATORS

Physics productions are divided by physics working group or by process type.

All productions are relative to the following locations on S3 and XRootD: * https://dtn01.sdcc.bnl.gov:9000/minio/eictest/ATHENA/ (see s3_file_storage.rst for instructions) * root://sci-xrootd.jlab.org//osgpool/eic/ATHENA/ (see xrootd_file_storage.rst for instructions)

Links point to the S3 location; the conversion to XRootD URIs should be trivial. Mirroring from S3 to XRootD may take up to one day.

In case of questions or problems with this data, please let the Software Working Group know in the #software-helpdesk channel, or via email to eic-ip6-software-l@lists.bnl.gov.

8.1 Deep Inelastic Scattering

- **geometry: *master* (1M each, last update: 2021-08-18)**
  - 5x41: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 5x100: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 10x100: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 10x275: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 18x275: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
- **geometry: *acadia-v1.0-alpha* (1M each, last update: 2021-08-19)**
  - 5x41: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 5x100: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 10x100: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 10x275: minQ2=1, minQ2=10, minQ2=100, minQ2=1000
  - 18x275: minQ2=1, minQ2=10, minQ2=100, minQ2=1000

8.2 Deeply Virtual Compton Scattering

- **geometry:** ***acadia-v1.0-alpha*** **(last update: in progress)**
  - 18x275 (2M events)
  - 10x100 (2M events)
  - 5x41 (1M events)

8.3 Timeline Compton Scattering

- **geometry:** ***acadia-v1.0-alpha*** **(last update: in progress)**
  - 18x275
  - 5x100
  - 5x41

8.4 Spectroscopy

- **geometry:** ***acadia-v1.0-alpha*** **(last update: in progress)**
  - 10x100 psi2s, x, y

CHAPTER

NINE

FREQUENTLY ASKED QUESTIONS

9.1 It seems that the *ReconstructedParticles* branches are empty, but used to be filled.

ReconstructedParticles used to be filled with just smeared info (so analyzers could start developing scripts even if the values in there were not from proper track reconstruction and event building). In the first *acadia* production we had an unexpected issue filling *ReconstructedParticles*, but that should be resolved soon. Plan on having access to the actual momentum from track reconstruction in *ReconstructedParticles* (with exact PID from truth, and energy from momentum and PID).

9.2 Should the physics working groups plan to use the calorimeter clusters?

We are still working on full event building that will also take into account energy from clusters, so in *acadia* the energy is based on the momentum and truth PID.

9.3 Which four-vectors should I plan to use?

As for four-vectors, this is somewhat up to you. We are not providing four-vectors in *ReconstructedParticles*. For some particles they will be four-vectors, though: particles without info from either tracking or calorimetry. Depending on the analysis you are performing, you may want to use momentum direction and energy (say, higher energy central electrons) while for others you may want to use momentum direction and momentum magnitude (say, low energy central electrons).

9.4 Why did the files I was using disappear? Will the files disappear?

- Files on S3 will not disappear, though RECO files may be replaced by updated reconstruction.
- Files on XRootD may disappear when the entire directory is deleted to stay within the required space allocations.

9.5 How is GeneratedParticles different from mcparticles?

mcparticles contains the full event generator record, including internal particles that are not in the final state (think: a pi0 with a strong decay into two pions). It is a record that is uses geant4 steps as the basis for its structure, and therefore includes start and end values.

GeneratedParticles contains only the event generator final state particles only (those with HepMC status 1). It is structured in exactly the same way as *ReconstructedParticles* and allows for comparison of truth with reconstruction.

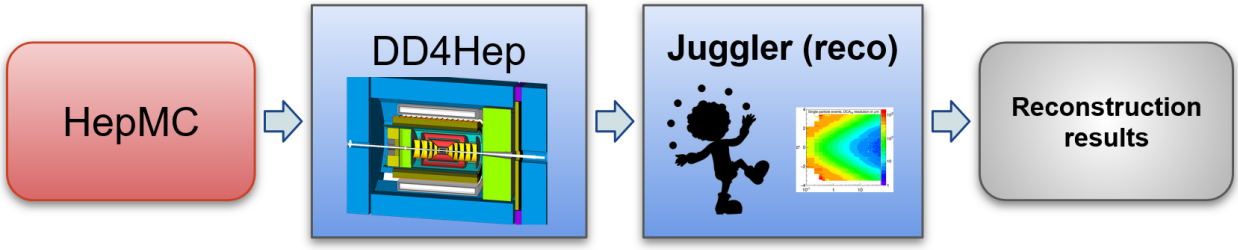
CHAPTER

TEN

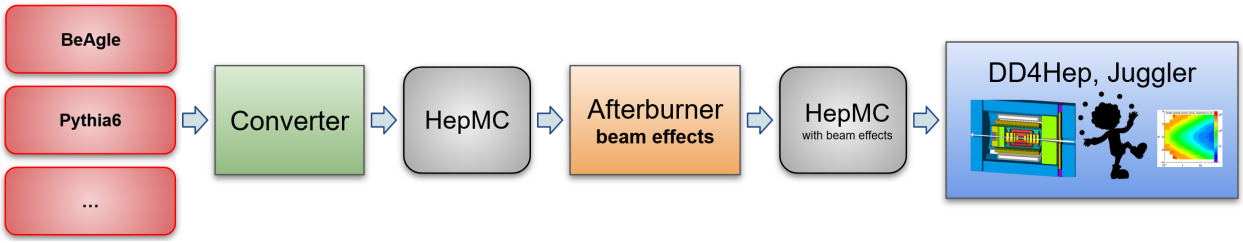
PROCESS MCEG

- How to run MCEG files throught the Software
- What are the requirements and limitations for the MCEG
- How to convert files to the right format
- How to apply beam effects and crossing angle

10.1 MCEG Overview



Athena full simulation and reconstruction stack accepts only HepMC2 and HepMC3 as an input format. The reason for that is that support for many formats on simulation/reconstruction level leads to increasing number of bugs witch are difficult to mitigate with testing because of the combinatorics. It is much easier in terms of testing, robustness and reprodusobility to have a reliable chain that works with one (universal) format and have well tested standalone satelite conventers, that allow to work with different MCEG software (see the next picture).



- **Converters** allow to convert other formats to HepMC
- **Afterburner** applies crossing angle and beam effects

10.2 Convert MCEG

Currently there are 2 packages that allow to convert different MCEG to HepMC

- **EicSmear - and old (means well tested) and reliable package**
  that allows to convert various (and especially old) MCEG to HepMC format. Instruction how to convert to HepMC It is recommended to use EicSmear especially for BNL RootTree files.
- **mcconv - new and less reliable, but kind of easier**
  to use for users and also has convenient python API. mcconv correctly works with GEMC LUND (Clas12) format.

10.2.1 Example converting CLAS12 file to HepMC

```
# install mcconv
pip3 install --upgrade mcconv

# if there is a sertificate problems
python3 -m pip install --trusted-host pypi.python.org --trusted-host files.pythonhosted.
↪org --trusted-host pypi.org --user -U mcconv

# Add home bin directory to the PATH environment variable
export PATH=~/.local/bin:$PATH

# convert lund gemc
mcconv input.txt -i lund_gemc -b 10x100 -v

# File from example
mcconv /work/eic/mc/meson_MC/OUTPUTS/0mrad_IR0/nov_2021/pi_n_10.0on100.0_x0.0010-1.0000_
↪q1.0-100.0_lund.dat -v -p 1000
```

10.3 Crossing angle and beam effects

In order to apply crossing and beam effects to existing HepMC files EIC Afterbur is to be used. The package provides framework independent well validated crossing angle and beam effects C++ library and HepMC file converter (abconv) for Electron Ion Collider.

Physics simulated:

- Crossing angle
- Beam effects (divergence, crabbing kick, etc.)
- Vertex spread (position, time)

Please follow the instruction of how to use the afterburner.

10.4 Using converters and afterburner

Both converters and aferburner are parts of the container. And could be used from JLab and BNL farm. Follow singularity chapter.

CHAPTER

ELEVEN

RUN FAST SIMULATION

11.1 Delphes

https://delphes-eic.readthedocs.io/en/latest/

11.2 EicSmear

https://github.com/eic/eic-smear#procedure

CHAPTER

TWELVE

FULL SIMULATION AND RECO

12.1 Quick tutorials

This is quick tutorial with the steps of how to run sim&recon from scratch. This work both on BNL and JLab farms as well as personal PCs.

12.1.1 0. Install eic-shell

```
> curl https://eicweb.phy.anl.gov/containers/eic_container/-/raw/master/install.sh | bash

# to run eic environment in singularity container
> ./eic-shell
```

More on installing and using singularity here: Use singularity

12.1.2 1. Convert MCEG to HepMC

The input format should be in HepMC format. If the conversion is needed: Convert MCEG

12.1.3 2. Run DD4HEP simulation

Select a detector

```
# Detectors live in
# /opt/detectors
# one can select particular configuration as
# source /opt/detector/athena-deathvalley-1.5T/setup.sh
#
# or one can set the latest detector
source /opt/detector/setup.sh

# Run simulation
npsim --compactFile=$DETECTOR_PATH/athena.xml --runType=run -N=2 --outputFile=sim_output.
↪edm4hep.root --inputFiles mceg.hepmc
```

12.1.4 3. Run Juggler/Gaudi reconstruction

```
#set the same detector as in the simulations
source /opt/detector/setup.sh

export JUGGLER_SIM_FILE=sim_output.edm4hep.root JUGGLER_REC_FILE=rec_output.edm4hep.root␣
↪JUGGLER_N_EVENTS=10
gaudirun.py /opt/benchmarks/physics_benchmarks/options/reconstruction.py
```

12.2 Full Tutorials

More detailed tutorials are available on a dedicated websites:

12.2.1 Detector simulation

- Full simulation tutorials
- Quick start guide
- Part I. Simple detecotor
- Part II. Part 2: Modifying and Adding Detectors

Perticipate in feedback!

12.2.2 Tracking

- (in development) ACTS tracking repo

12.3 Full simulation tips

12.3.1 Particle gun

There are at least 2 ways of running a particle gun out of the box:

- using npsim command line
- using geant macro file and invoke gps

Using npsim (wrapper around ddsim) command line:

```
# Assumed to run from the root of Athena detector repo
# no spread
npsim --compactFile=athena.xml --runType=run -G -N=2 --outputFile=test_gun.root --gun.
↪position "0.0 0.0 1.0*cm" --gun.direction "1.0 0.0 1.0" --gun.energy 100*GeV --part.
↪userParticleHandler=''

# uniform spread inside an angle:
npsim --compactFile=athena.xml -N=2 --random.seed 1 --enableGun --gun.energy 2*GeV --gun.
↪thetaMin 0*deg --gun.thetaMax 90*deg --gun.distribution uniform --outputFile test.root
```

Using GPS

General Particle Source tutorial

GPS is configured in Geant4 macro files. An example of such file may be found here

To run npsim with GPS you have to add *–enableG4GPS* flag and specify Geant4 macro file:

```
npsim --runType run --compactFile athena.xml --enableG4GPS --macro macro/gps.mac --
↪outputFile gps_example.root
```

12.3.2 Geometry visualization

There are many ways to see the geometry and tracks:

1. Through Geant4 event display
2. geoDisplay (root geoViewer)
3. ddeve (root EVE based event display…)
4. dd_web_display (using browser and jsroot library)

To run Geant4 event display:

```
# Assumed to run from the root of Athena detector repo
npsim --runType vis --compactFile athena.xml --macro macro/vis.mac --outputFile test.
↪root --enableG4GPS --enableQtUI
```

12.4 Geometry conversion

12.4.1 Convert to GDML

There is a convert_to_gdml.py script in the detector repository (https://eicweb.phy.anl.gov/EIC/detectors/athena/-/blob/master/scripts/convert_to_gdml.py). That can be used to export ALL of ATHENA to gdml, but not individual detector systems.

This is actually done on every commit and the results are saved as job artifacts.

The latest athena.gdml from the master branch

12.4.2 Convert to root

One can use dd_web_display to actually just save root geometry

```
dd_web_display --export athena.xml  # will create a .root file with the geometry
```

12.5 How XML is invoked from C++ in DD4Hep

> How does the XML file in the compact directory know which c++ file to load when we include the respective XML file in the main athena.xml file?

1. Xml compact files has <*detector* . . . ><*/detector*> tag which has *type* attribute that tells which C++ type to use:

```
<detector id="ForwardRICH_ID" name="DRICH" type="athena_DRICH" ... >
```

2. C++ file usually has *createDetector(. . .)* function and *DECLARE_DETELEMENT* macro which binds the function to the name used in xml <*detector*> type attribute. C++ code looks like this:

```
static Ref_t createDetector(Detector& desc, xml::Handle_t handle, SensitiveDetector␣
↪sens) {
    // ...
}

DECLARE_DETELEMENT(athena_DRICH, createDetector)
```

3. How DD4Hep finds and loads compiled components?

   > This is going into technical details which users usually don't need. Installation paths and environment variables are set by container/spack and should work out of the box.

   DD4Hep uses modular plugin mechanism to load C++ code. When athena C++ is compiled, two files are created: - *athena.components* - a text file stating what components one can find in athena.so. From our example, there will be a record like *v2:libathena.so:athena_DRICH* among other records. - *libathena.so* - compiled C++ library with all detectors from athena repo

   So when the type of the detector is given, like *athena_DRICH*. DD4Hep uses *LD_LIBRARY_PATH* to look through .components files and then figures out what .so file to load to get the correct C++ code executed.

CHAPTER

THIRTEEN

USE SINGULARITY

13.1 Install singularity

Eic image require singularity > 3.0

The oficial installation instructions have many steps.

If you have ubuntu, there is a debian repo with 3.5.2 version, which works pretty nicely (You will have to install dependencies, it will print them...)

```
sudo apt install containernetworking-plugins
wget http://ftp.fi.debian.org/debian/pool/main/s/singularity-container/singularity-
↪container_3.5.2+ds1-1_amd64.deb
sudo dpkg -i singularity-container_3.5.2+ds1-1_amd64.deb
```

Please don't install NeuroDebian repo from the repo as it holds v2.6 and eic image require singularity > 3.0.

13.2 Install work environment

The below command automatically creates the right working environment for detector development and running the reconstruction. It checks if there are CVMFS images available (which is true for JLab and BNL farms) and links them or downloads images (which is a scenario for users laptops). It also creates eic_shell with the right environment setup, prepares the current dir to work with detector or etc.

```
# Easy to remember link:
curl -L get.athena-eic.org | bash

# Which is an alias to:
curl https://eicweb.phy.anl.gov/containers/eic_container/-/raw/master/install.sh | bash
```

install.sh checks if it is run on BNL or JLab farms, so existing CVMFS images are used and installation is almost instant. On local systems singularity images will be downloaded.

It might be handy to copy install.sh locally and control where singularity images are being copied, disable CVMFS behaviour, and other parameters:

```
curl -L get.athena-eic.org -o install.sh
chmod +x install.sh
./install.sh --help
```

Flag	Description
-p,–prefix	Working directory to deploy the environment (D: /home/romanov/anl)
-t,–tmpdir	Change tmp directory (D: /tmp)
-n,–no-cvmfs	Disable check for local CVMFS (D: enabled)
-c,–container	Container version (D: jug_xl)
-v,–version	Version to install (D: nightly)
-h,–help	Print this message

Example of controlling the container version and image location:

```
# installs testing variant and stores image at /mnt/work/images
# (!) one has to create <prefix>/local/lib for images
mkdir -p /mnt/work/images/local/lib/
./install.sh -v testing -p /mnt/work/images
```

13.3 Detector simulation

After the installation you should have an executable script named **eic-shell** which basically just runs singularity setting the proper environment (more information about the script is below)

13.3.1 Precompiled detector

The jug_xl container comes with precompiled detecor repository. It could be used out of the box for simulations or even changing detector parameters that doesn't require recompilation.

The precompiled detector is installed in **/opt/detector** directory. And can be used like this:

```
# Setup the proper detector environemnt
source /opt/detector/setup.sh

# Run particle gun simulation
npsim -N2 --compactFile=$DETECTOR_PATH/athena.xml --random.seed 1 --enableGun --gun.
↪energy 2*GeV --gun.thetaMin 0*deg --gun.thetaMax 90*deg --gun.distribution uniform --
↪outputFile ~/test.root
```

13.3.2 Clone detector from repository

```
git clone https://eicweb.phy.anl.gov/EIC/detectors/athena.git
git clone https://eicweb.phy.anl.gov/EIC/detectors/ip6.git
ln -s ../ip6/ip6 athena/ip6

# Build athena
mkdir athena/build && cd athena/build
cmake ../ -DCMAKE_INSTALL_PREFIX=$ATHENA_PREFIX
cmake --build ./ --target all -- -j 8
cmake --install ./
cd -
```

(continues on next page)

(continued from previous page)

```
# Build IP6
mkdir ip6/build && cd ip6/build
cmake ../ -DCMAKE_INSTALL_PREFIX=$ATHENA_PREFIX
cmake --build ./ --target all -- -j 8
cmake --install ./
cd -

# Test run geometry browser:
dd_web_display athena/athena.xml

# Test run particle gun:
npsim -N2 --compactFile=athena/athena.xml --random.seed 1 --enableGun --gun.energy 2*GeV␣
↪--gun.thetaMin 0*deg --gun.thetaMax 90*deg --gun.distribution uniform --outputFile ~/
↪test.root
```

13.4 Advanced information

13.4.1 CVMFS

For farms like at BNL or JLab the images are automatically replicated to CVMS:

```
/cvmfs/singularity.opensciencegrid.org/eicweb/jug_xl*

# example to run
singularity run /cvmfs/singularity.opensciencegrid.org/eicweb/jug_xl:4.0-acadia-stable
```

13.4.2 eic-shell explained

There are actually two eic-shell scripts. One is created by the install scripts and the other lives in the container.

The one outside the container just sets ATHENA_PREFIX and runs singularity like:

```
# $PREFIX here is where you installed everything (by default where install.sh executed)
export ATHENA_PREFIX=$PREFIX/local
export SINGULARITY_BINDPATH=/mnt
singularity exec $PREFIX/local/lib/jug_xl-nightly.sif eic-shell $@
```

The **eic-shell** inside the container loads the proper environemnt and SHELL look correctly

```
## Properly setup environment
. /etc/eic-env.sh

# What eic-env does in the end is
export LD_LIBRARY_PATH=$ATHENA_PREFIX/lib:$LD_LIBRARY_PATH
export PATH=$ATHENA_PREFIX/bin:$PATH

# Run bash shell
```

CHAPTER

FOURTEEN

S3 FILE STORAGE

To share files S3 (Simple Storage Service) is used. In this tutorial we use MinIO client to work with it.

You can get access, browse and download files using your browser, using this link:

ATHENA S3 Space

The LOGIN and PASSWORD for read access are available for all members (but can't be published at open places like this site). Please ask on #software-helpdesk Slack channel or email one of the SWG members.

One can access S3 directly using TFile like this

```
TFile* file = TFile::Open("s3https://dtn01.sdcc.bnl.gov:9000/eictest/ATHENA/<path>.root",
↪"AUTH=<login>:<password>");
```

Or set environment variables *S3_ACCESS_KEY* and *S3_SECRET_KEY* instead of using AUTH and putting login and password to code;

```
export S3_ACCESS_KEY=<login>
export S3_SECRET_KEY=<password>
```

If you encounter *Forbidden (403)* errors at BNL RCF, try adding the *NOPROXY* option:

```
TFile* file = TFile::Open("s3https://dtn01.sdcc.bnl.gov:9000/eictest/ATHENA/<path>.root",
↪"NOPROXY AUTH=<login>:<password>");
```

14.1 XRootD access

One can utilize XRootD server to get access to files without credentials using:

```
root://sci-xrootd.jlab.org//osgpool/eic/ATHENA/<path to file>
```

```
std::string fileName = "RECO/JETS/crossDivNrgCrab/DIS_NC_Q2gt10_crossDivNrgCrab_25mRad_
↪18x275_v1.0001.root";
TFile* file = TFile::Open("root://sci-xrootd.jlab.org//osgpool/eic/ATHENA/" + fileName);
// ...
```

(!) Currently (July 2021) only RECO is on xrootd (since we only have a 2.5 TB limit there)

14.2 Installation

Installing MinIO client:

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
mkdir -p $HOME/bin && mv mc $HOME/bin/mc && chmod +x  $HOME/bin/mc
alias mc="$HOME/bin/mc"
mc --autocompletion
```

> You may name the command diffirently if mc aka midnight commander which also uses *mc* is installed

14.3 Getting access

Public read-only access

```
mc config host add S3 https://dtn01.sdcc.bnl.gov:9000 <login> <password>
```

Private write-enabled access

```
mc config host add S3rw https://dtn01.sdcc.bnl.gov:9000/ $user $key
```

Private write-enabled access at BNL

```
mc config host add S3rw http://eicoss01.sdcc.bnl.local:9000 $user $key
```

14.4 Operations

Listing contents (the eictest is historical and may change)

```
mc tree S3/eictest/ATHENA
mc ls S3/eictest/ATHENA
mc find S3/eictest/ATHENA --name '*.hepmc*'
mc du S3/eictest/ATHENA
```

Copying contents out

```
mc cp --insecure S3/eictest/ATHENA/EVGEN/JETS/crossDivNrgCrab/test_crossDivNrgCrab_
↪25mRad_18x275_v1.hepmc .
```

Mirroring directories

```
mc mirror --insecure S3/eictest/ATHENA/EVGEN/JETS/crossDivNrgCrab/ crossDivNrgCrab/
```

Streaming contents out

```
mc cat --insecure S3/eictest/ATHENA/EVGEN/JETS/crossDivNrgCrab/test_crossDivNrgCrab_
↪25mRad_18x275_v1.hepmc | head -n 20
mc cat S3/eictest/ATHENA/EVGEN/JETS/crossDivNrgCrab/test_crossDivNrgCrab_25mRad_18x275_
↪v1.hepmc.gz | gunzip -c | head -n 20
```

Copying contents in

```
mc cp /gpfs02/eic/bpage/home/eicBeamSimu/Pythia8/headonTestJin/test_crossDivNrgCrab_*␣
↪S3rw/eictest/ATHENA/EVGEN/JETS/crossDivNrgCrab
```

Streaming contents in

```
cat /gpfs02/eic/bpage/home/eicBeamSimu/Pythia8/headonTestJin/test_crossDivNrgCrab_25mRad_
↪18x275_v1.hepmc | mc pipe S3rw/eictest/ATHENA/EVGEN/JETS/crossDivNrgCrab/test_
↪crossDivNrgCrab_25mRad_18x275_v1.hepmc
gzip -c /gpfs02/eic/bpage/home/eicBeamSimu/Pythia8/headonTestJin/test_crossDivNrgCrab_
↪25mRad_18x275_v1.hepmc | mc pipe S3rw/eictest/ATHENA/EVGEN/JETS/crossDivNrgCrab/test_
↪crossDivNrgCrab_25mRad_18x275_v1.hepmc.gz
```

CHAPTER

FIFTEEN

BEAGLE GENERATOR

BeAGLE - Benchmark eA Generator for LEptoproduction

The documentation on eic.github.io

Currently hosted at https://gitlab.in2p3.fr/BeAGLE/BeAGLE

Contacts:

Mark Baker mdbaker@bnl.gov Kong Tu zhoudunming@bnl.gov

15.1 CVMFS use

Here my hero lays the quest. No mere mortal is capable of building BeAGLE on his system and not everyone is able to run it on farms. But if you are still dare to try, here are the steps:

```
# Everything in CSH, you are doomed with your bash, zsh and others

setenv EICDIRECTORY /cvmfs/eic.opensciencegrid.org/x8664_sl7/MCEG/releases/env/EIC2021a/
setenv BEAGLESYS $EICDIRECTORY/PACKAGES/BeAGLE
setenv LHAPDF_DATA_PATH /cvmfs/sft.cern.ch/lcg/external/lhapdfsets/current
setenv LHAPDF_DATA_PATH ${LHAPDF_DATA_PATH}:${EICDIRECTORY}/share/LHAPDF/
setenv LHAPATH /cvmfs/sft.cern.ch/lcg/external/lhapdfsets/5.9.1/share/PDFsets
setenv LHAPDF5 /cvmfs/sft.cern.ch/lcg/external/lhapdfsets/5.9.1/share/PDFsets
setenv LD_LIBRARY_PATH $EICDIRECTORY/gcc-8.3/lib:$LD_LIBRARY_PATH
source /cvmfs/eic.opensciencegrid.org/gcc-8.3/opt/fun4all/core/gcc/8.3.0.1-0a5ad/x86_64-
↪centos7/setup.csh

# (nice idea would be to add this to a single file to source. But for hardcore knights␣
↪of keyboard and console
#  it is adviced not to do so, memorize the commands and type them in manually every␣
↪time to feel the glory)

# Now when you are equipped, let the quest begin:
mkdir My_Beagle_Quest      # We name it awkwardly, so it is inline with everything else
cd My_Beagle_Quest
mkdir outForPythiaMode

cp $BEAGLESYS/nuclear.bin .
cp $BEAGLESYS/Examples/eD_18x135_Q2_1_10_y_0.01_0.95_test40k_Shd1_tau7_kt=ptfrag=0.32_
↪shdfac=1.32.Jpsidiffnodecay.highpf.inp example.inp
cp $BEAGLESYS/Examples/eAt1dfJn .
```

(continues on next page)

(continued from previous page)

```
$BEAGLESYS/BeAGLE < example.inp

#(!) Beware, example.inp has 1mil events to generate. You probably would like to␣
↪decrease the number for the first run
```

CHAPTER

SIXTEEN

GEOMETRY VIEWER

The latest geometry for **master** branch is available:

- Central detector (no obj browser)
- Central detector (+ obj browser)
- Full detector with beamline (no browser)
- Full detector with beamline (+ obj browser)

It might take several minutes to load due to a number of ECal fibers. We are working on fixing the performance issue.

On each commit to the **master** branch detector geometry is saved to ROOT TGeo and GDML formats. File links are:

- **detector_geo_full.root**
  https://eicweb.phy.anl.gov/api/v4/projects/473/jobs/artifacts/master/raw/geo/detector_geo_full.root
- **detector_geo.root**
  https://eicweb.phy.anl.gov/api/v4/projects/473/jobs/artifacts/master/raw/geo/detector_geo.root
- **athena.gdml**
  https://eicweb.phy.anl.gov/api/v4/projects/473/jobs/artifacts/master/raw/geo/athena.gdml

If one gets 404 or 400 error retriving those file, that is most probably means, that a master branch code and its artifacts (resulting files) are being built right now and not yet ready. You can check the staus at repository pipelines

CHAPTER

SEVENTEEN

LOCAL INSTALLATION

The supported local installation is done with Spack

Spack is a package management tool designed to support multiple versions and configurations of software on a wide variety of platforms and environments. It was designed for large supercomputing centers but can also be used by a regular users.

Spack installation allows to install ATHENA stack natively on your system (as much as possible). It is easier to work with in terms of IDEs, OpenGL (Geant4 event display), debugging, etc. The downside of this is that it requires of compilation of lots of packages, which takes time (hours) and is prone to errors depending on your system. If setup fails, it require some spack-kung-foo to debug and recover the installation. Thus:

THIS TYPE OF INSTALLATION IS CONSIDERED FOR EXPERTS. DO IT ON YOUR OWN RISK.

17.1 Installation

To install Spack and EIC-Spack repository it is recommended to use master branch of spack as several of issues we submitted are not yet on any tag.

```
git clone https://github.com/spack/spack.git
. spack/share/spack/setup-env.sh                # Add this to your e.g. ~/.bashrc
git clone https://github.com/eic/eic-spack.git
spack repo add eic-spack
```

Create environment file **athena.yaml** with the next content:

```
spack:
    specs:
    - acts +dd4hep +digitization +examples +fatras +geant4 +identification +json +tgeo␣
↪+ipo
    - boost@1.76.00
    - cmake
    - clhep cxxstd=17
    - dd4hep +geant4 +assimp +hepmc3 +lcio +ipo
    - eigen
    - gaudi +ipo
    - geant4 +ipo +qt +opengl -python +threads +vecgeom cxxstd=17
    - genfit +ipo
    - hepmc3 +interfaces +python +rootio
    - lcio +ipo
    - mesa -llvm swr=none
```

(continues on next page)

(continued from previous page)

```
    - nano
    - podio +ipo
    - pythia8 +fastjet
    - root cxxstd=17 +fftw +fortran +gdml +mlp +pythia8 +root7 +tmva +vc +xrootd +ssl
    - xrootd cxxstd=17 +python
    - igprof
    - npdet +geocad
    - eicd
    - log4cxx
    - afterburner +zlib +root
    - juggler
    - athena-eic +reconstruction
    - py-pandas
    - py-matplotlib
  concretization: together
  config:
      install_missing_compilers: true
  # Optional: install path and view path
  #   install_tree:
  #    root: /opt/software
  #view: /opt/local
```

Create the environement from the file

```
spack env create athena athena.yaml
spack env activate athena
spack concretize
spack install --fail-fast -v -j8
```

17.2 References

- Spack main documentation
- Spack environments tutorial
- EIC-spack repository

CHAPTER

EIGHTEEN

EDIT THIS SITE!

eic.phy.anl.gov/ip6/

This site is open to everyone. Users are encouraged to contribute to this documentation! If you would like to add/edit the information on this site - you are welcome to do this!

The pages are automatically updated, when changes are commited to the master branch of

ATHENA SWG documentation GitLab repo

P.S. The suggested update workflow is by creating merge requests

18.1 Running on your machine

For more extensive editing it is convenient to run the site on your local machine to instantly see the changes.

```
# get the website
git pull https://eicweb.phy.anl.gov/EIC/documentation/athena athena_doc

# install sphinx python libraries
python3 -m pip install sphinx sphinx_rtd_theme recommonmark sphinx-autobuild

# start the site
cd athena_doc
sphinx-autobuild docs docs/_build/html
```

By default the site will be available at http://127.0.0.1:8000/ (you can look at sphinx-autobuild flags to change that)

18.2 Development

More formal explanation:

18.2.1 Requirments:

- Sphinx - Python documentation generator
- Read the Docs Sphinx Theme - Theme for final output
- recommonmark - Markdown pages support Sphinx

18.2.2 Building

```
# from project root
sphinx-autobuild docs docs/_build/html

# from docs root
sphinx-autobuild . _build/html

# one time build without watching for changes
sphinx-build docs docs/_build/html
```