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

FCC @ iLCDIRAC

André Sailer, Lorenzo Valentini
CERN-EP-SFT

FCC Software Meeting
June 26, 2023

Introduction

The objective of this project is providing large scale Monte Carlo productions with, for example, the Delphes Fast Simulation Framework, in particular its standalone executables using Pythia 8 for event generation.

Here we present one of the solutions for this problem: Transformations using the DIRAC grid tool, through the iLCDirac instance using the Key4hep software stack.

We show the as an example the workflow for event generation of the inclusive bb process.

Table of contents:

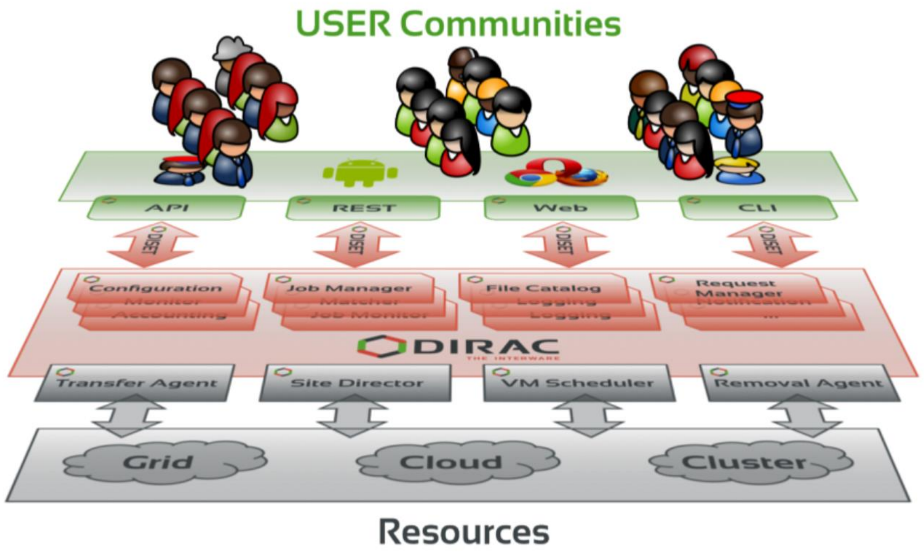
- (iLC)DIRAC in a nutshell
- Delphes Executables
- Delphes + Pythia workflow
- User Jobs : Delphes Application Interface
- Production Jobs: Transformation Config File
- Information, Support and Documentation

(iLC)DIRAC in a nutshell

iLCDirac is based on the DIRAC interware originally developed for LHCb.

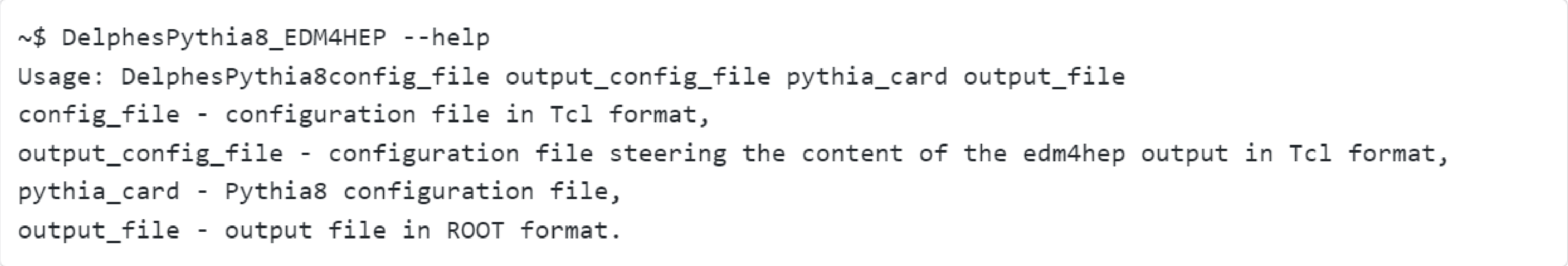
- Dirac (Distributed Infrastructure with Remote Agent Control): High level interface between users and distributed resources
- Distributed Workload Management: one interface to execute anywhere: batch farms, grid computing elements, HPCs
- Data Management (file transfers, meta data augmented file catalog)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

- The iLCDirac extension of Dirac is set up for the ILC, Calice, and FCC Virtual Organisations. It allows centralized MC Production.
- It also allows User Jobs to be run locally.
- iLCDirac uses almost all the functionalities provided by DIRAC.



Delphes Executables

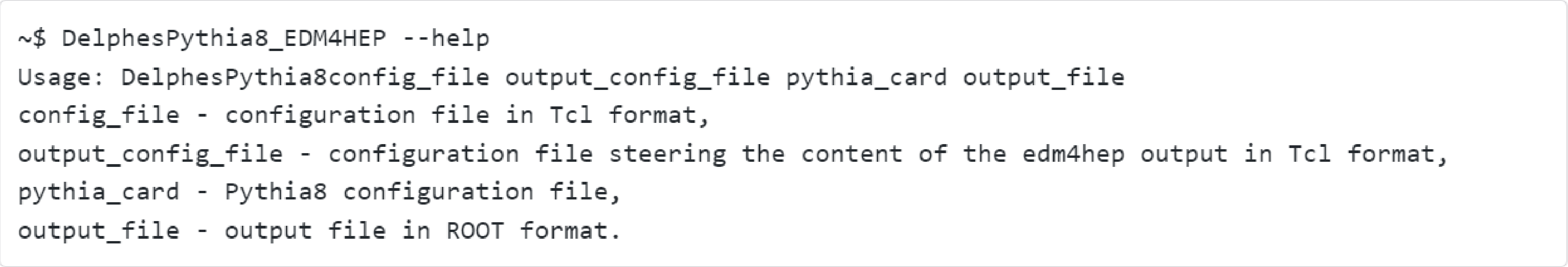
- The k4SimDelphes package offers standalone executables, similar to the ones Delphes offers:
  - DelphesSTDHEP - for reading STDHEP inputs
  - DelphesROOT - for reading ROOT files in the Delphes format
  - DelphesPythia8_EDM4HEP - for running Pythia8 as part of the simulation



```
~$ DelphesPythia8_EDM4HEP --help
Usage: DelphesPythia8config_file output_config_file pythia_card output_file
config_file - configuration file in Tcl format,
output_config_file - configuration file steering the content of the edm4hep output in Tcl format,
pythia_card - Pythia8 configuration file,
output_file - output file in ROOT format.
```

- New Delphes specific interface on iLCDirac, that can be used for these executables.

DelphesPythia8_EDM4HEP



```
~$ DelphesPythia8_EDM4HEP --help
Usage: DelphesPythia8config_file output_config_file pythia_card output_file
config_file - configuration file in Tcl format,
output_config_file - configuration file steering the content of the edm4hep output in Tcl format,
pythia_card - Pythia8 configuration file,
output_file - output file in ROOT format.
```

- The pythia card can be used for:
  - Event generation
  - LHE reader
- Two possible workflows for DelphesPythia8_EDM4HEP:
  - Event generation with Pythia and simulation with Delphes, in a single step,
  - Event generation with external generator (as long as has outputs in LHEf) + Delphes simulation using the pythia card for reading LHEf.

Delphes + Pythia workflow

- Implementing Delphes workflow #3 as described in: https://docs.google.com/document/d/18TAhC62jkE0C5rikOUq9xM-oIL6LpEccLvo4f2HofyY/
- Output folder structure as described in: https://docs.google.com/document/d/10zgE2gyoRbwV6qC9KB7W8SRiwL7fjcXMBkP6I722C-E/

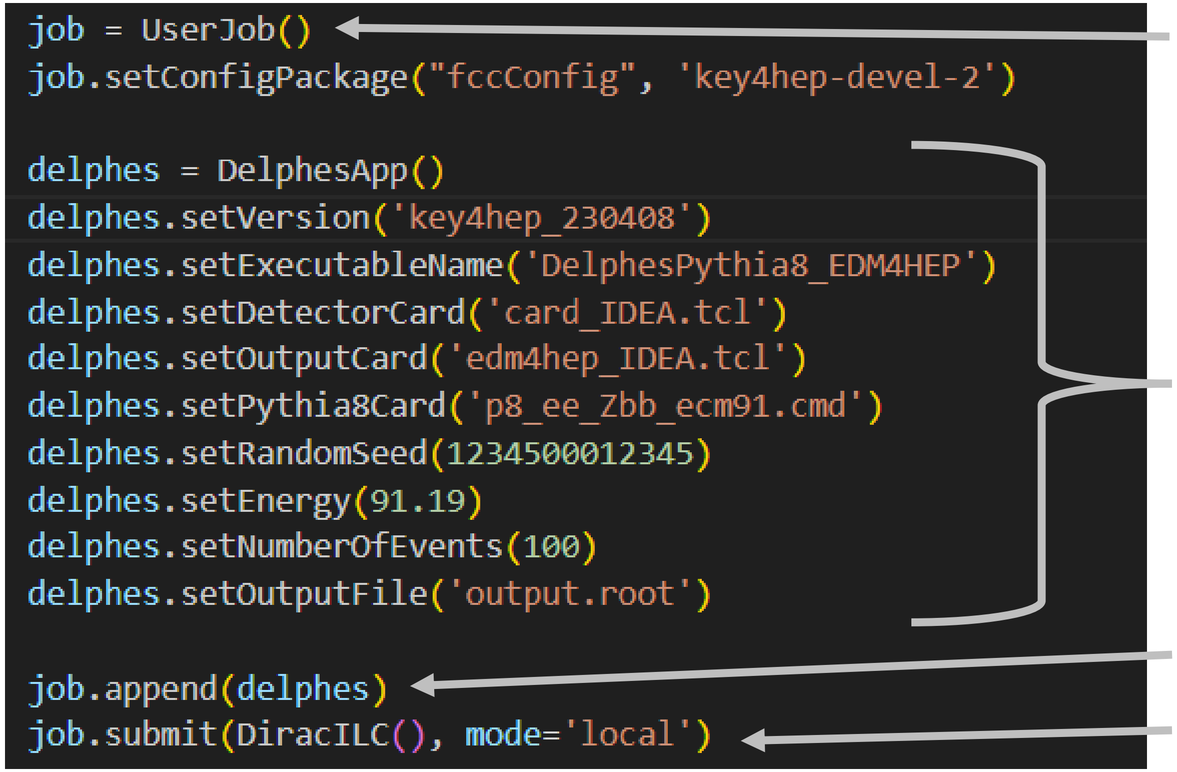
- Run Delphes standalone executable, generating events with Pythia:

```
DelphesPythia8_EDM4HEP card_IDEA.tcl edm4hep_IDEA.tcl p8_ee_Zbb_ecm91.cmd output.root
```

- Output location:

```
/eos/experiment/fcc/prod/fcc/ee/winter2023/91.19gev/Zbb/idea/delphes/00012345/
```

Delphes App | interface



```
job = UserJob()
job.setConfigPackage("fccConfig", 'key4hep-devel-2')

delphes = DelphesApp()
delphes.setVersion('key4hep_230408')
delphes.setExecutableName('DelphesPythia8_EDM4HEP')
delphes.setDetectorCard('card_IDEA.tcl')
delphes.setOutputCard('edm4hep_IDEA.tcl')
delphes.setPythia8Card('p8_ee_Zbb_ecm91.cmd')
delphes.setRandomSeed(1234500012345)
delphes.setEnergy(91.19)
delphes.setNumberOfEvents(100)
delphes.setOutputFile('output.root')

job.append(delphes)
job.submit(DiracILC(), mode='local')
```

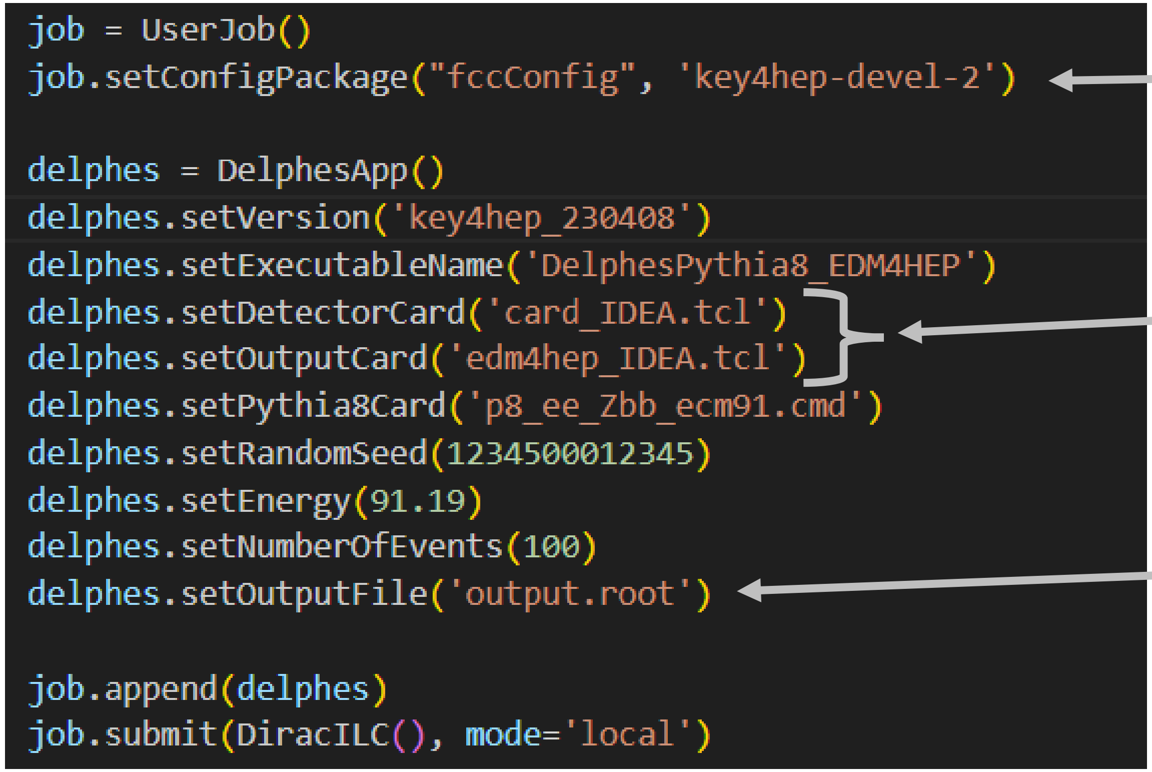
UserJob submission as an example

Configuration of the application

Adding the application to the job

Submission of the UserJob locally

Delphes App interface



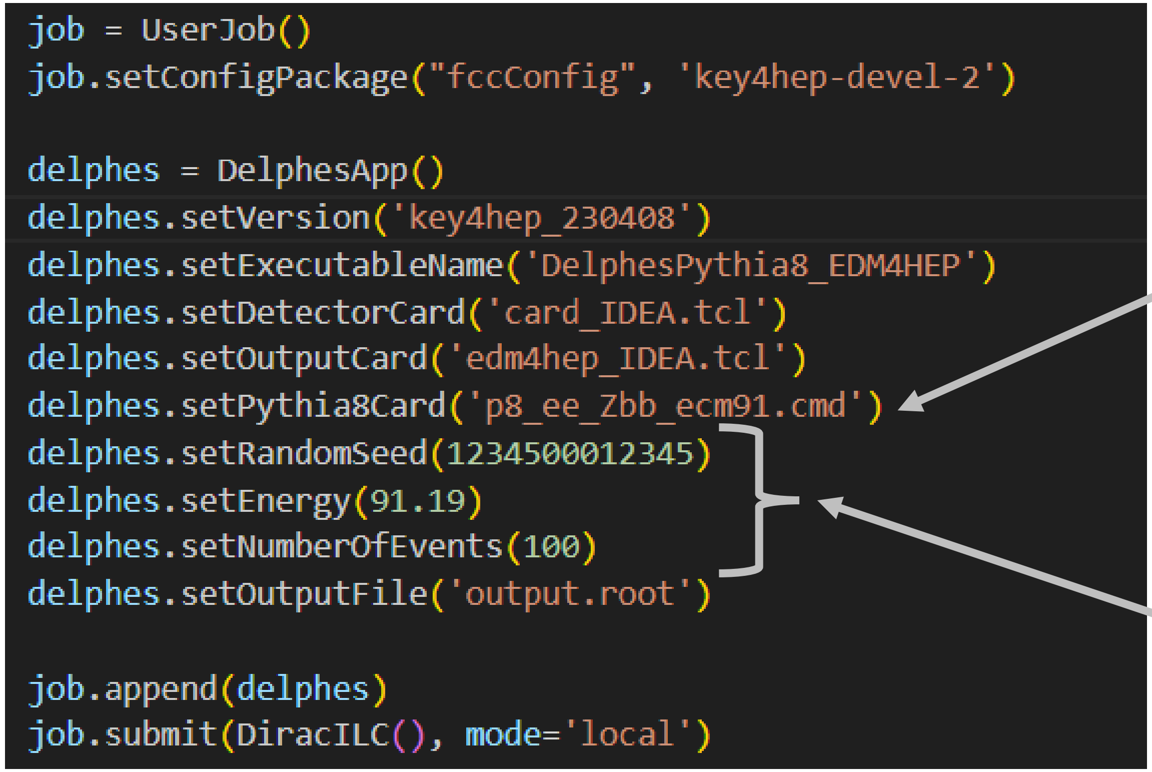
```
job = UserJob()
job.setConfigPackage("fccConfig", 'key4hep-devel-2')

delphes = DelphesApp()
delphes.setVersion('key4hep_230408')
delphes.setExecutableName('DelphesPythia8_EDM4HEP')
delphes.setDetectorCard('card_IDEA.tcl')
delphes.setOutputCard('edm4hep_IDEA.tcl')
delphes.setPythia8Card('p8_ee_Zbb_ecm91.cmd')
delphes.setRandomSeed(1234500012345)
delphes.setEnergy(91.19)
delphes.setNumberOfEvents(100)
delphes.setOutputFile('output.root')

job.append(delphes)
job.submit(DiracILC(), mode='local')
```

Retrieve the corresponding tarball.

Contained in the tarball. Called by their names from the command line.

Goes directly to the command line. In a real job would be **`Zbb_delphes_<…>.root`**.

Delphes App interface



```
job = UserJob()
job.setConfigPackage("fccConfig", 'key4hep-devel-2')

delphes = DelphesApp()
delphes.setVersion('key4hep_230408')
delphes.setExecutableName('DelphesPythia8_EDM4HEP')
delphes.setDetectorCard('card_IDEA.tcl')
delphes.setOutputCard('edm4hep_IDEA.tcl')
delphes.setPythia8Card('p8_ee_Zbb_ecm91.cmd')
delphes.setRandomSeed(1234500012345)
delphes.setEnergy(91.19)
delphes.setNumberOfEvents(100)
delphes.setOutputFile('output.root')

job.append(delphes)
job.submit(DiracILC(), mode='local')
```

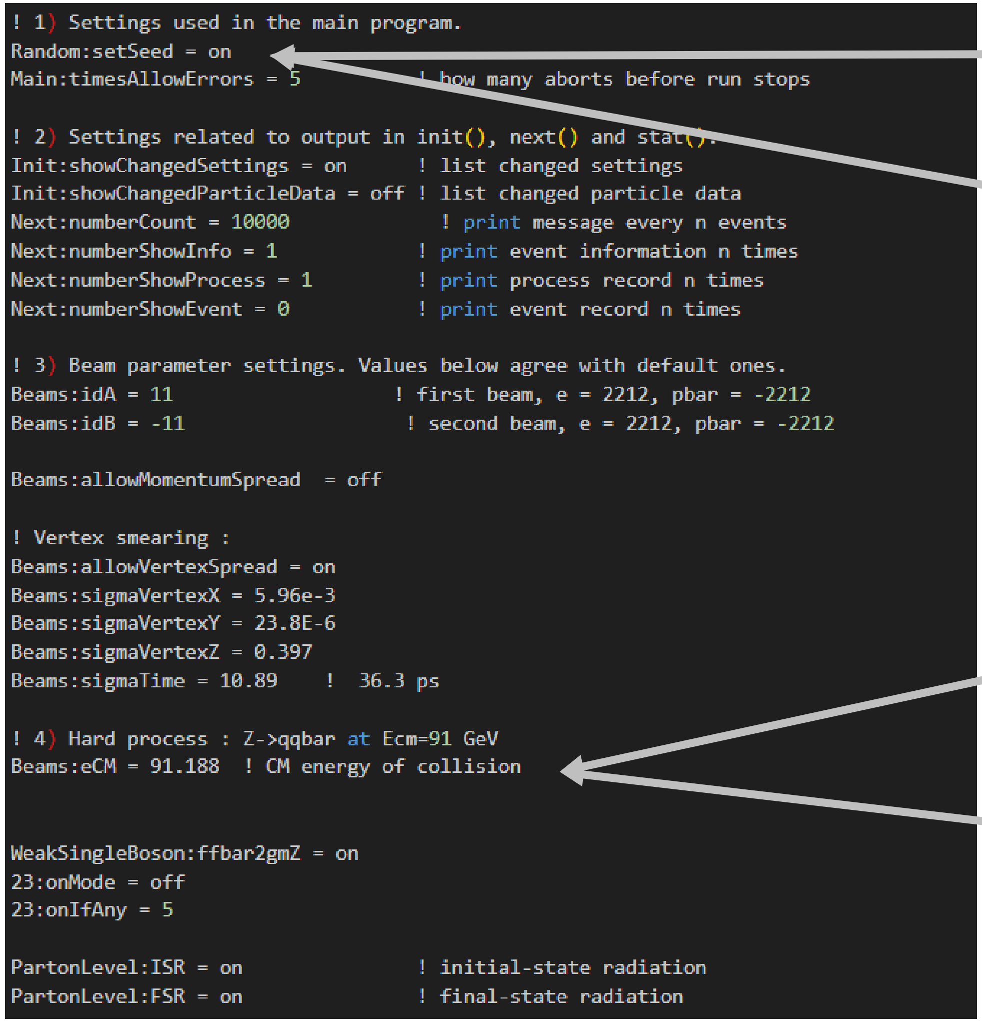
The pythia card is not in the tarball.

It is copied from the user’s local folder as a string and pasted in a new file on the remote machine just before the executable is run.

If not found in the local folder, it is recovered from EOS.

These values are inserted in the pythia card.

Pythia Card Consistency Checks



```
! 1) Settings used in the main program.
Random:setSeed = on
Main:timesAllowErrors = 5          ! how many aborts before run stops

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 10000             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 11                   ! first beam, e = 2212, pbar = -2212
Beams:idB = -11                   ! second beam, e = 2212, pbar = -2212

Beams:allowMomentumSpread  = off

! Vertex smearing :
Beams:allowVertexSpread = on
Beams:sigmaVertexX = 5.96e-3
Beams:sigmaVertexY = 23.8E-6
Beams:sigmaVertexZ = 0.397
Beams:sigmaTime = 10.89     !  36.3 ps

! 4) Hard process : Z->qqbar at Ecm=91 GeV
Beams:eCM = 91.188  ! CM energy of collision


WeakSingleBoson:ffbar2gmZ = on
23:onMode = off
23:onIfAny = 5

PartonLevel:ISR = on               ! initial-state radiation
PartonLevel:FSR = on               ! final-state radiation
```

Insert or replace the random seed value.

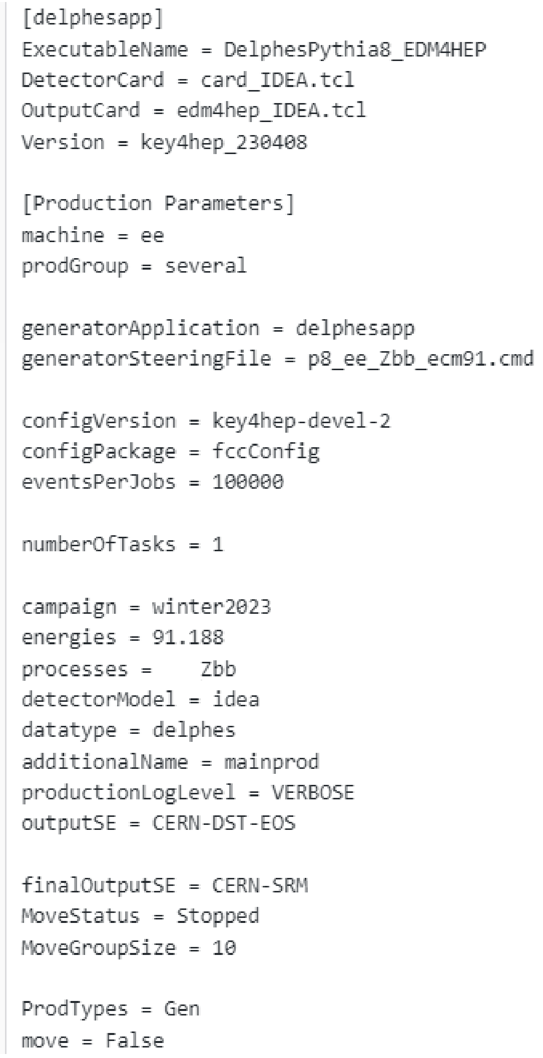
Check if the random seed does not exceed pythia limits (9*10^8)

Insert or replace the number of events to generate

Check that the energy of the card is the same as the one specified from the interface.

Check that the card is for event generation, and not an LHE reader.

Config file

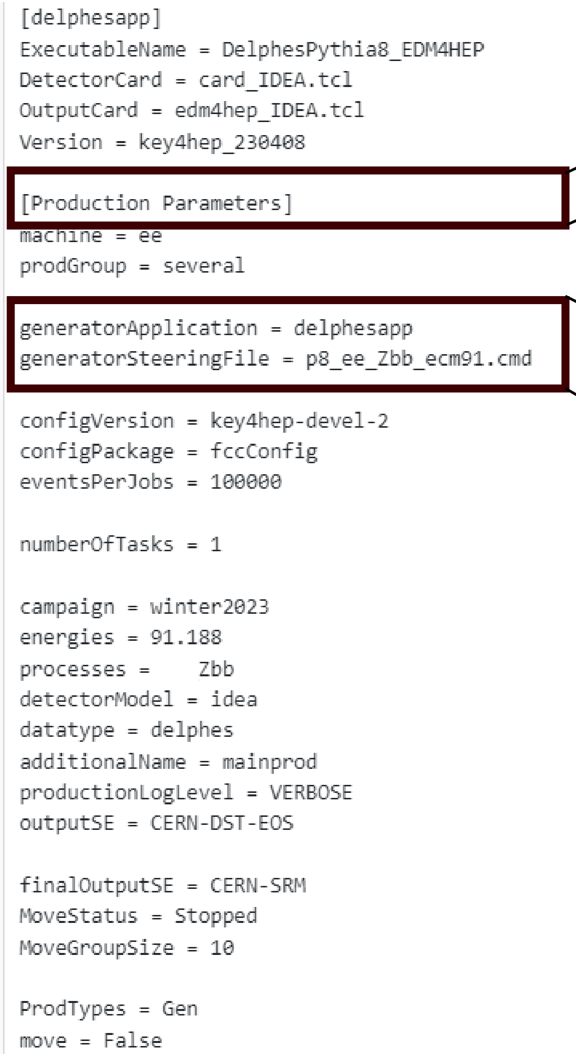


```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408

[Production Parameters]
machine = ee
prodGroup = several

generatorApplication = delphesapp
generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

configVersion = key4hep-devel-2
configPackage = fccConfig
eventsPerJobs = 100000

numberOfTasks = 1

campaign = winter2023
energies = 91.188
processes =    Zbb
detectorModel = idea
datatype = delphes
additionalName = mainprod
productionLogLevel = VERBOSE
outputSE = CERN-DST-EOS

finalOutputSE = CERN-SRM
MoveStatus = Stopped
MoveGroupSize = 10

ProdTypes = Gen
move = False
```

Transformation are submitted on the grid by using this kind of configuration file, which is described in detail in the next slides.

Only privileged users have the possibility to submit transformations in this way.

Config File



```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408

[Production Parameters]
machine = ee
prodGroup = several

generatorApplication = delphesapp
generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

configVersion = key4hep-devel-2
configPackage = fccConfig
eventsPerJobs = 100000

numberOfTasks = 1

campaign = winter2023
energies = 91.188
processes =    Zbb
detectorModel = idea
datatype = delphes
additionalName = mainprod
productionLogLevel = VERBOSE
outputSE = CERN-DST-EOS

finalOutputSE = CERN-SRM
MoveStatus = Stopped
MoveGroupSize = 10

ProdTypes = Gen
move = False
```

[Production Parameters]

Production Parameters affect the whole transformation.

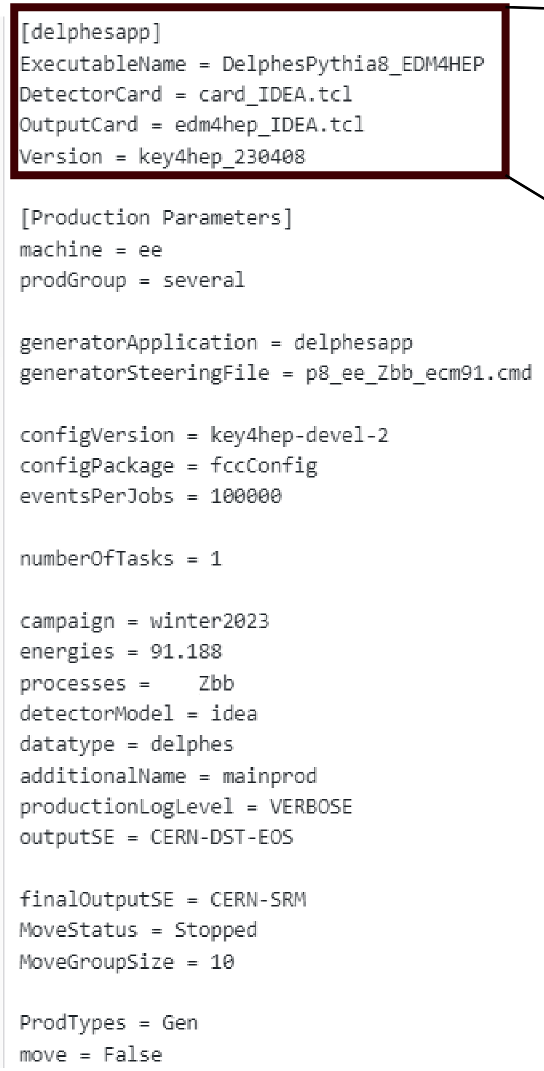
generatorApplication = delphesapp

generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

Setting the application used for the generation process.

Giving the path to the Steering File (in this case the pythia card), or its name on EOS.

Config File



```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408

[Production Parameters]
machine = ee
prodGroup = several

generatorApplication = delphesapp
generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

configVersion = key4hep-devel-2
configPackage = fccConfig
eventsPerJobs = 100000

numberOfTasks = 1

campaign = winter2023
energies = 91.188
processes =    Zbb
detectorModel = idea
datatype = delphes
additionalName = mainprod
productionLogLevel = VERBOSE
outputSE = CERN-DST-EOS

finalOutputSE = CERN-SRM
MoveStatus = Stopped
MoveGroupSize = 10

ProdTypes = Gen
move = False
```

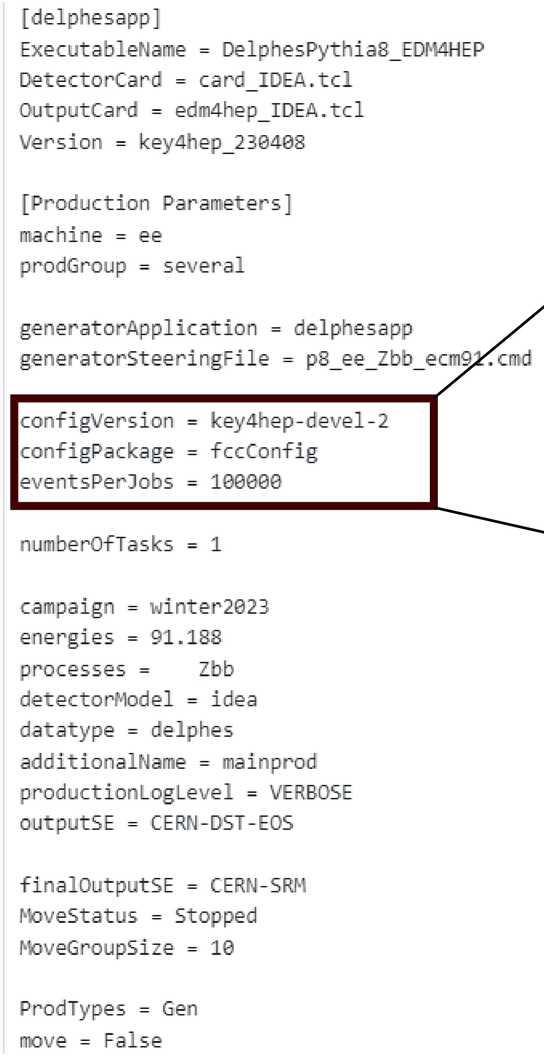
```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408
```

Application specific parameters only affect a specific application in the transformation (in this case Delphes).

Here we choose:

- our Delphes executable
- IDEA Detector Card and Output Card
- Delphes Software Version

Config File



```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408

[Production Parameters]
machine = ee
prodGroup = several

generatorApplication = delphesapp
generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

configVersion = key4hep-devel-2
configPackage = fccConfig
eventsPerJobs = 100000

numberOfTasks = 1

campaign = winter2023
energies = 91.188
processes =    Zbb
detectorModel = idea
datatype = delphes
additionalName = mainprod
productionLogLevel = VERBOSE
outputSE = CERN-DST-EOS

finalOutputSE = CERN-SRM
MoveStatus = Stopped
MoveGroupSize = 10

ProdTypes = Gen
move = False
```

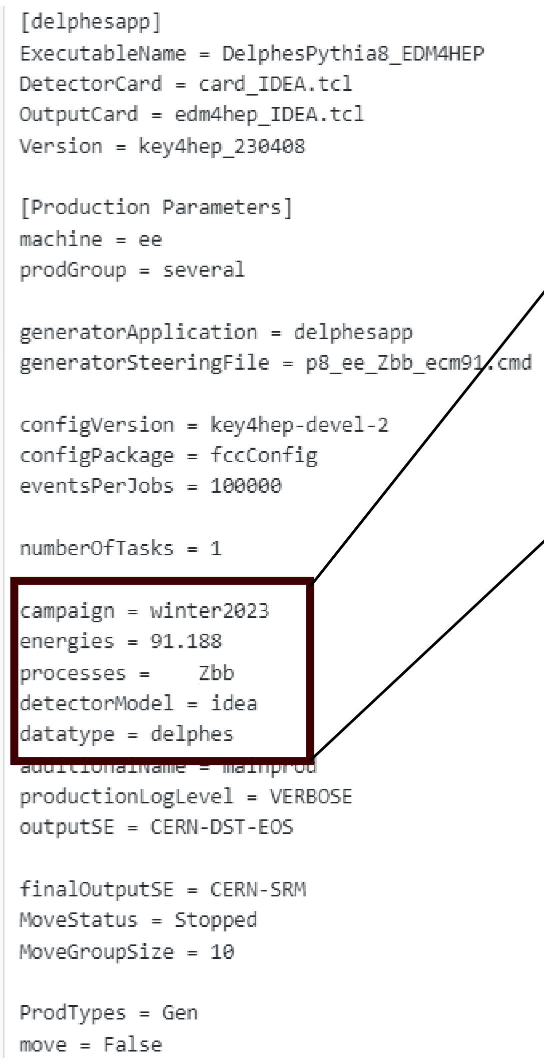
```
configVersion = key4hep-devel-2
configPackage = fccConfig
```

Name of the tarball containing some files required for the transformation (IDEA card and output card).

```
eventsPerJobs = 100000
```

Number of events to generate in a job: 100k. The total number of events will be the product of this by the number of jobs.

Config File



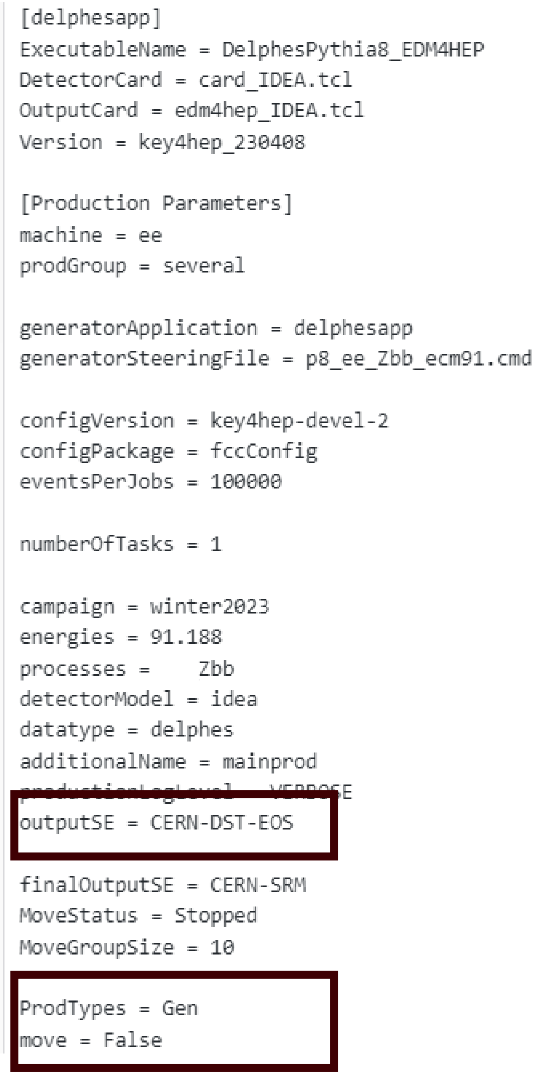
```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408

[Production Parameters]
machine = ee
prodGroup = several

generatorApplication = delphesapp
generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

configVersion = key4hep-devel-2
configPackage = fccConfig
eventsPerJobs = 100000

numberOfTasks = 1

campaign = winter2023
energies = 91.188
processes =     Zbb
detectorModel = idea
datatype = delphes
additionalName = mainprod
productionLogLevel = VERBOSE
outputSE = CERN-DST-EOS

finalOutputSE = CERN-SRM
MoveStatus = Stopped
MoveGroupSize = 10

ProdTypes = Gen
move = False
```

```
Campaign = winter2023
Energies = 91.188
Processes = Zbb
detectorModel = idea
Datatype = delphes
```

These parameters decide the output file path, as:
…/campaign/energy/processes/detectorModel/datatype/

Energies will be compared with the pythia cards contents.

Config File



```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408

[Production Parameters]
machine = ee
prodGroup = several

generatorApplication = delphesapp
generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

configVersion = key4hep-devel-2
configPackage = fccConfig
eventsPerJobs = 100000

numberOfTasks = 1

campaign = winter2023
energies = 91.188
processes =    Zbb
detectorModel = idea
datatype = delphes
additionalName = mainprod
productionLogLevel = VERBOSE
outputSE = CERN-DST-EOS

finalOutputSE = CERN-SRM
MoveStatus = Stopped
MoveGroupSize = 10

ProdTypes = Gen
move = False
```

`outputSE = CERN-DST-EOS`

Setting the Storage Element for the outputs

`ProdTypes = Gen`

`Move = False`

Specifying that our production type is a Generation.

Specifying that we are not going to save the outputs on magnetic tape

Creating Transformations

```
~$ dirac-proxy-init -g fcc_prod
~$ dirac-fcc-make-productions -p > configFile # generate a standard config file (to be personalized)
~$ dirac-fcc-make-productions -f configFile # generate job description xml file (without creating transf.)
~$ dirac-fcc-make-productions -f configFile -x # create transformations
~$ dirac-ilc-add-tasks-to-prod ProdID TasksToAdd [-Total] # extend the number of jobs


~$ dirac-jobexec jobDescription.xml # run production job locally (for debugging purposes)
```

Creating Transformations

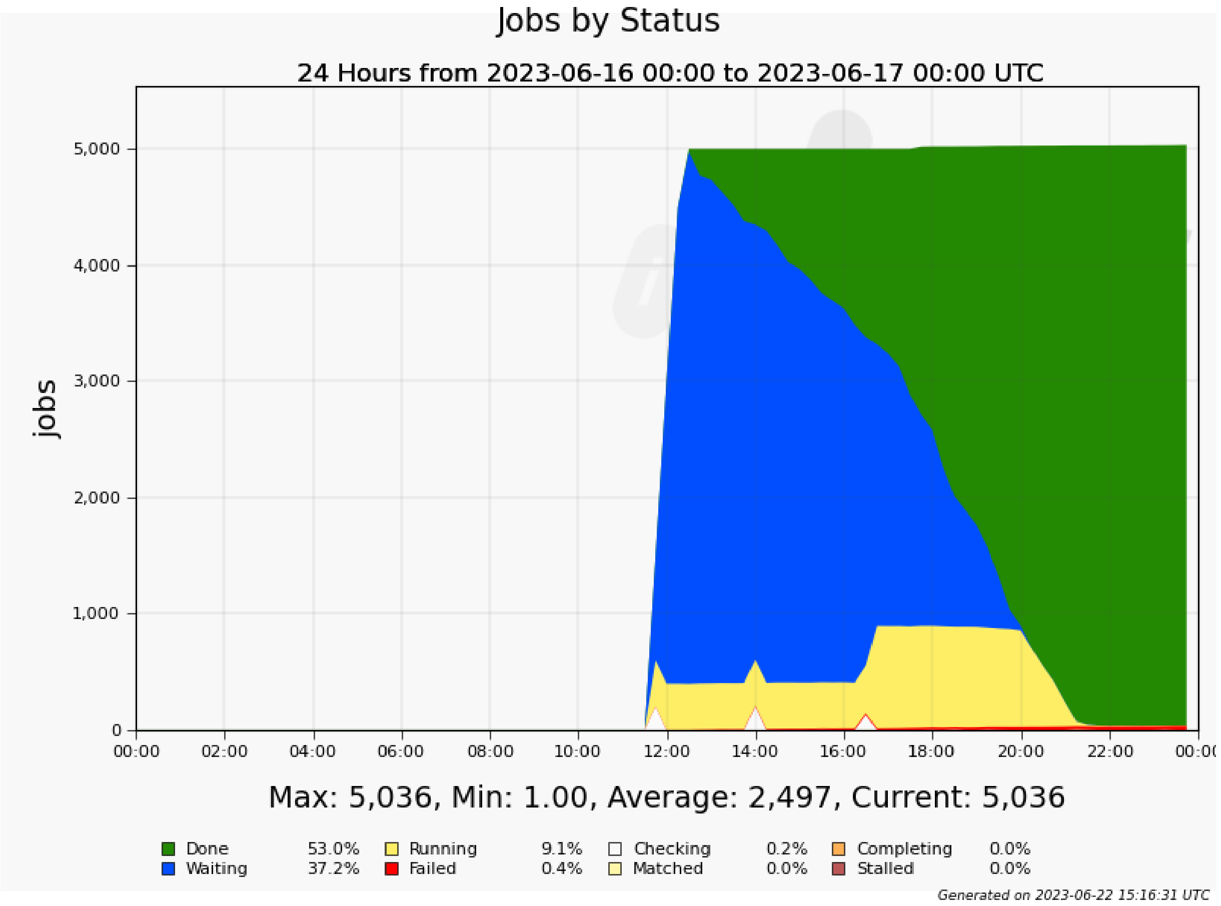
Metadata set at creation of transformation:

```
* /fcc/ee/winter2023/91.19gev/: {'Energy': ‘91.19'}
* /fcc/ee/winter2023/91.19gev/Zbb/: {'EvtType': ‘Zbb'}
* /fcc/ee/winter2023/91.19gev/Zbb/idea: {'DetectorType': 'idea'}
* /fcc/ee/winter2023/91.19gev/Zbb/idea/delphes: {'Datatype': 'delphes'}
* /fcc/ee/winter2023/91.19gev/Zbb/idea/delphes/00016139: {'ProdID’: 16139, 'NumberOfEvents’: 100000}
```

Non searchable metadata set at creation of transformation:

```
{'/fcc/ee/winter2023/91.19gev/Zbb/idea/delphes/00016139': {'SWPackages': 'delphesapp.key4hep-latest'}}
```

Creating Transformations

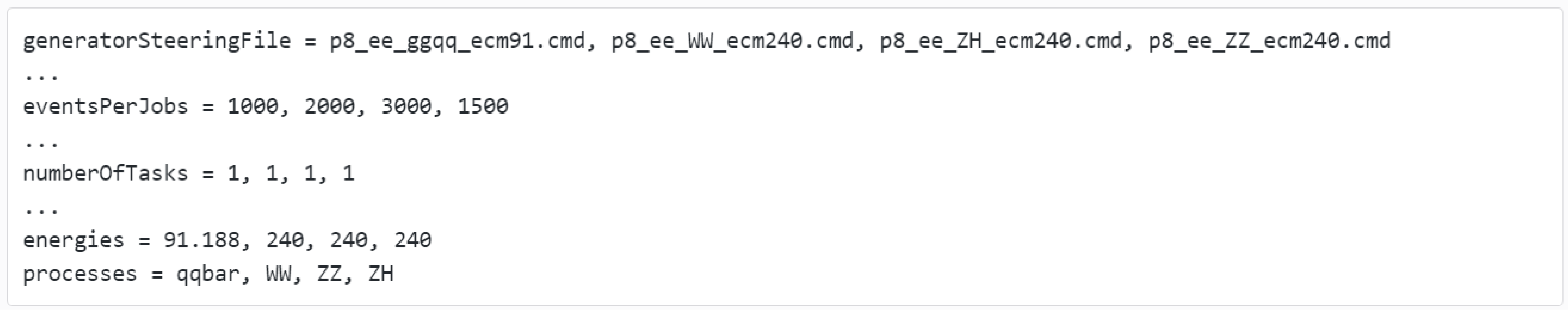


The first practical use of the Delphes interface was the generation of events for the inclusive bb production process.

The transformation described in the config file was extended to 5000 jobs, producing 500M events.

A small number of jobs failed. New jobs were automatically created until reaching 5000 successful jobs.

Multiple Transformations



```
generatorSteeringFile = p8_ee_ggqq_ecm91.cmd, p8_ee_WW_ecm240.cmd, p8_ee_ZH_ecm240.cmd, p8_ee_ZZ_ecm240.cmd
...
eventsPerJobs = 1000, 2000, 3000, 1500
...
numberOfTasks = 1, 1, 1, 1
...
energies = 91.188, 240, 240, 240
processes = qqbar, WW, ZZ, ZH
```

It is possible to create multiple transformations from the same config file. It is sufficient to specify the different pythia cards, energies, processes, events per jobs and number of tasks for each of them.

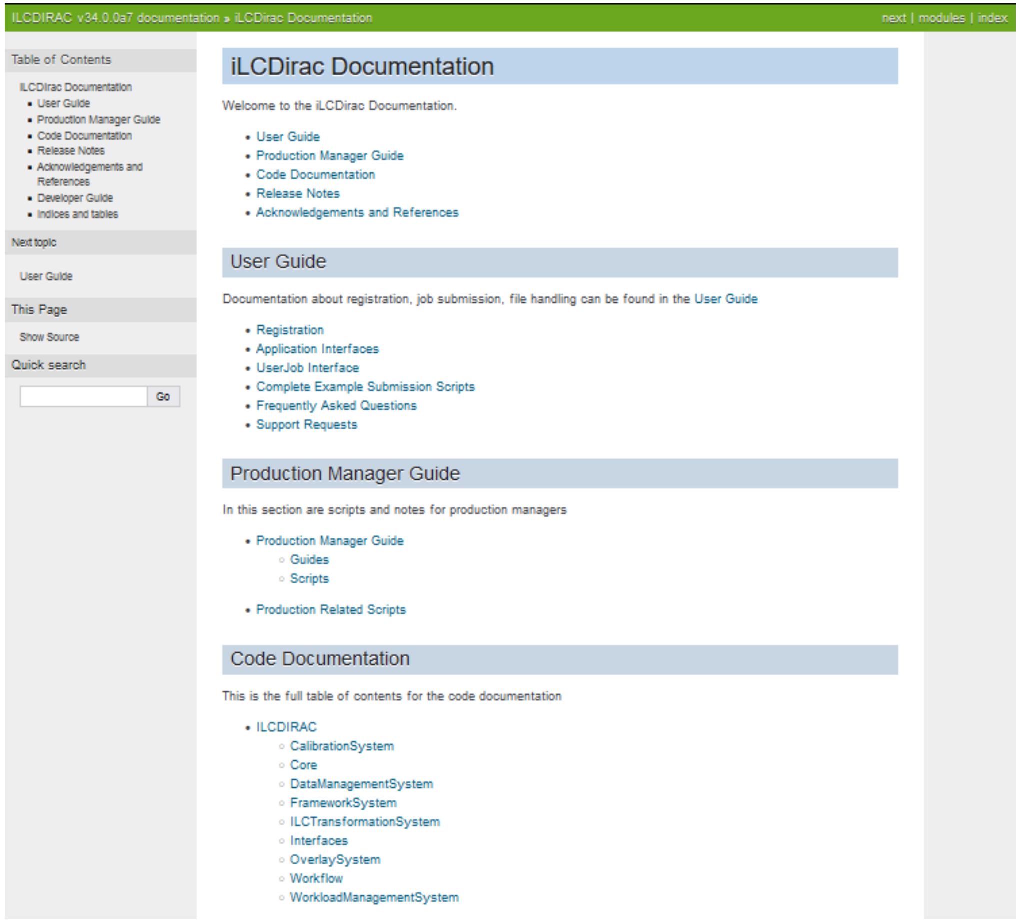
Outputs and production info

```
~$ dirac-ilc-get-info -p 16139 # to get information about the transformation
~$ dirac-ilc-get-prod-log -P 16139 # to download the production logs of then jobs
~$ dirac-dms-find-lfns Path=/ ProdID=16139 Datatype=delphes
~$ dirac-dms-find-lfns Path=/ EvtType=Zbb Energy=91.19 Datatype=delphes
~$ dirac-dms-filecatalog-cli # to access the outputs of the jobs
```

(Most of these actions can also be done through the Web Interface)

Documentation

- http://lcd-data.web.cern.ch/lcd-data/doc/ilcdiracdoc/
- Information about commands (scripts) including options
- API, examples for all applications



iLCDirac Documentation

Welcome to the iLCDirac Documentation.

- User Guide
- Production Manager Guide
- Code Documentation
- Release Notes
- Acknowledgements and References

User Guide

Documentation about registration, job submission, file handling can be found in the User Guide

- Registration
- Application Interfaces
- UserJob Interface
- Complete Example Submission Scripts
- Frequently Asked Questions
- Support Requests

Production Manager Guide

In this section are scripts and notes for production managers

- Production Manager Guide
  - Guides
  - Scripts
- Production Related Scripts

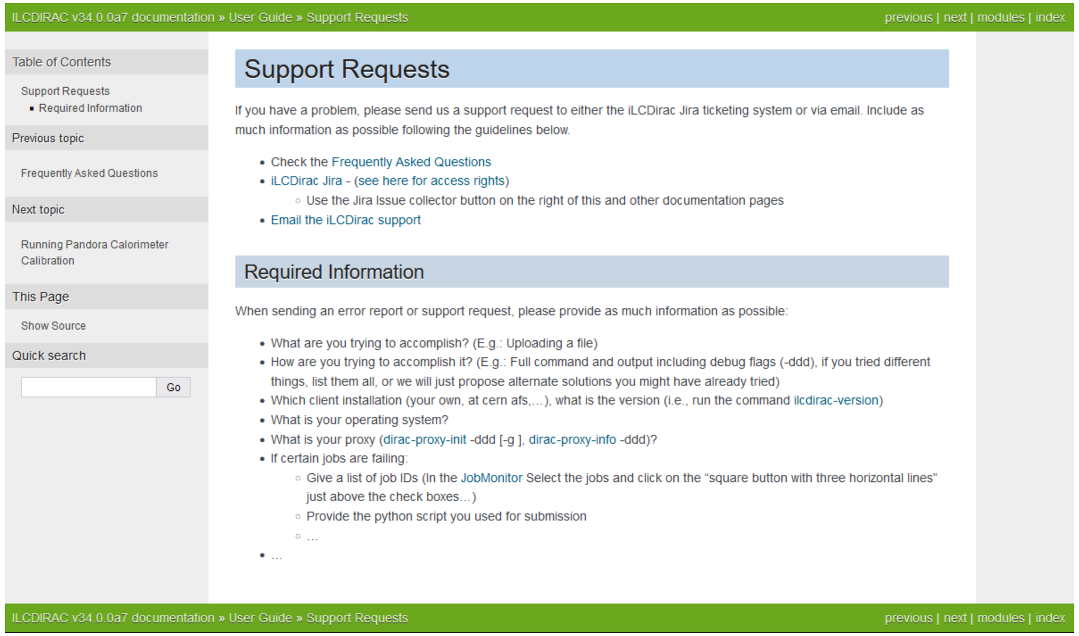
Code Documentation

This is the full table of contents for the code documentation

- ILCDIRAC
  - CalibrationSystem
  - Core
  - DataManagementSystem
  - FrameworkSystem
  - ILCTransformationSystem
  - Interfaces
  - OverlaySystem
  - Workflow
  - WorkloadManagementSystem

Support

- ▶ In case of fire:
  1. Consult documentation: http://lcd-data.web.cern.ch/lcd-data/doc/ilcdiracdoc/
  2. Before submitting a ticket, see: http://lcd-data.web.cern.ch/lcd-data/doc/ilcdiracdoc/DOC/Files/UserGuide/support.html
  3. Submit a ticket to the issue tracker https://its.cern.ch/jira/browse/ILCDIRAC
- ▶ See also "Report a Problem" buttons in web portal and documentation
  4. Email: ilcdirac-support@cern.ch


ILCDIRAC v34.0.0a7 documentation » User Guide » Support Requests
previous | next | modules | index
Table of Contents
Support Requests
Required Information
Previous topic
Frequently Asked Questions
Next topic
Running Pandora Calorimeter Calibration
This Page
Show Source
Quick search
Go

Support Requests

If you have a problem, please send us a support request to either the iLCDirac Jira ticketing system or via email. Include as much information as possible following the guidelines below.

- Check the Frequently Asked Questions
- iLCDirac Jira - (see here for access rights)
  - Use the Jira Issue collector button on the right of this and other documentation pages
- Email the iLCDirac support

Required Information

When sending an error report or support request, please provide as much information as possible:

- What are you trying to accomplish? (E.g.: Uploading a file)
- How are you trying to accomplish it? (E.g.: Full command and output including debug flags (-ddd), if you tried different things, list them all, or we will just propose alternate solutions you might have already tried)
- Which client installation (your own, at cern afs,…), what is the version (i.e., run the command ilcdirac-version)
- What is your operating system?
- What is your proxy (dirac-proxy-init -ddd [-g], dirac-proxy-info -ddd)?
- If certain jobs are failing:
  - Give a list of job IDs (In the JobMonitor Select the jobs and click on the "square button with three horizontal lines" just above the check boxes…)
  - Provide the python script you used for submission
  - …
- …

ILCDIRAC v34.0.0a7 documentation » User Guide » Support Requests
previous | next | modules | index