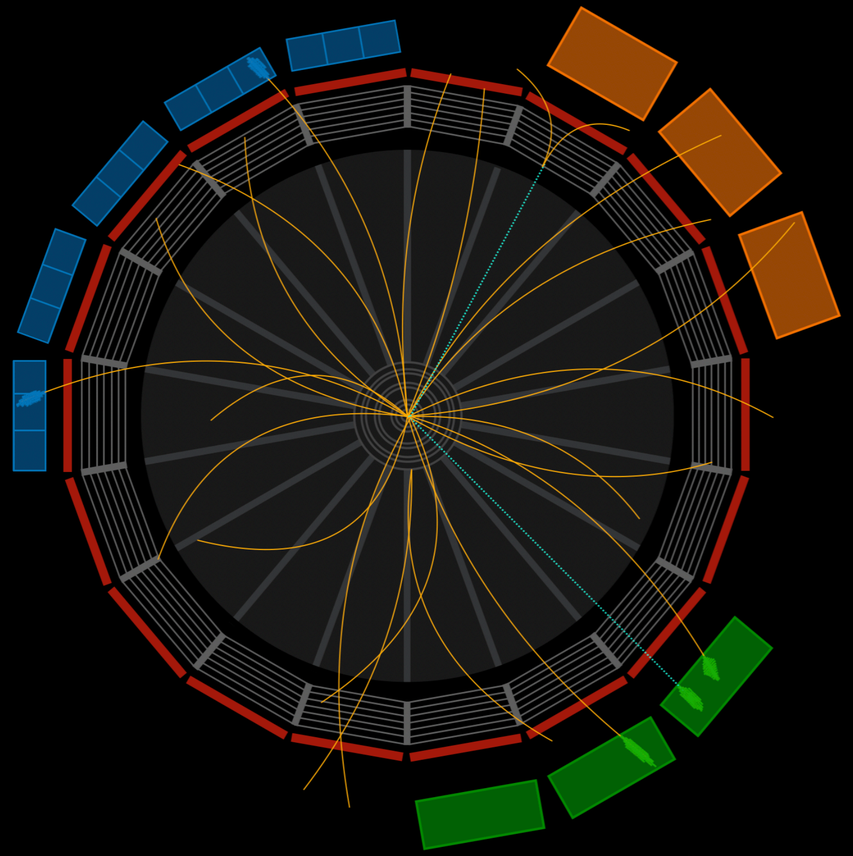
Tutorials on how to run MC simulations with O2DPG



From zero to hero

Alberto Calivà for the DPG

CERN

DPG

ALICE • Data Preparation Group

ALICE

New policy for MC requests

New policy for MC requests presented by the DPG at the Physics Forum of 17 January [Link]

1. Software updates should be prepared and tested by the requesting PWG

2. When ready, a tag will be created following the instructions by the PWG.
   This should be used by the requesting PWG to run a test on the Grid.

3. When the test is successful, a request at the PB should be made showing
   the first QC from the test production

4. Once the request is approved, a JIRA ticket can be opened. Mandatory fields:

- Expected storage (in TB), running time (in days at 10k CPU) and number of events requested
- Link to the GRID folder with configuration/JDL and to AODs
- QA responsible (if not specified, the PWG conveners will be added automatically)
- pdf of presentation at the PB (the production manager cannot access the PB)

PB approval only if running time > 1d@10kCPU

General information

How to get in touch with the simulation developers

- Simulation e-group (for meeting announcements) + WP12 meetings
- Mattermost channels (preferred over private email): O2-simulation + O2DPG
- JIRA tickets for feature requests/bug reports (components simulation or O2DPG)

Where to find information about simulation

- Online documentation: https://aliceo2group.github.io/simulation/
- Documentation in AliceO2: DetectorSimulation.md
- Some info in O2DPG: WorkflowRunner.md
- Various examples at O2/SimExamples or nightly-tests
- Anchored MC: https://aliceo2group.github.io/simulation/docs/o2dpgworkflow/anchored.html

Installing O2DPG

In order to use O2DPG, you need to install it in your local machine
(Assuming here that you have O2 already installed)

Download the software:

```
$ cd ~/alice
$ aliBuild init O2DPG
```

Build the software:

```
$ aliBuild build O2sim --defaults o2
```

Load the environment:

```
$ alienv enter O2sim/latest
```

Alternatively, you could build O2PDPSuite

→ meta package steering the build of everything that one could "possibly" need for simulation

LXPLUS

If you don't have a working software locally
→ connect to the LXPLUS cluster

- LXPLUS (Linux Public Login User Service) is the interactive logon service to Linux for all CERN users.
- The cluster LXPLUS consists of public machines provided by the CERN IT Department for interactive work
- Detailed documentation maintained by the IT Department is available here

Use precompiled packages on LXPLUS

Connect to lxplus using Secure Shell protocol (SSH):

```
albertocaliva@MacBook-Pro-AlbertoCaliva1-3 ~ % ssh -X alcaliva@lxplus.cern.ch
(alcaliva@lxplus.cern.ch) Password:
```

Enter the password of your CERN account
(The same as your CERN e-mail)
N.B. while typing the password characters
are invisible

Load the environment using the nightly precompiled builds:

```
$ alienv enter O2sim::v20240120-1
```

Version of
20 January 2024

You could list all available packages

```
$ alienv q
```

→ It takes ages !

Grid certificate

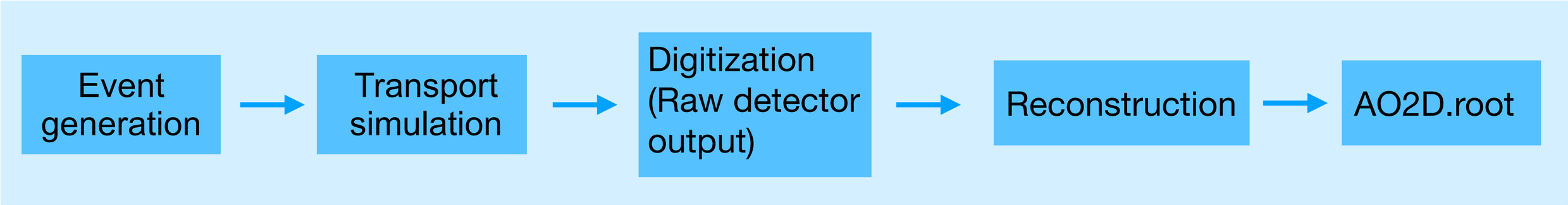
To run MC simulations with O2DPG you need a valid GRID certificate
→ needed to access information on the **CCDB** file

The *Condition* and *Calibration Data Base* (CCDB) is a ROOT file used to store calibration and alignment data (centrality, TPC splines, calibration maps for space-charge distortions, …)

More info at this link

Instructions on how to download your GRID user certificate and import it to your browser in backup

MC simulations: full chain



Running MC simulations with O2DPG is extremely simple:

→ 1 script + 1 line of commands on the terminal

Quick start: usage of o2-sim

Load the environment:

```
$ alienv enter O2sim/latest
```

Get a token (valid 24h)

```
[O2Sim/latest] ~ %> alien-token-init
Enter PEM pass phrase:
DN >>> C=ch/O=AliEn2/CN=Users/CN=alcaliva/OU=alcaliva
ISSUER >>> C=ch/O=AliEn2/CN=AliEn CA
BEGIN >>> 2024-01-26 14:01:56
EXPIRE >>> 2024-02-26 16:01:56
```

Enter the password of your GRID User certificate
(N.B. the characters are invisible)

First example:

```
[O2Sim/latest] ~ %>  o2-sim -e TGeant4 -g pythia8pp -n 10 -j 2
```

Simulator

"engine": transport code
(Geant4 is the default)

Generator

Number of events

Number of parallel simulation workers

o2-sim options

List all options and show defaults:

```
[O2Sim/latest] ~ %> o2-sim -h
```

Main options:

- `-e [ --mcEngine ] arg (=TGeant4)` → transport code (default GEANT4)
- `-g [ --generator ] arg (=boxgen)` → event generator
- `-t [ --trigger ] arg` → event generator trigger
- `-m [ --modules ] arg (=all modules)` → modules in the geometry (contribute to mat. budget)
- `--skipModules arg` → modules excluded (precendence over `-m`)
- `--readoutDetectors arg` → list of detectors creating hits (default=all)
- `--skipReadoutDetectors arg` → skip hit creation for these detectors
- `-n [ --nEvents ] arg (=0)` → number of events

o2-sim options

- `--configKeyValues arg` → semicolon separated list of configuration keys = strings that correspond to configuration parameters (e.g.: 'TPC.gasDensity=1;...
- `--configFile arg` → /path_to/configuration_file.ini/json
- `--seed arg (=0)` → initial seed (default: 0 == random)
- `--field arg (=-5)` → Magnetic field in kGauss, allowed values +-2,+-5 and 0; +-<intKGaus>U for uniform field; "ccdb" for taking it from CCDB
- `-j [ --nworkers ] arg (=5)` → number of parallel simulation workers

Log files

Example:

```
[O2Sim/latest] ~ %> o2-sim -e TGeant4 -g pythia8pp -m PIPE ITS TPC TOF --
readoutDetectors TPC -n 10 --field -5 -j 2 --noemptyevents --configKeyValues
"PrimaryGenerator.id=202"
```

o2-sim produces 3 internal log files that are useful to see what is going on and to trouble-shoot

- `o2sim_serverlog` → produced by the generator (list of particles and parents, their properties, activated processes, settings, …)
- `o2sim_workerlog0` → produced by the propagation code (Geant4)
- `o2sim_mergerlog` → produced by the *merger service* when merging all the hits produced by the activated detectors and writing them into corresponding files (e.g. `o2sim_HitsITS.root, o2sim_HitsTOF.root,...`)

Kinematics file

`o2sim_Kine.root` contains information on

1. the generated particles, their properties (p_x,p_y,p_z, E, …), parent, …
2. secondary particles produced during transport

meta-information about each generated event is available in a separate file:
`o2sim_MCHeader.root`

Reading and navigating manually through kinematics can be cumbersome
(“ROOT-IO boilerplate”)

→ Offer 2 main utility classes making this easy for users

- MCKinematicsReader → Class to read and retrieve tracks for given event or a Monte Carlo label
- MCTrackNavigator → Class to navigate through mother-daughter tree of MC tracks and to query physics properties

Generators implemented in O2

Pythia8 is the mostly used generator (recommended whenever possible)

Pythia8 needs a configuration file (.cfg):

```
[O2Sim/latest] ~ %> o2-sim -g pythia8 --configKeyValues
"GeneratorPythia8.config=/path_to/pythia8.cfg"
```

Indicate the location (path) of the configuration file

Name of the configuration file

```
### random
Random:setSeed = on
Random:seed = 0

### beams
Beams:idA = 2212
Beams:idB = 2212
Beams:eA = 6800.000000
Beams:eB = 6800.000000

### processes
SoftQCD:inelastic = on


### decays
ParticleDecays:limitTau0 = on
ParticleDecays:tau0Max = 10.
```

Example of a configuration file

Generators implemented in O2

To produce the configuration file you could use the script mkpy8cfg.py

Load the environment:

```
$ alienv enter O2sim/latest
```

Run the script with the required parameters:

```
[O2sim/latest] ~ %> ${O2DPG_ROOT}/MC/config/common/pythia8/utils/mkpy8cfg.py --output=/
path_to_your_dir/pythia8.cfg --seed=0 --idA 2212 --idB 2212 --eA 6800.0 --eB 6800.0 --
process inel
```

This will produce the pythia8.cfg file that you
can use to run your desired pythia8 simulation

Generators implemented in O2

Preconfigured pythia8 for pp collisions:

```
[O2sim/latest] ~ %>  o2-sim -g pythia8pp
```

```
### beams
Beams:idA 2212   # proton
Beams:idB 2212.  # proton
Beams:eCM 14000. # GeV

### processes
SoftQCD:inelastic on # all inelastic processes

### decays
ParticleDecays:limitTau0 on
ParticleDecays:tau0Max 10.
```

Generators implemented in O2

Preconfigured pythia8 for heavy-ion collisions:

```
[O2sim/latest] ~ %>  o2-sim -g pythia8hi
```

```
### beams
Beams:idA 1000822080              # Pb
Beams:idB 1000822080              # Pb
Beams:eCM 5520.                   # GeV

### heavy-ion settings (valid for Pb-Pb 5520 only)
HeavyIon:SigFitNGen 0
HeavyIon:SigFitDefPar 14.82,1.82,0.25,0.0,0.0,0.0,0.0,0.0
HeavyIon:bWidth      15.                  # impact parameter from 0-x [fm]

### processes (apparently not to be defined)

### decays
ParticleDecays:limitTau0 on
ParticleDecays:tau0Max 10.
```

Generators implemented in O2

Preconfigured pythia8 for pp collisions for heavy flavor

```
[O2sim/latest] ~ %>  o2-sim -g pythia8hf
```

```
### beams
Beams:idA 2212                          # proton
Beams:idB 2212                          # proton
Beams:eCM 14000.                  # GeV

### processes
HardQCD:hardccbar on              # scatterings g-g / q-qbar -> c-cbar
HardQCD:hardbbbar on              # scatterings g-g / q-qbar -> b-bbar

### decays
ParticleDecays:limitTau0 on
ParticleDecays:tau0Max 10.
```

Generators implemented in O2

Preconfigured pythia8 for pp collisions for heavy flavor

```
[O2sim/latest] ~ %>  o2-sim -g pythia8hf
```

```
### beams
Beams:idA 2212                                  # proton
Beams:idB 2212                                  # proton
Beams:eCM 14000.                    # GeV

### processes
HardQCD:hardccbar on                # scatterings g-g / q-qbar -> c-cbar
HardQCD:hardbbbar on                # scatterings g-g / q-qbar -> b-bbar

### decays
ParticleDecays:limitTau0 on
ParticleDecays:tau0Max 10.
```

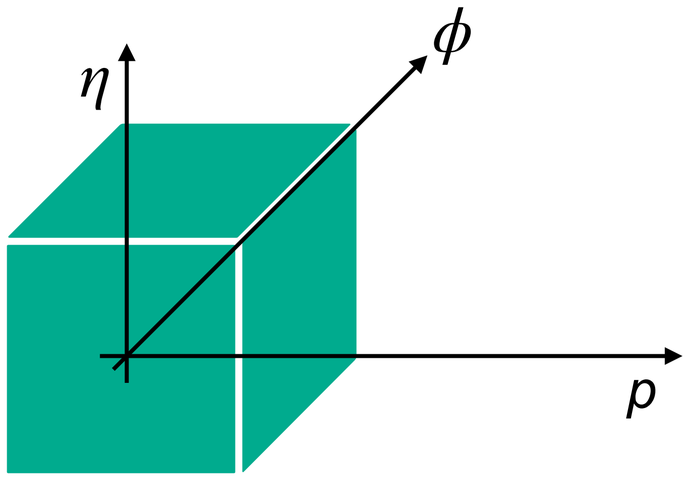
Codes for all processes can be found in the latest Pythia manual

Box Generator

Box generator:

- simple mono-PDG particle generator
- produces particles with uniform distributions in p,η,ϕ

→ that's why it is called *box* (distribution of 3d (p,η,ϕ) points looks like a box)




```
[O2sim/latest] ~ %> o2-sim -e TGeant4 -g boxgen -m PIPE ITS --readoutDetectors ITS
-n 1 --field -5 -j 2 --noemptyevents --configKeyValues "BoxGun.pdg=211 ;
BoxGun.eta[0]=-0.5 ; BoxGun.eta[1]=0.5; BoxGun.number=5 ; BoxGun.prange[0]=0.5 ;
BoxGun.prange[1]=4.5 ; BoxGun.phirange[0]=0.0 ; BoxGun.phirange[1]=180.0"
```

You can find the parameters and their default values here

Extkin02 Generator

`extkin02` takes particles from an existing MC kinematics file and simulates only the transport process

Such a kinematics file is produced always by a (previous) run of o2-sim.


```
[O2sim/latest] ~ %> o2-sim --noGeant -g boxgen -n 1 --configKeyValues "BoxGun.pdg=211
; BoxGun.eta[0]=-0.5 ; BoxGun.eta[1]=0.5; BoxGun.number=5 ; BoxGun.prange[0]=0.5 ;
BoxGun.prange[1]=4.5 ; BoxGun.phirange[0]=0.0 ; BoxGun.phirange[1]=180.0"
```

```
[O2sim/latest] ~ %> o2-sim -g extkin02 -e TGeant4 -m PIPE ITS --readoutDetectors ITS --
extKinFile /path_to/o2sim_Kine.root
```

Reading particles from an HepMC file

An HepMC file is “an object oriented event record written in C++ for High Energy Physics Monte Carlo Generators”.

HepMC files provide a convenient and universal format for storing all information from MC event generators.

It is possible to read primary particles to be transported from an existing HepMC file by specifying "hepmc" as an option for the generator:

```
[O2sim/latest] ~ %> o2-sim -g hepmc --configKeyValues "HepMC.fileName=/path_to/file.hepmc"
```

External Generators

- PWG specific generators belong to the category of "external" generators
- These are just-in-time ROOT macros → Only when ROOT (or the LLVM machinery) digests them, they are translated into machine code

General syntax:

Need to specify that the generator is external

```
[O2sim/latest] ~ %> o2-sim -g external --configKeyValues
'GeneratorExternal.fileName=path_to/generator_name.C;
GeneratorExternal.funcName=function(par1,par2,…)'
```

Path and name of the generator

name of the function to generate the particles according to the desired scheme

Example:

```
[O2sim/latest] ~ %> o2-sim -n 2 -g external -m "PIPE ITS TPC" --configKeyValues
'GeneratorExternal.fileName=${O2DPG_ROOT}/MC/config/PWGLF/pythia8/
generator_pythia8_LF.C;GeneratorExternal.funcName=generateLF({1000010020,
1000010030}, {10, 10}, {0.5, 0.5}, {10, 10})'
```

Triggering

It is possible to select events with the property of interest
(presence of a given particle, decay channel, particles inside acceptance, multiplicity, …)

General syntax:

```
[O2sim/latest] ~ %> o2-sim -g pythia8pp -t external --configKeyValues
'TriggerExternal.fileName=path_to/triggerMacro.C;
TriggerExternal.funcName=triggerFunction(par1,par2,…)'
```

The trigger function inspects the vector of all generator particles and returns true if the event is of interest

Triggering

External generator, trigger and other settings can be all specified in one single .ini file that must be located inside the folder ${O2DPG_ROOT}/MC/config/PWGxy/ini

→ Recommended method

GeneratorHF_ccbarToDielectrons.ini

```
### The setup uses an external event generator
### This part sets the path of the file and the function call to retrieve it

[GeneratorExternal]
fileName = ${O2DPG_ROOT}/MC/config/PWGEM/external/generator/GeneratorCharmToEle_EvtGen.C
funcName = GeneratorCharmToEle_EvtGen(-1.5,1.5)

### The external generator derives from GeneratorPythia8.
### This part configures the bits of the interface: configuration and user hooks

[GeneratorPythia8]
config = ${O2DPG_ROOT}/MC/config/common/pythia8/generator/pythia8_pp_cr2.cfg
hooksFileName = ${O2DPG_ROOT}/MC/config/PWGHF/pythia8/hooks/pythia8_userhooks_qqbar.C
hooksFuncName = pythia8_userhooks_ccbar(-1.5,1.5)

### The setup uses an external even generator trigger which is
### defined in the following file and it is retrieved and configured
### according to the specified function call

[TriggerExternal]
fileName = ${O2DPG_ROOT}/MC/config/PWGDQ/trigger/selectDaughterFromHFwithinAcc.C
funcName = selectDaughterFromHFwithinAcc(11,kFALSE,-1.,1.)
```

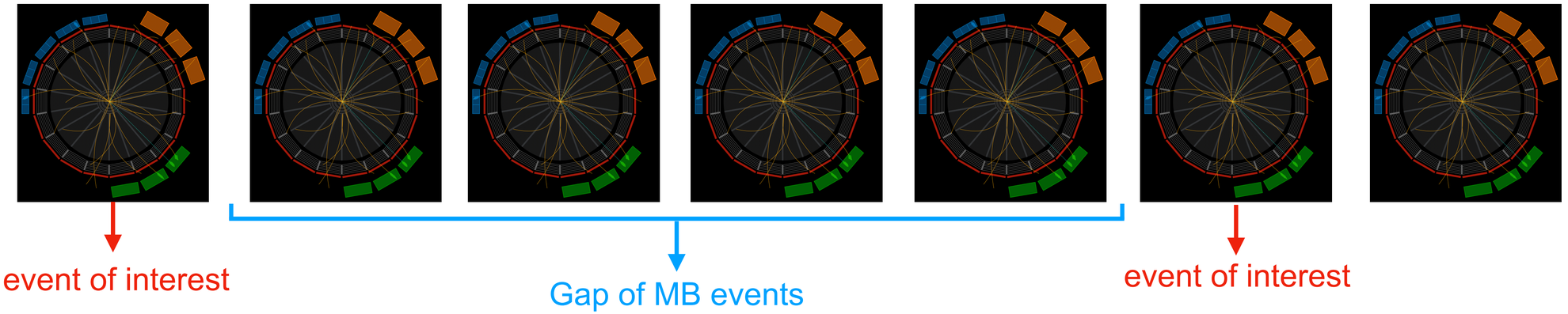
Example:

```
[O2sim/latest] ~ %> o2-sim -n 2 -g external -t external --configFile ${O2DPG_ROOT}/MC/config/PWGEM/ini/GeneratorHF_ccbarToDielectrons.ini
```

Gap-triggered generators

"Gap-triggered" generator:

- The event of interest is selected only every *n* events.
- The "gap" in between is filled with minimum bias events.



Events of interest and MB events are identified using different sub-generator IDs:

- `mcCollision::getSubGeneratorId()=0` → event of interest
- `mcCollision::getSubGeneratorId()=1` → gap (MB event)

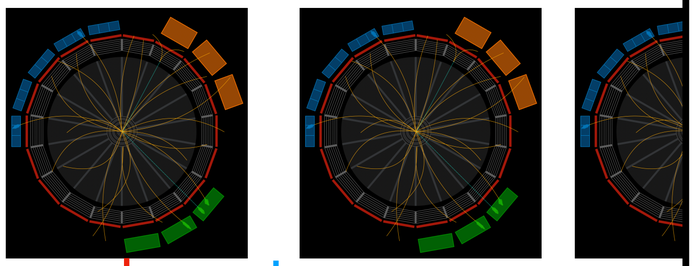
Feature used to mimic real data-taking conditions with continuous readout

→ need to properly "dilute" events of interest into MB events (optimal gap/trigger to be studied)

Gap-triggered generators

"Gap-triggered" generator:

- The event of interest is selected
- The "gap" in between is filled w



event of interest

GeneratorHFTrigger_Xi_Omega_C.ini

```
### The external generator derives from GeneratorPythia8.
[GeneratorExternal]
fileName=${O2DPG_ROOT}/MC/config/PWGHF/external/generator/generator_pythia8_gaptriggered_hf.C
funcName=GeneratorPythia8GapHF(5,-1.5,1.5,-1.5,1.5,{4,5}, {4132, 4332})

[GeneratorPythia8]
config=${O2DPG_ROOT}/MC/config/PWGHF/pythia8/generator/pythia8_charmtriggers_xicomegac.cfg
```

Events of interest and MB events

- `mcCollision::getSubGenerat`
- `mcCollision::getSubGenerat`

Feature used to mimic real data-ta

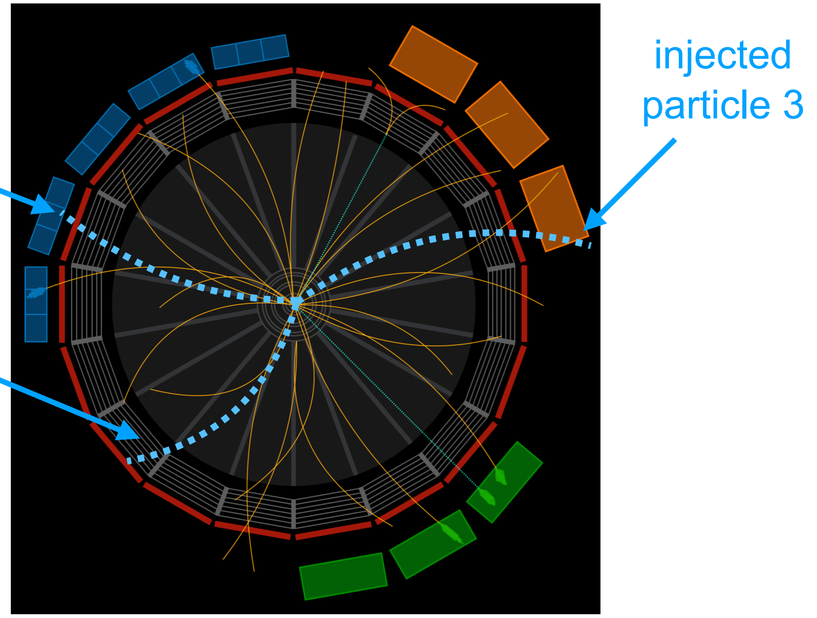
→ need to properly "dilute" events of interest into MB events (optimal gap/trigger to be studied)

Embedding

It is possible to inject particles of interest on top of an underlying event (generated with Pythia, Hijing, …)

Injected particles and particles from the underlying event have a different source ID:

- `mcCollision::getSourceId()=0` → injected particles
- `mcCollision::getSourceId()=1` → background particles

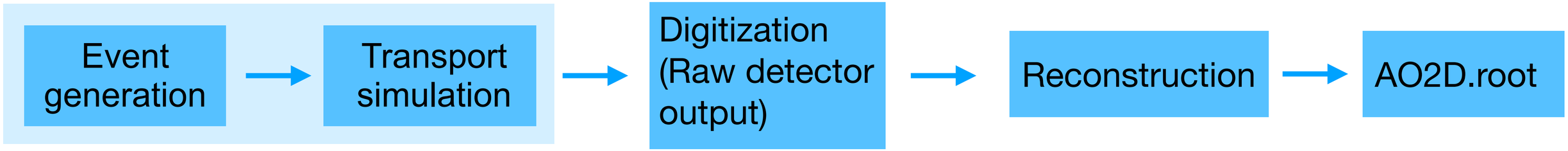


ATTENTION:
Real events containing particles of interest might have different properties than minimum-bias events + injected particles of interest

→ running your efficiency task on simulated events (MB + injected particles) might introduce a bias !

Recommendations: study real event properties in case a particle of interest is present, quantify the potential bias and tune the injection scheme or properties of the simulated underlying event

Digitization and reconstruction



o2-sim is used only for the first two steps

In order to produce simulated AODs, we need to go beyond o2-sim and event generation and run the complete pipeline including **digitization** and **reconstruction** steps

Very complex system, consisting of many executables or tasks, requiring consistent application and propagation of settings/configuration to work together

Achieved by creating and running workflows

Running MC jobs on the GRID

To run MC simulations on the GRID from generation to AOD

→ write a shell script (my_script.sh) with two lines:

```
${O2DPG_ROOT}/MC/bin/o2dpg_sim_workflow.py -eCM 13600 -col pp -gen pythia8
-proc cdiff -tf 1 -ns 200 -e TGeant4 -interactionRate 500000

${O2DPG_ROOT}/MC/bin/o2_dpg_workflow_runner.py -f workflow.json -tt aod
-cpu-limit 8
```

Target Task: AOD creation

- The first line is used to create the workflow and produces a .json file
- The second script executes the workflow

Submit jobs on the GRID

Submit jobs on the GRID:

```
[O2sim/latest] ~ %> ${O2DPG_ROOT}/GRID/utils/grid_submit.sh --script my_script.sh --
jobname test --outputspec "*.log@disk=1","*.root@disk=2" --packagespec
"VO_ALICE@O2sim::v20240103-1" --wait --fetch-output
```

Assign a name (this will appear on MonALISA)

After launching the jobs on the GRID the system waits until they are DONE and then downloads the output files to your local disc automatically.

Specification of files you would like to have saved (everything else is deleted immediately in the workspace)
The disk specifier denotes the number of replicas that are saved for security reasons:

- 1 replica for the log files
- 2 replicas for the ROOT files

To keep root files produced by reconstruction add:
`"tf*/*.root@disk=2"`

The usage of the workflows requires at least 16 GB of RAM and an 8-core machine
You could try to fool the runner (if you are running pp collisions, most probably you will never reach the limit):

```
% o2_dpg_workflow_runner.py -f workflow.json -tt aod --mem-limit 16000
```

Output location

```
Waiting for jobs to return ... Last status : - SJob done
Fetching results
Fetching result files for subjob 1 into /tmp/alien_work/test-20240204-082533
```

This is where the output is downloaded

It is not in the current working directory!

Estimate expected resources

1. Running time can be found in the `logtmp` file

```
[O2sim/latest] ~ %> cd /tmp/alien_work/test-20240204-082533
[O2sim/latest] ~ %> cd 001
[O2sim/latest] ~ %> vi logtmp_3014395312.txt
```

```
**** Pipeline done success (global_runtime : 533.892s) *****
```

$$\text{Expected running time} = \frac{N_{\text{events}}^{\text{target}}}{N_{\text{events}}^{\text{test}}} \times \Delta t_{\text{test}} \times \frac{N_{\text{parallel workers}}}{10\ 000} \qquad (N_{\text{parallel workers}} = 8 \text{ on the GRID})$$

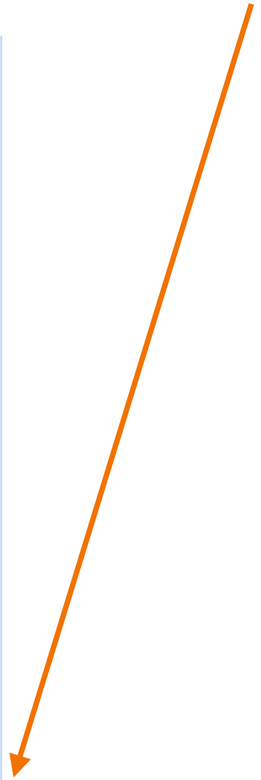
$$1\text{d} = 24 \cdot 60 \cdot 60\ \text{s} = 86400\ \text{s} \rightarrow 1\text{s} = 0.00001157407\ \text{d}$$

$$\text{Expected running time} = \frac{N_{\text{events}}^{\text{target}}}{N_{\text{events}}^{\text{test}}} \times \Delta t_{\text{test}} \times \frac{N_{\text{parallel workers}}}{10\ 000} \times 0.00001157407\ [\text{days} @ 10\text{kCPU}]$$

Estimate expected resources

2. Storage can be obtained by adding the total size of all files produced (bottom right number)

Permissions	Owner	Timestamp	Size	Filename
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	116.7 MB	AO2D.root
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	5.688 KB	async_pass_log.log
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	4.178 KB	config-json.json
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	5.593 MB	debug_log_archive.tgz
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	1.053 KB	dpl-config.json
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	113.1 KB	log_archive.zip
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	76.03 KB	pipeline_action_1157.log
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	108.5 KB	pipeline_action_38315.log
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	225.2 KB	pipeline_metric_1157.log
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	40.9 KB	pipeline_metric_38315.log
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	34.17 KB	QC_production.json
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	40.93 MB	qc_archive.zip
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	3.509 KB	stderr
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	246.9 KB	stdout
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	5.133 KB	timestampsampling_544510.log
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	4.178 KB	user_config.json
-rwxr-xr-x	aliprod:aliprod	03 Feb 2024 00:32	105.9 KB	workflow.json
Edit new file				**164.1 MB in 17 files**



$$\text{Expected storage} = \frac{N_{\text{events}}^{\text{target}}}{N_{\text{events}}^{\text{test}}} \times \text{size}_{\text{test}}$$

Script examples

Scripts with several workflow configurations from different PWGs can be found in https://github.com/AliceO2Group/O2DPG/tree/master/MC/run

Example: runPromptJpsi_midy_pp.sh

```
#!/usr/bin/env bash

# make sure O2DPG + O2 is loaded
[ ! "${O2DPG_ROOT}" ] && echo "Error: This needs O2DPG loaded" && exit 1
[ ! "${O2_ROOT}" ] && echo "Error: This needs O2 loaded" && exit 1


# ----------- LOAD UTILITY FUNCTIONS -------------------------
. ${O2_ROOT}/share/scripts/jobutils.sh

RNDSEED=${RNDSEED:-0}
NSIGEVENTS=${NSIGEVENTS:-1}
NBKGEVENTS=${NBKGEVENTS:-1}
NWORKERS=${NWORKERS:-8}
NTIMEFRAMES=${NTIMEFRAMES:-1}

${O2DPG_ROOT}/MC/bin/o2dpg_sim_workflow.py -eCM 900 -gen external -j ${NWORKERS} -ns ${NSIGEVENTS} -tf ${NTIMEFRAMES} -e TGeant4 -mod "--
skipModules ZDC" \
  -confKey "GeneratorExternal.fileName=${O2DPG_ROOT}/MC/config/PWGDQ/external/generator/
GeneratorParamPromptJpsiToElectronEvtGen_pp13TeV.C;GeneratorExternal.funcName=GeneratorParamPromptJpsiToElectronEvtGen_pp13TeV()"  \
         -genBkg pythia8 -procBkg inel -colBkg pp --embedding -nb ${NBKGEVENTS}

# run workflow
${O2DPG_ROOT}/MC/bin/o2_dpg_workflow_runner.py -f workflow.json
```

Anchored MC Productions

Anchored MC simulations: real detector conditions (dead channels, distortions, alignment, …)

Example of a script (run_anchored_mc.sh) to run anchored MC simulation (more info here):

```
export ALIEN_JDL_LPMANCHORPASSNAME=apass2
export ALIEN_JDL_MCANCHOR=apass2
export ALIEN_JDL_COLLISIONSYSTEM=pp
export ALIEN_JDL_CPULIMIT=8
export ALIEN_JDL_LPMPASSNAME=apass2
export ALIEN_JDL_LPMRUNNUMBER=544167
export ALIEN_JDL_LPMPRODUCTIONTYPE=MC
export ALIEN_JDL_LPMINTERACTIONTYPE=pp
export ALIEN_JDL_LPMPRODUCTIONTAG=LHC24a1
export ALIEN_JDL_LPMANCHORRUN=544167
export ALIEN_JDL_LPMANCHORPRODUCTION=LHC22o
export ALIEN_JDL_LPMANCHORYEAR=2023
export ALIEN_JDL_SIM_OPTIONS="-gen external -proc cdiff
config/PWGLF/ini/GeneratorLFStrangenessTriggered.ini"

export NTIMEFRAMES=2
export NSIGEVENTS=2
export SPLITID=100
export PRODSPLIT=153
export CYCLE=0
export ALIEN_PROC_ID=2963436952

${O2DPG_ROOT}/MC/run/ANCHOR/anchorMC.sh
```

How to run on different run numbers?

→ Not foreseen at the moment

→ workaround: launch anchorMC.sh several times with different run numbers (maybe write another script for that)

Anchored MC Productions

Anchored MC simulations: real detector conditions (dead channels, distortions, alignment, …)

Example of a script (run_anchored_mc.sh) to run anchored MC simulation (more info here):

```
export ALIEN_JDL_LPMANCHORPASSNAME=apass2
export ALIEN_JDL_MCANCHOR=apass2
export ALIEN_JDL_COLLISIONSYSTEM=pp
export ALIEN_JDL_CPULIMIT=8
export ALIEN_JDL_LPMPASSNAME=apass2
export ALIEN_JDL_LPMRUNNUMBER=544167
export ALIEN_JDL_LPMPRODUCTIONTYPE=MC
export ALIEN_JDL_LPMINTERACTIONTYPE=pp
export ALIEN_JDL_LPMPRODUCTIONTAG=LHC24a1
export ALIEN_JDL_LPMANCHORRUN=544167
export ALIEN_JDL_LPMANCHORPRODUCTION=LHC22o
export ALIEN_JDL_LPMANCHORYEAR=2023
export ALIEN_JDL_SIM_OPTIONS="-gen external -proc cdiff -ini ${O2DPG_ROOT}/MC/
config/PWGLF/ini/GeneratorLFStrangenessTriggered.ini"

export NTIMEFRAMES=2
export NSIGEVENTS=2
export SPLITID=100
export PRODSPLIT=153
export CYCLE=0
export ALIEN_PROC_ID=2963436952

${O2DPG_ROOT}/MC/run/ANCHOR/anchorMC.sh
```

How to run on different run numbers?

→ Launch anchorMC.sh several times with different run numbers (maybe write another script for that)

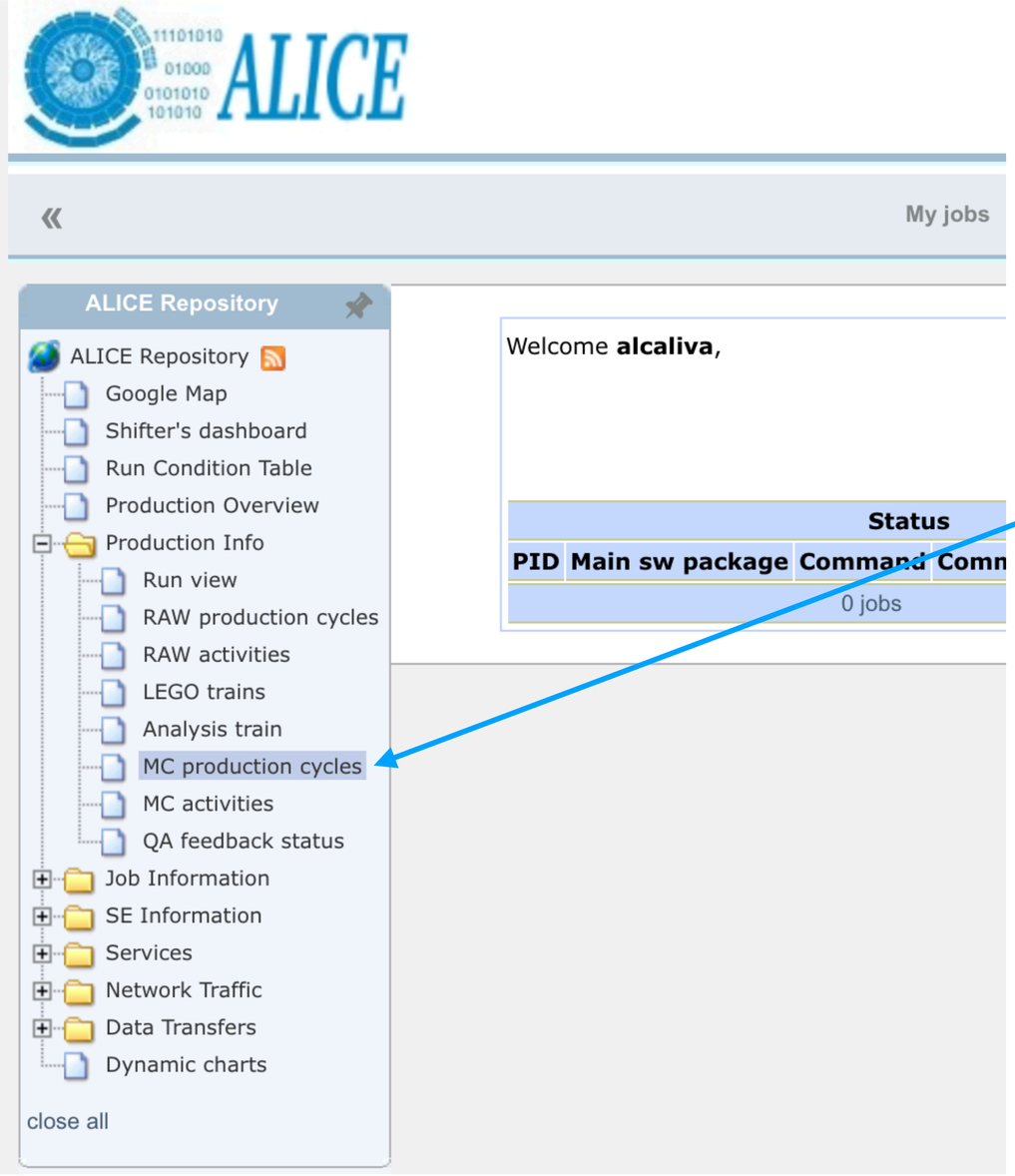
Anchored MC Productions on the GRID

To run an anchored MC production on the GRID:

```
[O2sim/latest] ~ %> ${O2DPG_ROOT}/GRID/utils/grid_submit.sh --script
run_anchored_mc.sh --jobname name --outputspec "*.log@disk=1","*.root@disk=2" --
packagespec "VO_ALICE@O2sim::v20240221-1" --wait --fetch-output
```

Shell script that you have produced (see previous slide)

List of existing MC productions



To view the complete list of MC productions and their status (running, completed, software update, …):

1. Go to Monalisa
2. Click on "MC production cycles"

List of existing MC productions

PRODUCTION CYCLES

Job Details » No filter

Ma

				Runs				Remaining time					»	
Production	**Description**		**Status**	**Range**	**Count**	**Event Count**	**Requested**	**@10000 cores**	**Comment**	**Known issues**	**Running time**	**Saving time**	**Output size**	**Software packages**
LHC23k6_TRDFix	Pb-Pb, 5.36 TeV - General Purpose anchored to Pb-Pb data periods in LHC23, anchored to apass2 (TRD fix), O2-4613		Running	544013-544510	30	2,134,800			RAW OCDB		95y 65d	294d 8:01	6.275 TB	O2PDPSuite::async-async-20240115.1.trd-slc7-alidist-O2PDPSuite-daily-20231208-0100-1, jq::v1.6-3
LHC23k4c	pp, 13.6 TeV - General Purpose anchored to apass1 of 2023 13.6 TeV pp data, production tag, no MFT ideal geo, O2-4278		Running	535069-535069	1	0			RAW OCDB		-	-	0 B	O2PDPSuite::async-async-20231103.1g-slc7-alidist-O2PDPSuite-daily-20231006-0200-1, jq::v1.6-3
LHC24a3_mumu_high	Pb-Pb, 5.36 TeV - Simulation of forward dimuons anchored to Pb-Pb data periods in LHC23 - mumu_high, O2-4591		Completed	544013-544510	30	986,170	1,000,000		RAW OCDB		31y 93d	128d 7:01	443.1 GB	O2PDPSuite::async-async-20240115.1-slc7-alidist-O2PDPSuite-daily-20231208-0100-1, jq::v1.6-3
LHC24a3_mumu_mid	Pb-Pb, 5.36 TeV - Simulation of forward dimuons anchored to Pb-Pb data periods in LHC23 - mumu_mid, O2-4591		Completed	544013-544510	30	985,090	1,000,000		RAW OCDB		31y 214d	126d 23:44	442.4 GB	O2PDPSuite::async-async-20240115.1-slc7-alidist-O2PDPSuite-daily-20231208-0100-1, jq::v1.6-3
LHC24a3_mumu_low	Pb-Pb, 5.36 TeV - Simulation of forward dimuons anchored to Pb-Pb data periods in LHC23 - mumu_low, O2-4591		Completed	544013-544510	30	1,478,900	1,500,000		RAW OCDB		47y 302d	183d 18:39	664.5 GB	O2PDPSuite::async-async-20240115.1-slc7-alidist-O2PDPSuite-daily-20231208-0100-1, jq::v1.6-3

Requesting a MC production: repetita juvant

1. Run a test on the GRID using your preferred generator and settings
2. Provide estimates for the running time, expected storage and number of events
3. Provide the link to the GRID folder with configuration/JDL used for tests and results

Requesting a MC production: repetita juvant

1. Run a test on the GRID using your preferred generator and settings
2. Provide estimates for the running time, expected storage and number of events
3. Provide the link to the GRID folder with configuration/JDL used for tests and results

Summary

Have fun running MC simulations

In case of problems or if you need assistance do not hesitate to contact us:

- Mattermost channels (preferred over private email): O2-simulation + O2DPG
- JIRA tickets for feature requests/bug reports (components simulation or O2DPG)

Summary

Have fun running MC simulations

In case of problems or if you need assistance do not hesitate to contact us:

- Mattermost channels (preferred over private email): O2-simulation + O2DPG
- JIRA tickets for feature requests/bug reports (components simulation or O2DPG)

Thank you for your attention!

Backup Slides

Grid certificate

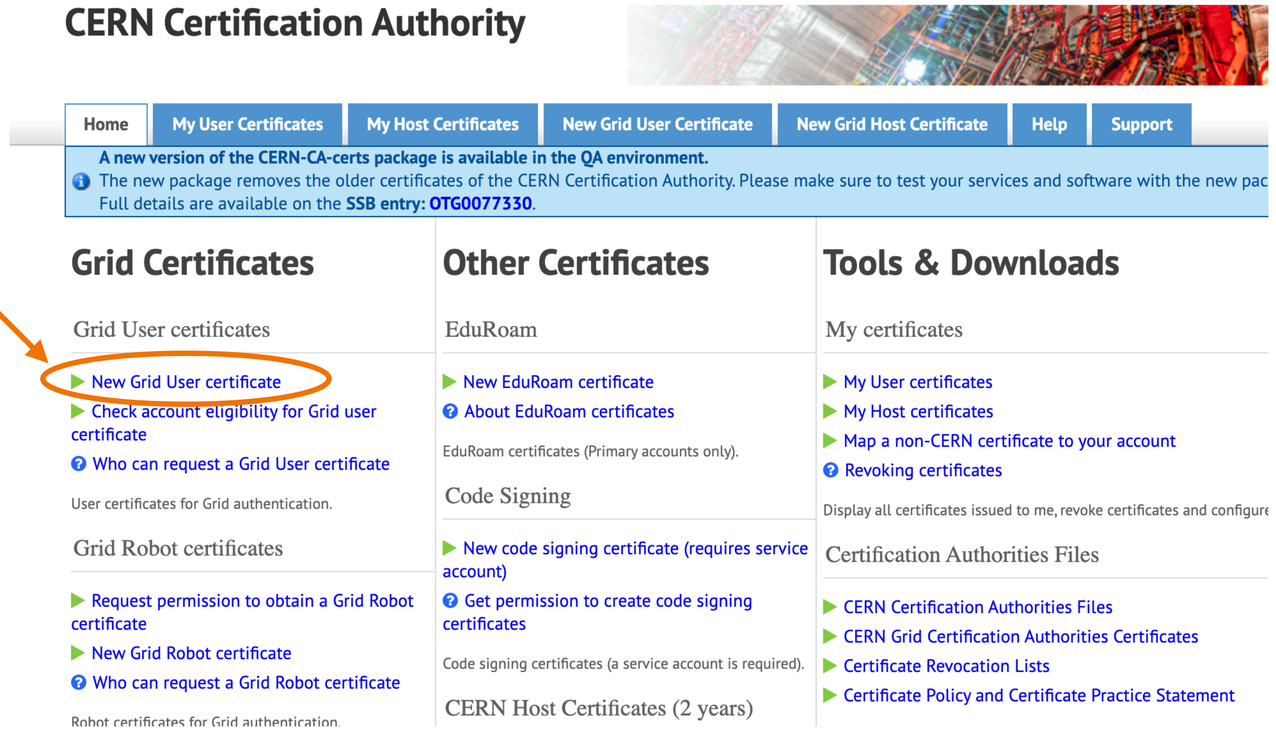
To run MC simulations with O2DPG you need a valid GRID certificate

→ needed to access information on the **CCDB** file

The *Condition* and *Calibration Data Base* (CCDB) is a ROOT file used to store calibration and alignment data

To get you GRID certificate

1. go to https://ca.cern.ch/ca/
2. Click on "New GRID User Certificate"



Grid certificate

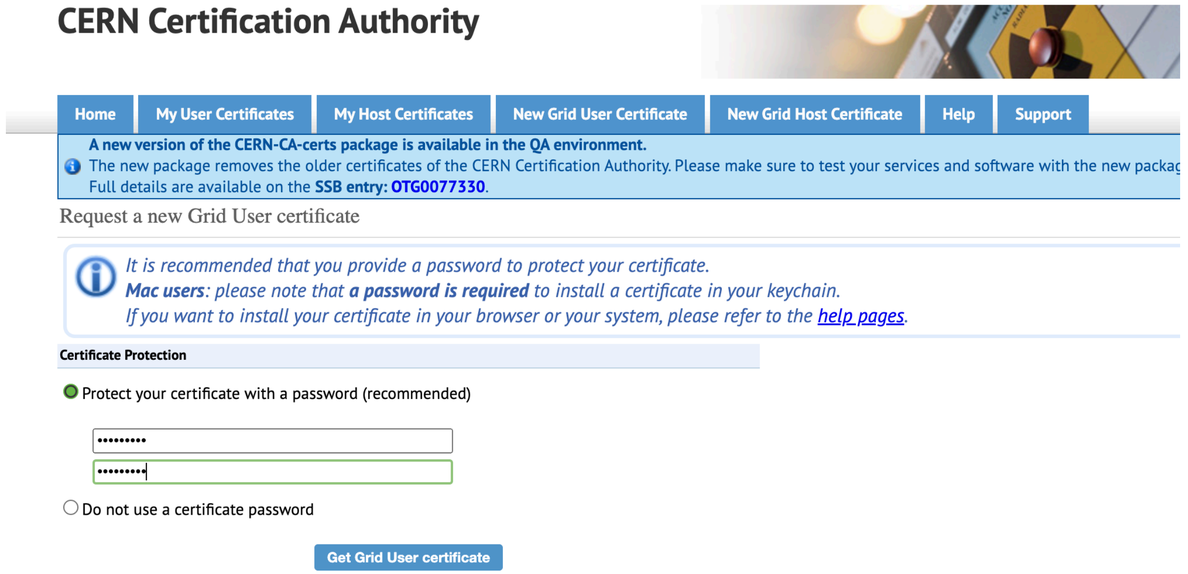
To run MC simulations with O2DPG you need a valid GRID certificate

→ needed to access information on the **CCDB** file

The Condition and Calibration Data Base (CCDB) is a ROOT file used to store calibration and alignment data

To get you GRID certificate

1. go to https://ca.cern.ch/ca/
2. Click on "New GRID User Certificate"
3. Invent a password to protect your certificate



Grid certificate

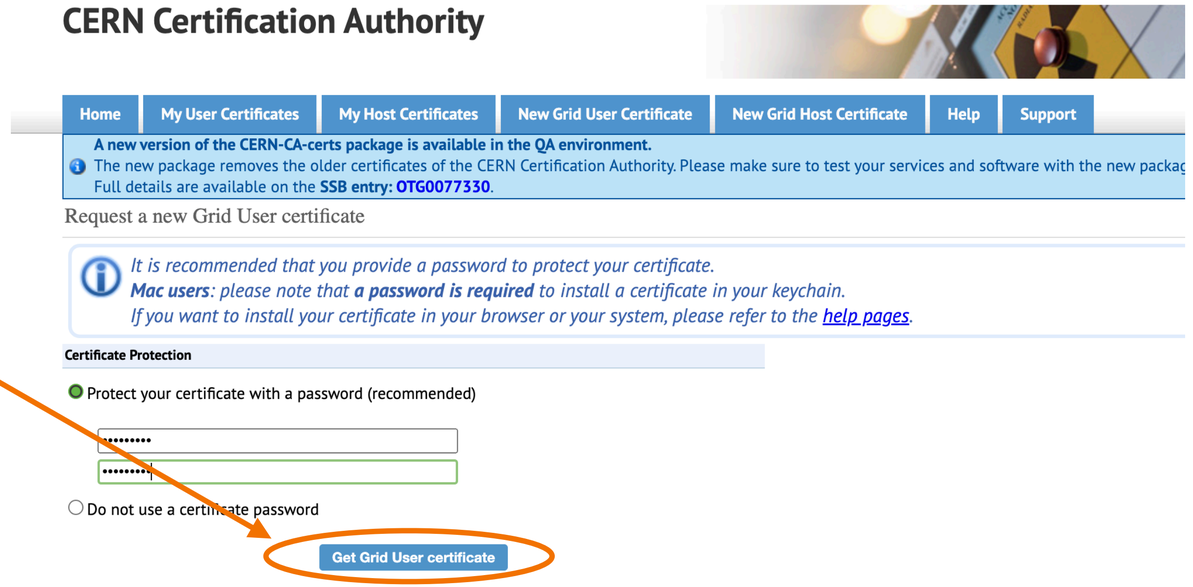
To run MC simulations with O2DPG you need a valid GRID certificate

→ needed to access information on the **CCDB** file

The Condition and Calibration Data Base (CCDB) is a ROOT file used to store calibration and alignment data

To get you GRID certificate

1. go to https://ca.cern.ch/ca/
2. Click on "New GRID User Certificate"
3. Invent a password to protect your certificate
4. Click on "Get Grid User certificate"



Grid certificate

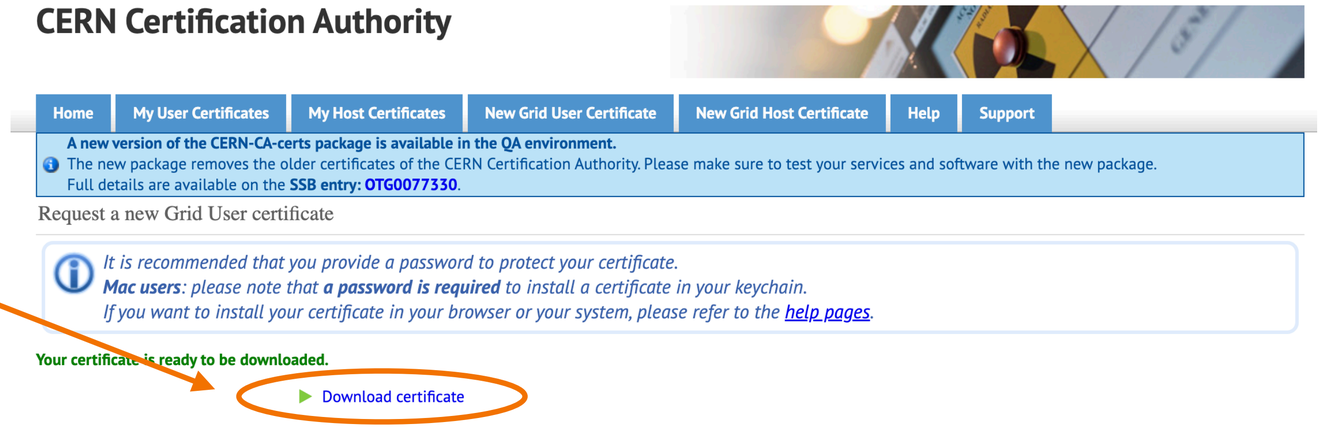
To run MC simulations with O2DPG you need a valid GRID certificate

→ needed to access information on the **CCDB** file

The Condition and Calibration Data Base (CCDB) is a ROOT file used to store calibration and alignment data

To get you GRID certificate

1. go to https://ca.cern.ch/ca/
2. Click on "New GRID User Certificate"
3. Invent a password to protect your certificate
4. Click on "Get Grid User certificate"
5. Once the certificate is ready download it



Grid certificate

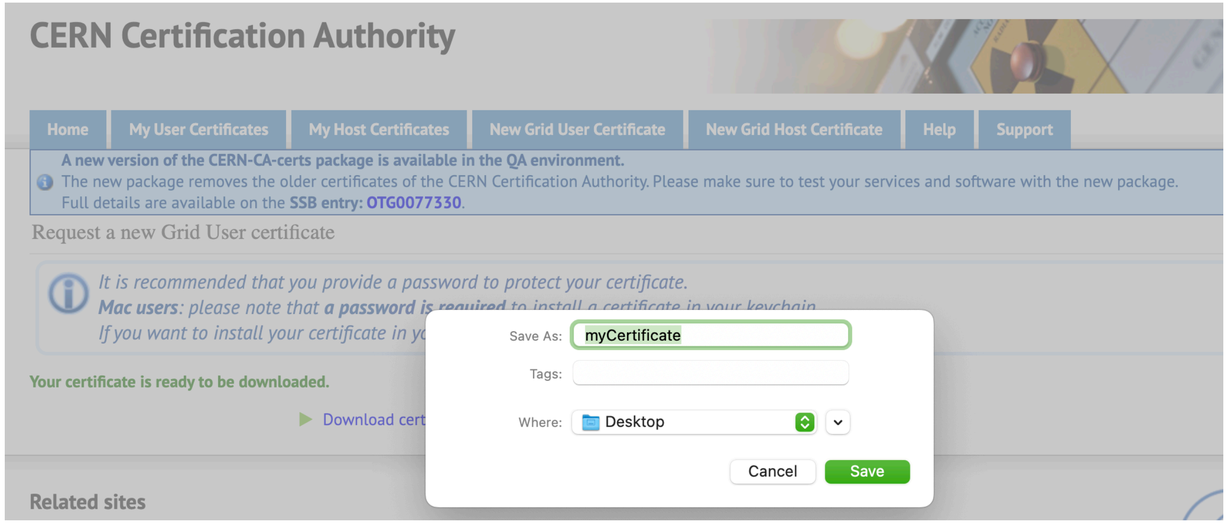
To run MC simulations with O2DPG you need a valid GRID certificate

→ needed to access information on the **CCDB** file

The Condition and Calibration Data Base (CCDB) is a ROOT file used to store calibration and alignment data

To get you GRID certificate

1. go to https://ca.cern.ch/ca/
2. Click on "New GRID User Certificate"
3. Invent a password to protect your certificate
4. Click on "Get Grid User certificate"
5. Once the certificate is ready download it
6. You get a file called "myCertificate.p12" that you can store where you prefer (also in your Desktop)



Grid certificate

Create the .globus directory (where the system expects to find your certificate) and copy myCertificate into it

```
$ mkdir ~/.globus
$ cp ~/Desktop/myCertificate.p12 ~/.globus
```

Go to the .globus directory and generate private and public keys from your certificate:

```
$ cd ~/.globus
$ openssl pkcs12 -in myCertificate.p12 -clcerts -nokeys -out usercert.pem
$ openssl pkcs12 -in myCertificate.p12 -nocerts -out userkey.pem
```

N.B. In both steps you are asked for the password used to protect your Certificate

Set the permissions on the files:

```
$ chmod 400 userkey.pem
$ chmod 400 usercert.pem
```

`chmod 400` = only the file owner has read permission

Grid certificate

You are ready to connect to the GRID!

Load the environment and then try to get a token (valid 24h)

```
$ alien-token-init
Enter PEM pass phrase:
DN >>> C=ch/O=AliEn2/CN=Users/CN=alcaliva/OU=alcaliva
ISSUER >>> C=ch/O=AliEn2/CN=AliEn CA
BEGIN >>> 2024-01-26 14:01:56
EXPIRE >>> 2024-02-26 16:01:56
```

Enter the certificate password
(N.B. the characters are invisible)

If you forget to load the environment you get an error like:

```
$ zsh: command not found: alien-token-init
```

Grid certificate on LXPLUS

If you are working on LXPLUS you need to copy your GRID certificate to your virtual space

Go to your .globus directory and copy your certificate, the userkey and usercert to LXPLUS:

```
$ cd ~/.globus
$ scp *.* alcaliva@lxplus.cern.ch:/afs/cern.ch/user/a/alcaliva/
```

Secure copy protocol (SCP) is a means of securely transferring computer files between a local host and a remote host

→ need to enter the password of your CERN account

Go to your virtual space on LXPLUS, create a .globus directory and move all the files inside it

```
$ ssh -X alcaliva@lxplus.cern.ch
$ (alcaliva@lxplus.cern.ch) Password:
$ mkdir .globus
$ mv myCertificate.p12 userkey.pem usercert.pem .globus
```

Now you are ready to get a token from LXPLUS!

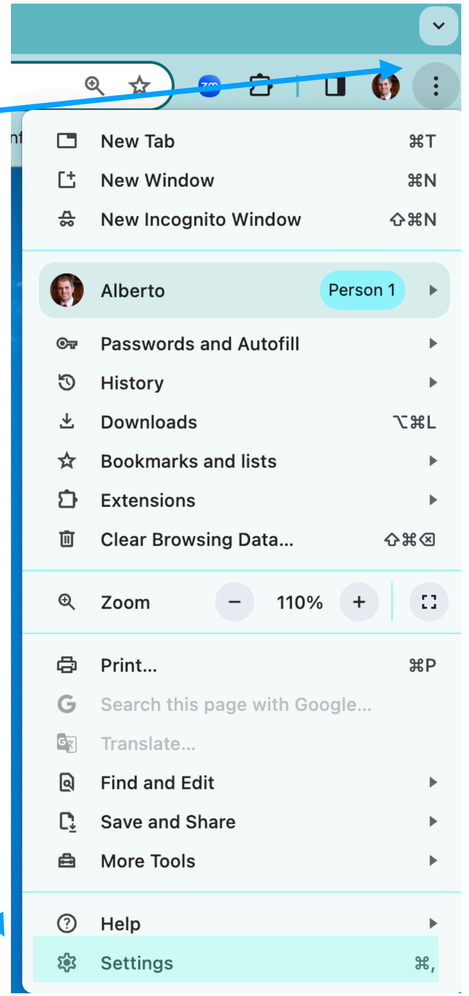
Importing your Grid certificate to the browser

To monitor your jobs running on the GRID go to alimonitor

To access it you need to import the GRID digital certificate to your browser

Instructions for Google Chrome:

1. Click on the ⋮ symbol (upper right corner of the web browser) and go to Settings



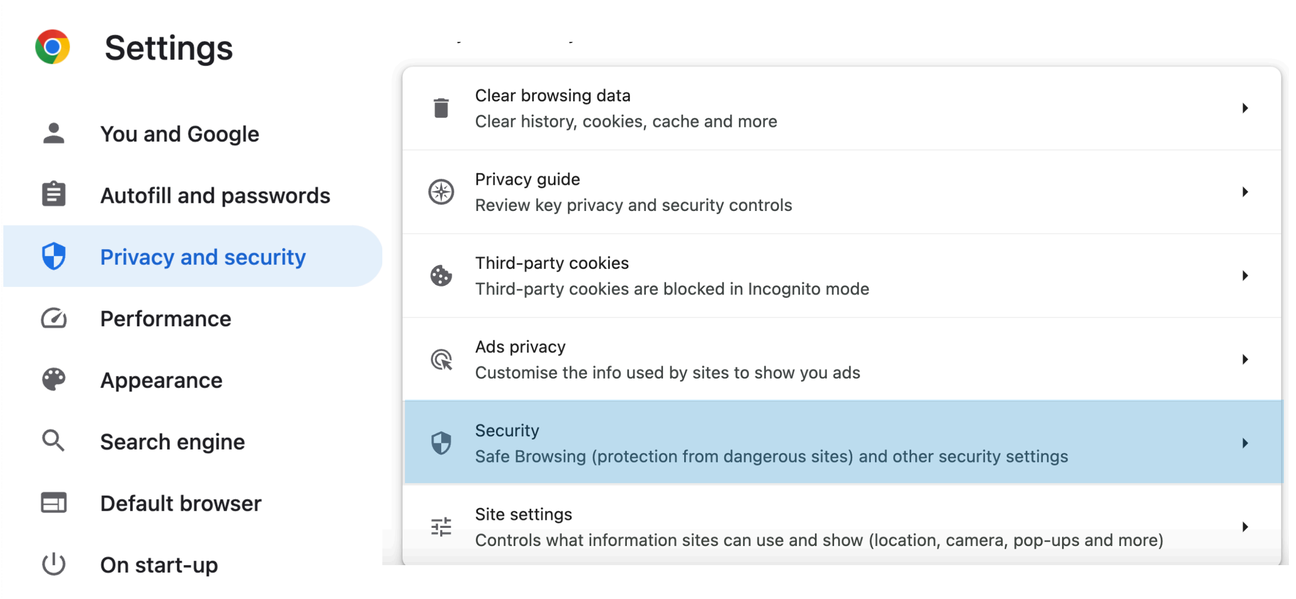
Importing your Grid certificate to the browser

To monitor your jobs running on the GRID go to [alimonitor]

To access it you need to import the GRID digital certificate to your browser

Instructions for Google Chrome:

1. Click on the ⋮ symbol (upper right corner of the web browser) and go to Settings
2. Click on "Privacy and security", then on "Security" and finally on "Manage certificates"



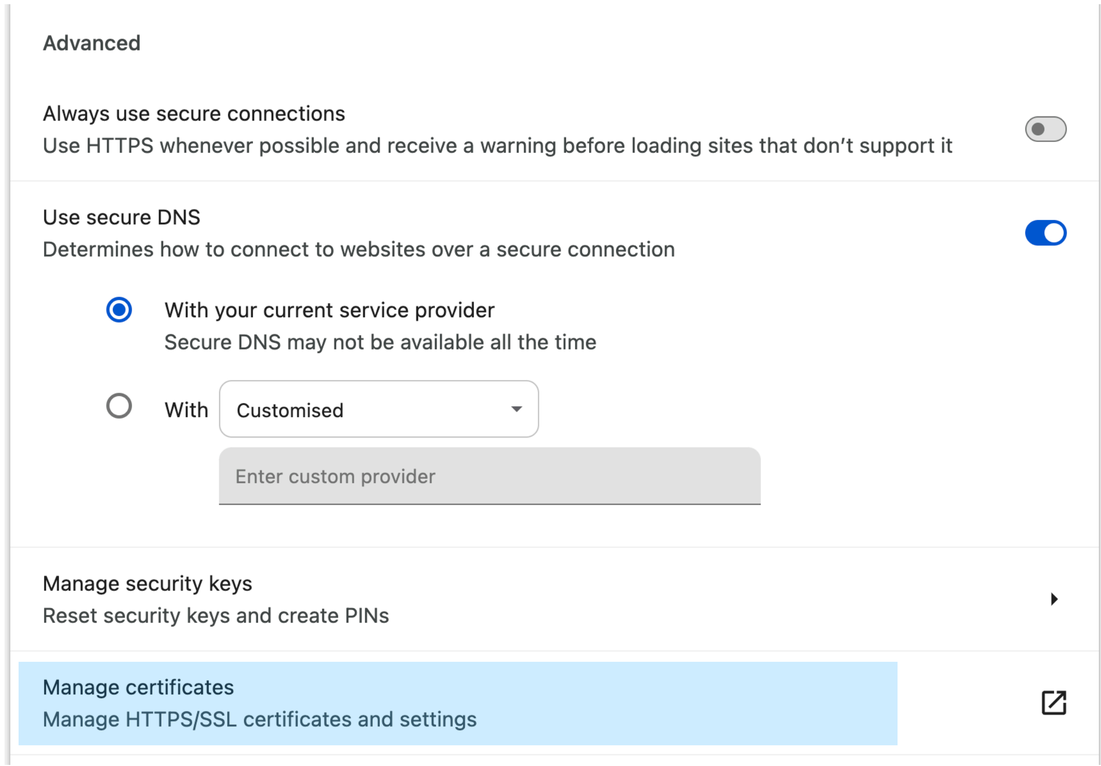
Importing your Grid certificate to the browser

To monitor your jobs running on the GRID go to alimonitor

To access it you need to import the GRID digital certificate to your browser

Instructions for Google Chrome:

1. Click on the ⋮ symbol (upper right corner of the web browser) and go to Settings
2. Click on "Privacy and security", then on "Security" and finally on "Manage certificates"



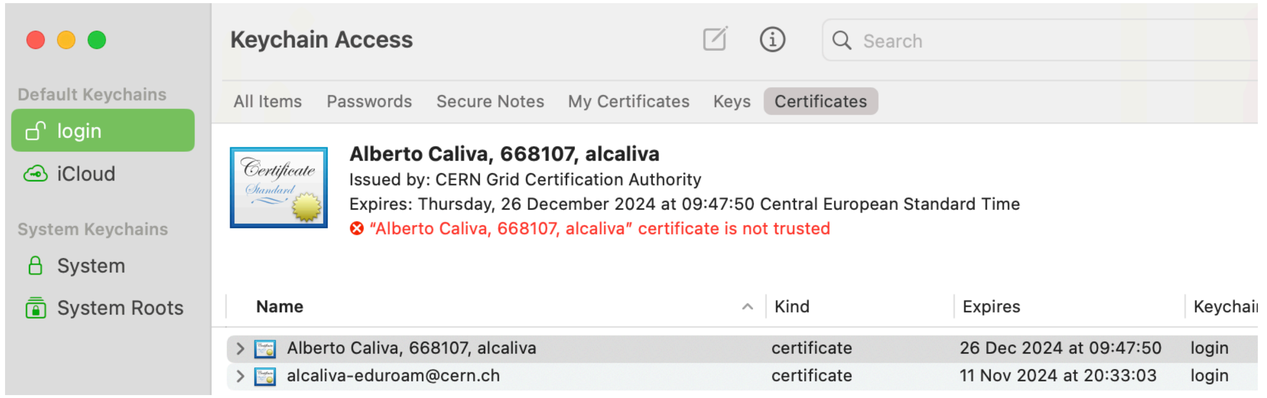
Importing your Grid certificate to the browser

To monitor your jobs running on the GRID go to alimonitor

To access it you need to import the GRID digital certificate to your browser

Instructions for Google Chrome:

1. Click on the ⋮ symbol (upper right corner of the web browser) and go to Settings
2. Click on "Privacy and security", then on "Security" and finally on "Manage certificates"
3. This will open the "keychain Access" board
4. Make sure "login" and "Certificates" are selected



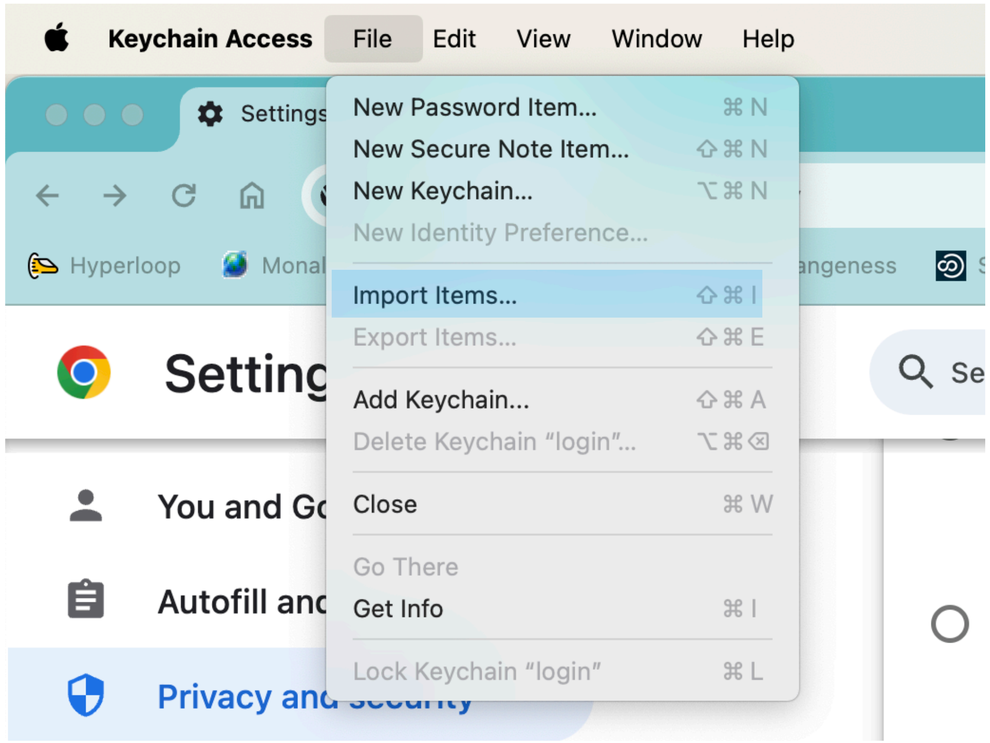
Importing your Grid certificate to the browser

To monitor your jobs running on the GRID go to [alimonitor]

To access it you need to import the GRID digital certificate to your browser

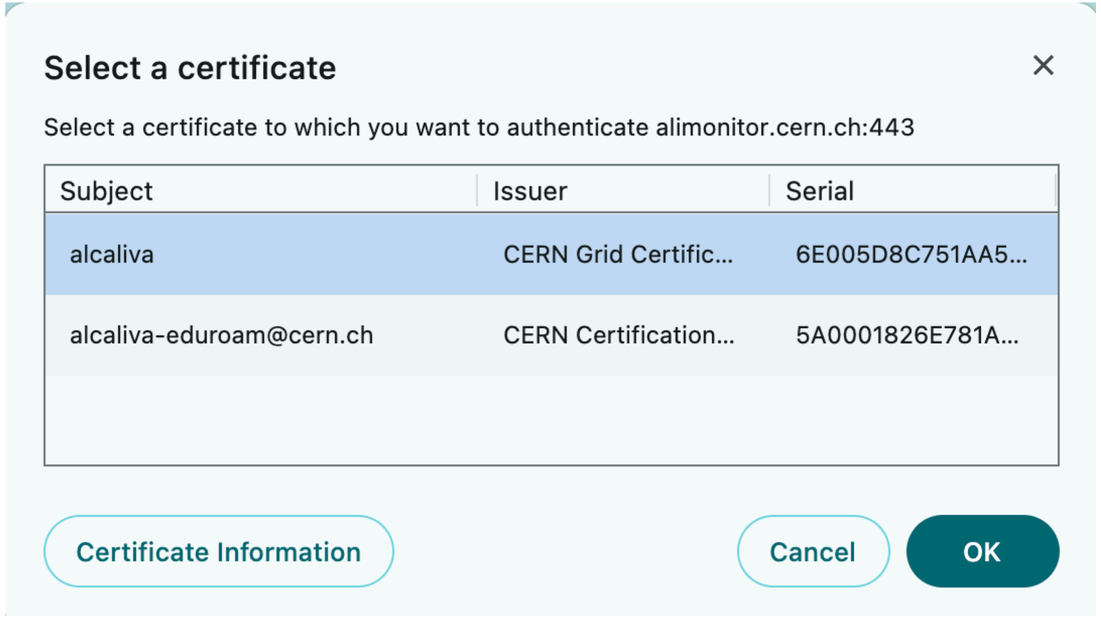
Instructions for Google Chrome:

1. Click on the ⋮ symbol (upper right corner of the web browser) and go to Settings
2. Click on "Privacy and security", then on "Security" and finally on "Manage certificates"
3. This will open the "keychain Access" board
4. Make sure "login" and "Certificates" are selected
5. Click on File and then on "import items"
6. Select the "myCertificate.p12" and click on OPEN
7. Quit Chrome and reopen it



Importing your Grid certificate to the browser

Go to alimonitor and select the GRID digital certificate



Now you are ready to monitor your jobs on the GRID !