

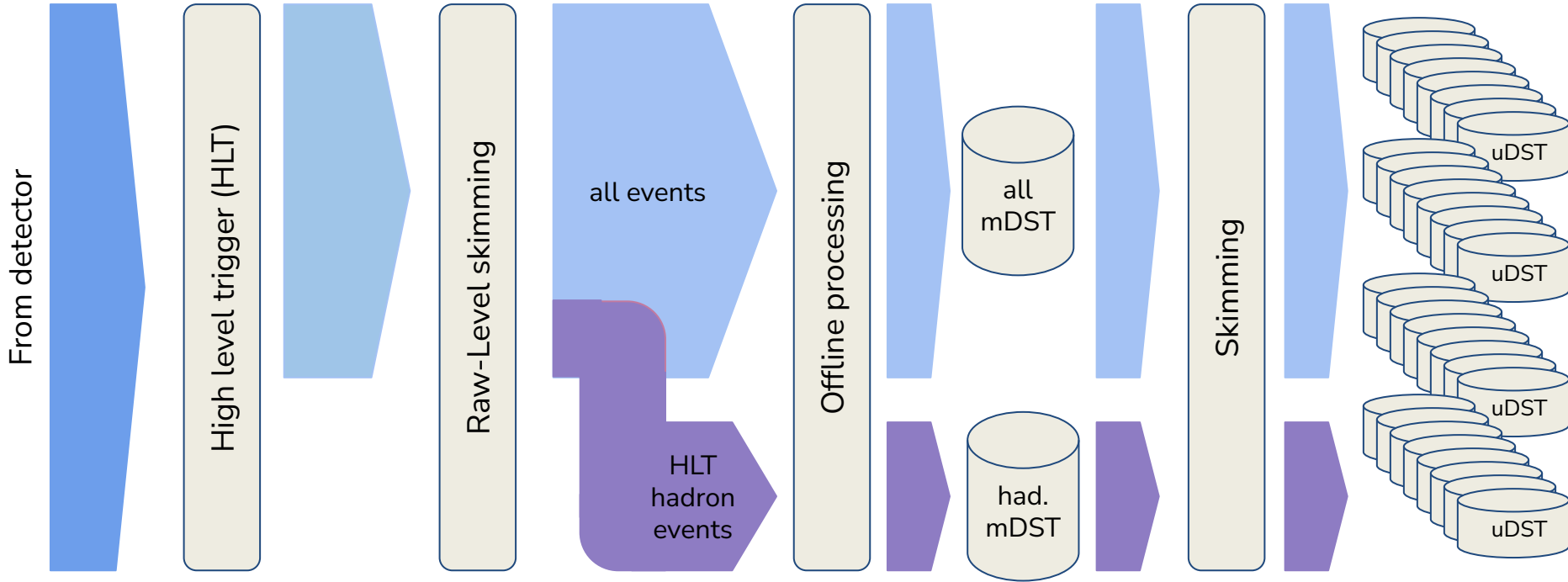
How to use our SW and DP tools

Physics week
KEK, October 31st 2023

*Giacomo De Pietro
Giulio Dujany
Stefano Lacaprara
Umberto Tamponi*

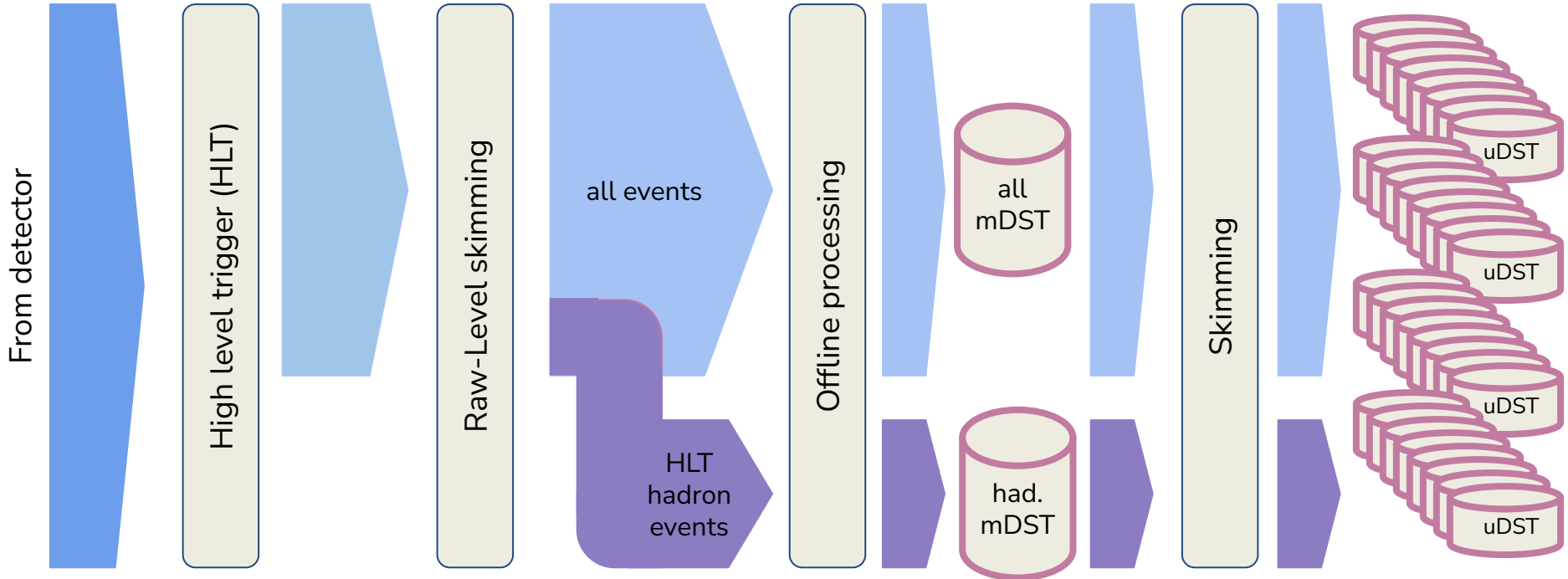
Introduction

From detector to analysis



From detector to analysis

mDSTs and uDSTs will always be available



Data overview

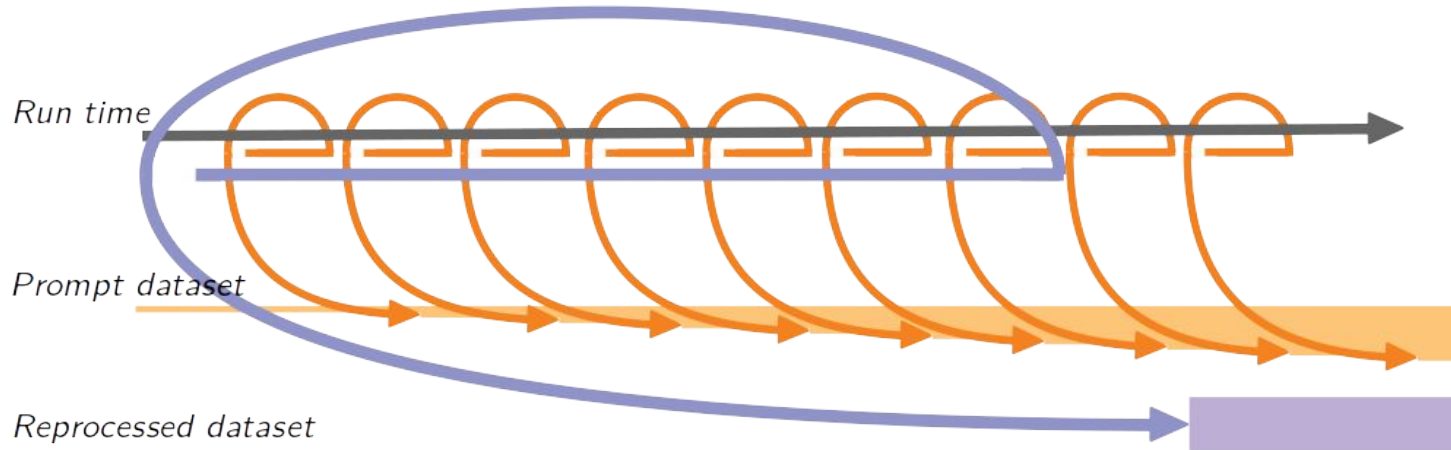
Prompt processing (prompt, buckets)

- New data, calibrated and processed in “real time”
- Available during data taking, with ~ 1 month of delay
- Use the latest major release

The latest proc and prompt are always provided with the same major release

Reprocessings (procXX)

- Old data re-calibrated and reprocessed with the latest release



MC overview

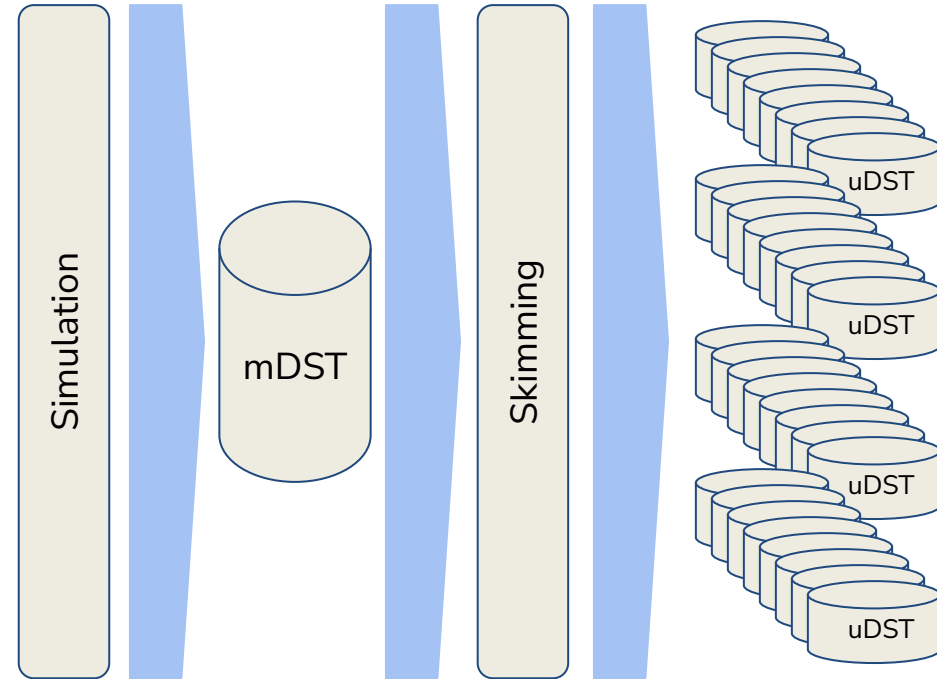
Each physical process ($q\bar{q}$, B^+B^- , $\mu\mu\mu$...) requires a different generator. We have to make a cocktail producing them separately

Generic MC = cocktail of all basic processes

- Produced by default

Signal MC = the specific process for your analysis

- Produced according to the WG needs
- Additional requests are always welcome



MC overview

Run-independent MC (MCri)

- Default channel masking and detector conditions
- Simulated backgrounds, same level for all events
- exp = 0 (full lumi), 1003 (early phase 3), 1004 (phase 3)
- Luminosity = 1 ab⁻¹

Run-dependent MC (MCrd)

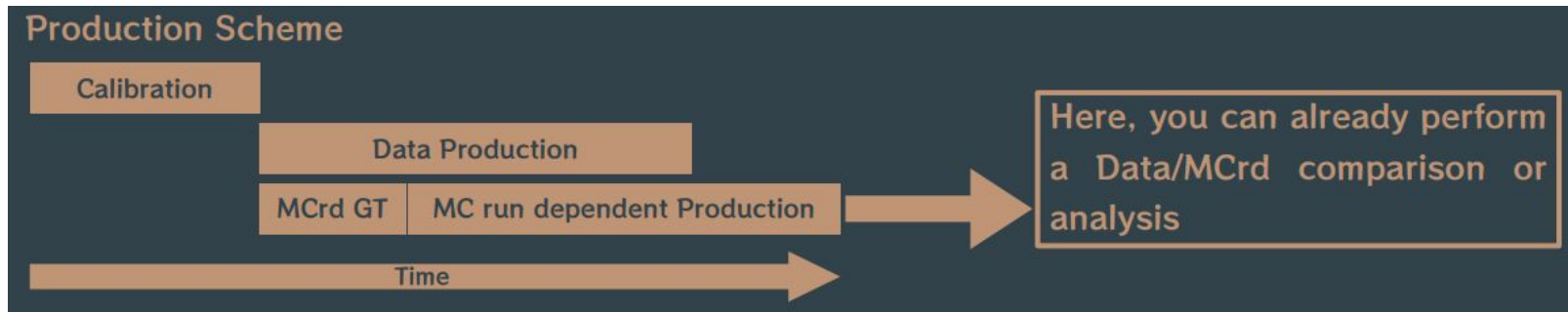
- Same masking and conditions as in data
- Realistic background from random triggers
- Events generated for all runs proportionally to the real data luminosity

Process	Streams
qqbar (uubar+ddbar+ssbar+ccbar)	4
mixed	4
charged	4
taupair	4
mumu	4
ee	0.1
gg	2
eemumu	1
eeee	1
llXX	1
hhISR	1
BB* + B*B*	(4)

Production flow

We aim to produce data and MC at the same pace

- You should see data and MC appearing more or less at the same time during data taking



How to run your analysis

General guidelines

Do development only locally

Run on as little events as possible

Re-run as little times as possible

Bookmark these pages:

<https://confluence.desy.de/display/BI/GBasf2+Troubleshooting>

<https://gbasf2.belle2.org/>

My analysis: testing

You most certainly WANT to test your code locally before submitting thousands jobs to the grid!

gbasf2 has scouting enabled, to protect against this kind of massive failures, but you don't want to submit O(10k) jobs just to find afterward that you forget a comma in your steering.

- 1) Get one file from grid via gb2_ds_get:

```
gb2_ds_search collection --list_datasets /belle/collection/Data/proc13_had_4S_v3
```

```
gb2_ds_search dataset --campaign MC15ri_b --data_type mc --mc_event 1110062100 --bkg_level BGx1
```

- 2) Use the data and MC available on the `dataprod/` disk at KEKCC:
`/group/belle2/dataprod/new/[MC,Data]/`

My analysis: How to find data

Go to the main data page:

<https://confluence.desy.de/display/BI/Data+Production+WebHome>

Data Production WebHome

Umberto Tamponi posted on 11. Mar. 2021 13:08h - last edited by Stefano Lacaprarà on 20. Oct. 2023 12:54h

Welcome to the **Data production** confluence page.

Here you will find all the **official information** about the available Data and MC samples.



	Data production status	
MC main page	Data main page	Skim main page
Luminosity main page	Validation page	Calibration page
DP repository	Special Processing page	

My analysis: How to find data

Go to the main data page:

<https://confluence.desy.de/display/BI/Data+Production+WebHome>

Data Production WebHome

Umberto Tamponi posted on 11. Mar. 2021 13:08h - last edited by Stefano Lacaprra on 20. Oct. 2023 12:54h

Welcome to the **Data production** confluence page.

Here you will find all the **official information** about the available Data and MC samples.

Your entry point



Detailed info

Data production status		
MC main page	Data main page	Skim main page
Luminosity main page	Validation page	Calibration page
DP repository	Special Processing page	

Skims

Skim: A rough selection of your input sample, which allow to run your refined selection much faster

Kind of skims:

- HLT skim: several skim for calibration purposes + *hlt_hadron*
- Physics skim: analysis-dedicated skim, which can run on unskimmed sample or on *hlt_hadron* skimmed sample

Skims are the most efficiency way to go

- You don't run on events that would never pass your selection
- Files are merged to minimize their number

Know you skim/DP liaison. They know how to navigate through DP

Go to <https://confluence.desy.de/display/BI/Skim+Production+Status> and check if there is a skim for you

Follow the instructions (they change over time)

Skims

Skims are:

- Prepared by any collaborator who needs them, with the help of the DP liaison of the WG
- Run by default on both data and MC.
- If you need to skim a signal MC, do it on the fly in analysis or ask your DP liaison

```
# Check the skim output
from skim.WGs.tdcpv import TDCPV_ccs # pick your favourite skim

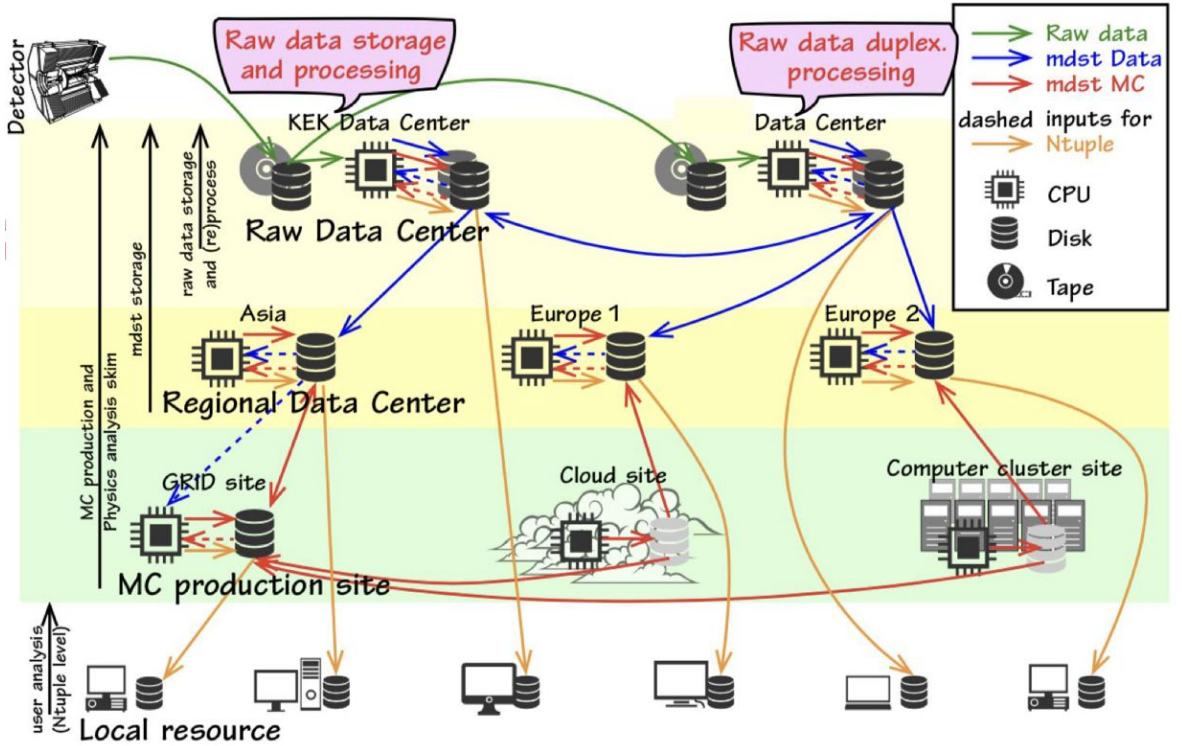
skim_ccs = TDCPV_qqs(udstOutput=False) # pass udstOutput=False to disable udstOutput
skim_ccs(path=my_path)
var.addAlias("skim_ccs", f"{skim_ccs.flag}")

# then save "skim_ccs" in your final tree
```

Analyzing data: the grid & gbasf2

Data are distributed over many different centers on a **computing grid**

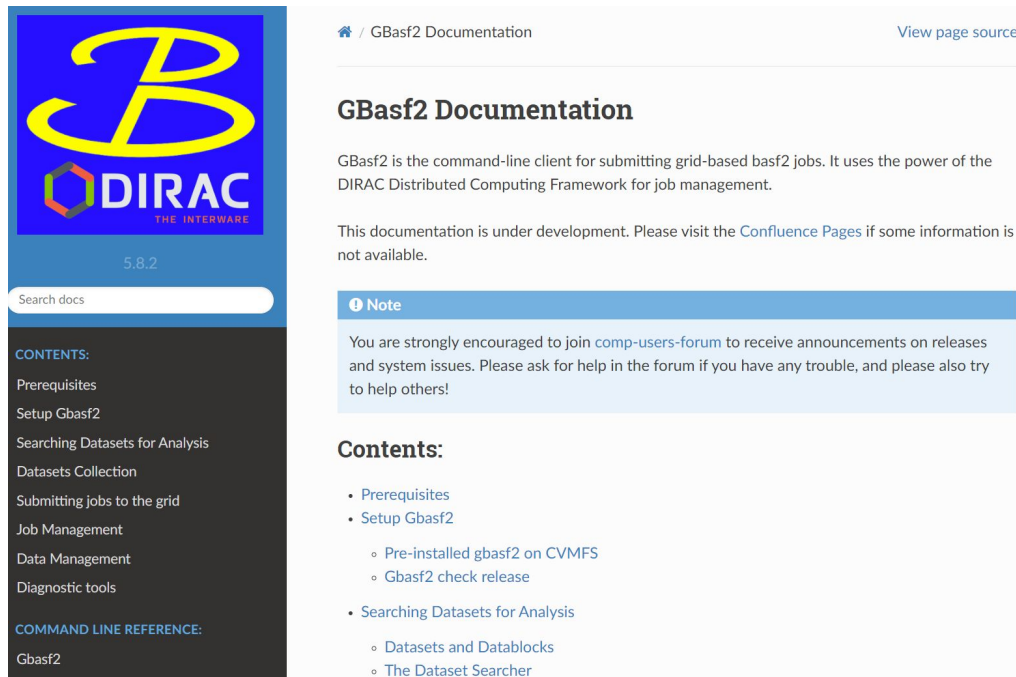
- Many networked loosely computers.
- Data centers keep two copies of the full raw data set.
- Raw data is staged, reprocessed, skimmed and distributed over sites.
- Analysers access data **sending jobs to the grid and downloading the output.**



Analyzing data: the grid & gbasf2

The computing group provides an extensive documentation of **gbasf2** and all the tools included:
gbasf2.belle2.org

Please check it regularly: gbasf2 developers are always including new features!



Home / GBasf2 Documentation [View page source](#)

GBasf2 Documentation

GBasf2 is the command-line client for submitting grid-based basf2 jobs. It uses the power of the DIRAC Distributed Computing Framework for job management.

This documentation is under development. Please visit the [Confluence Pages](#) if some information is not available.

Note

You are strongly encouraged to join [comp-users-forum](#) to receive announcements on releases and system issues. Please ask for help in the forum if you have any trouble, and please also try to help others!

Contents:

- [Prerequisites](#)
- [Setup Gbasf2](#)
 - [Pre-installed gbasf2 on CVMFS](#)
 - [Gbasf2 check release](#)
- [Searching Datasets for Analysis](#)
 - [Datasets and Datablocks](#)
 - [The Dataset Searcher](#)

What if I need help?

If you need help, you are invited to use questions.belle2.org, but please try to follow these few “rules” for increasing the chances to receive a quick answer:

Please follow these rules when asking questions:

1. Attempt to debug the problem / answer the question for yourself for a bit. Please also look at the [Software Documentation](#) and try searching the [Belle II Confluence](#)
2. State the problem/question. Cut-and-paste the **EXACT and FULL** error message or output, if it exists. Please also state the **version of the software** showing the problem (e.g. the output from running `basf2 --info`)
3. If relevant, post the **COMPLETE** script causing the error. For longer scripts please attach them as files. Otherwise please make sure you **format them as preformatted text/code** (select it and press the button with "101010" in the toolbar)
4. Optional but useful: State the overall context of the question/problem (i.e. what physics analysis or goal are you ultimately trying to accomplish).
5. Tag your question
6. Post the question, and continue to repeat Step 1 in the meantime.
7. Upvote correct/useful answers. When complete, mark the correct answer and close the entry.
8. Don't be afraid to downvote unhelpful or wrong answers
9. Share your knowledge/experience by answering other Questions.

What if I need help?

If you need help, you are invited to use questions.belle2.org, but please try to follow these few “rules” for increasing the chances to receive a quick answer:

Please follow these rules when asking questions:

1. Attempt to debug the problem / answer the question for yourself for a bit. Please also look at the [Software Documentation](#) and try searching the [Belle II Confluence](#)
2. State the problem/question. Cut-and-paste the **EXACT and FULL** error message or output, if it exists. Please also state the **version of the software** showing the problem (e.g. the output from running `basf2 --info`)
3. If relevant, post the **COMPLETE** script causing the error. For longer scripts please attach them as files. Otherwise please make sure you **format them as preformatted text/code** (select it and press the button with "101010" in the toolbar)
4. Optional but useful: State the overall context of the question/problem (i.e. what physics analysis or goal are you ultimately trying to accomplish).
5. Tag your question
6. Post the question, and continue to repeat Step 1 in the meantime.
7. Upvote correct/useful answers. When complete, mark the correct answer and close the entry.
8. Don't be afraid to downvote unhelpful or wrong answers
9. Share your knowledge/experience by answering other Questions.

A common error is cutting-and-pasting only part of the error message (if any):

if you are not sure what's the relevant part of the log to be included, attach the full log as a file!

Tips & tricks

My analysis: look at your logs

After having tested locally your steering file, please, PLEASE, check your log file:
is your job producing one or more [INFO] or [WARNING] messages per event?

If yes:

- if your job produces a tons of [WARNING] messages probably you are doing something wrong
 - **check carefully the messages and which module is printing them!**
- a job producing too many messages will likely crash on the grid
 - **there is an hard limit on the log size produced by a grid job!**

My analysis: look at your logs

After having tested locally your steering file, please, PLEASE, check your log file:
is your job producing one or more [INFO] or [WARNING] messages per event?

If yes:

- if your job produces a tons of [WARNING] messages probably you are doing something wrong
 - **check carefully the messages and which module is printing them!**
- a job producing too many messages will likely crash on the grid
 - **there is an hard limit on the log size produced by a grid job!**

If you verify that you are not doing anything wrong (sometimes WARNING messages thrown by basf2 are false-positive), then you have to suppress these messages:

```
import basf2
basf2.set_log_level(basf2.LogLevel.ERROR) # this suppresses both INFO and WARNING messages
```

My analysis: look at your logs

After having tested locally your steering file, please, PLEASE, check your log file:
is your job producing one or more [INFO] or [WARNING] messages per event?

If yes:

- if your job produces a tons of [WARNING] messages probably you are doing something wrong
 - **check carefully the messages and which module is printing them!**
- a job producing too many messages will likely crash on the grid
 - **there is an hard limit on the log size produced by a grid job!**

If you verify that you are not doing anything wrong (sometimes WARNING messages thrown by basf2 are false-positive), then you have to suppress these messages:

```
import basf2
basf2.set_log_level(basf2.LogLevel.ERROR) # this suppresses both INFO and WARNING messages
```

If you are flooded by ERROR messages:

- **it is very likely that you are doing something very wrong**
- and you can not suppress them!

My analysis: counting your events

When you submit your project you get a print of the number of events

- If you forgot, just run a dry project with `--dry`

```
Running gbasf2 -n 1 steering_etaSelection.py -p UThhbeta_mixed_Md15_test -i
/belle/collection/MC/MC15rd_mixed_exp21_5S_scan_v1 -s light-2305-korat --force --dry
```

```
*****
***** Project summary *****
** Project name: UThhbeta_mixed_Md15_test
** Dataset path: /belle/user/tamponi/UThhbeta_mixed_Md15_test
** Steering file: steering_etaSelection.py
** Job owner: tamponi @ belle (23:53:17)
** Preferred site / SE: None / DESY-TMP-SE
** Input files for first job:
LFN:/belle/MC/release-06-00-12/DB00002335/MC15rd_a/prod00028853/s00/e0021/5S_scan/r00083/m
ixed/mdst/sub00/mdst_000001_prod00028853_task2082000001.root
** Number of input files: 275
** Number of jobs: 275
** Processed data (MB): 166380
** Processed events: 15304731 events
** Estimated CPU time per job: 928 min
*****
```


My analysis: counting your events

When you submit your project you get a print of the number of events

- If you forgot, just run a dry project with `--dry`

Include an event-counting histogram in your steering file

```
import modularAnalysis as ma

ma.variablesToHistogram(decayString = '',
                        variables_2d=[('expNum', 30, 0, 30), ('runNum', 1500, 0, 15000)],
                        filename='ntuple.root',
                        path=my_path)
```

gb2_ds_count_events

My analysis: CPU time

When you submit your job on the grid, you need to pass the expected execution time per event (`evtpersec`) of your job:

- by default, `gbasf2` set a large time (to be on the safe side)
- passing a “realistic” execution time allows the grid to correctly schedule your job and “prioritize” them

How to estimate the execution time per event?

- download at KEKCC one file of your signal sample
- execute your steering file at KEKCC submitting a job on the KEKCC batch system with `bsub`
- take the CPU time from the `bsub` log

Resource usage summary:

```
CPU time :    20.07 sec.  <-- this is the number you need
```

- compute the CPU time per event via `evtpersec = nevents / (20 * <CPU time from the bsub log>)`
- submit your job via `gbasf2` using `--evtpersec <evtpersec>`

My analysis: many analyses in one job

Save multiple trees in one output file:

```
import modularAnalysis as ma

ma.reconstructDecay('B0:JpsiKS -> J/psi K_S0', cut='', path=my_path)
ma.variablesToNtuple(
    decayString='B0:JpsiKS',
    variables=['Mbc'],
    treename='JpsiKS',
    filename='ntuple.root', # same filename here
    path=my_path)
ma.reconstructDecay('B+:JpsiKp -> J/psi K+', cut='', path=my_path)
ma.variablesToNtuple(
    decayString='B+:JpsiKp',
    variables=['Mbc'],
    treename='JpsiKp',
    filename='ntuple.root', # and here!
    path=my_path)
```

More trees you add to your output files, **less projects (= jobs) you need to submit via gbasf2!**

Using your MVA weightfile on grid

You may be tempted to ship your weightfile via gbasf2 input sandbox, but:
if the weightfile is large ($O(\text{Mb})$ or more), it may create issues on the whole grid system!

Using your MVA weightfile on grid

You may be tempted to ship your weightfile via gbasf2 input sandbox, but:
if the weightfile is large (O(Mb) or more), it may create issues on the whole grid system!

How to correctly use it on grid?

long story short: look at my notes [here](#)

long story long: you need to learn:

- how to use a generic MVA weightfile with a basf2 job
- how to upload it on the Conditions Database

Using your MVA weightfile on grid

How to use a generic MVA weightfile with a basf2 job

You can store as payload (almost) any file. This is an example for dumping a “raw” file into a payload (in this case, a JSON file “a_dictionary.json”):

```
import basf2

# Here we do our black magic
# first: we import ROOT (ah, such lovely library)
import ROOT # noqa

# second: we invoke the Database class
db = ROOT.Belle2.Database.Instance()

# third: we define an interval-of-validity
# note the arguments of IntervalOfValidity() are
# experiment-low, run-low, experiment-high, run-high numbers
# IntervalOfValidity(int experimentLow, int runLow, int experimentHigh, int runHigh)
ioy = ROOT.Belle2.IntervalOfValidity(0, 0, -1, -1)

# fourth: we store the file as payload into a local database
# note that the arguments of 'Database::addPayload()' are
# payload name, path to the file to store, interval-of-validity
db.addPayload('ADictionaryJSON', basf2.find_file('a_dictionary.json'), ioy)
```

Using your MVA weightfile on grid

How to use a generic MVA weightfile with a basf2 job

After having executed the previous snippet, a folder `localdb` is created locally. Such folder contains:

- a file called `database.txt` (which is the so-called local database);
- a file called `dbstore_ADictionaryJSON_rev_bea927.root` (which is the so-called payload).

Running `cat localdb/database.txt` returns `dbstore/ADictionaryJSON bea927 0,0,-1,-1`:

- `ADictionaryJSON` is the payload name;
- `bea927` is the payload checksum;
- `0,0,-1,-1` is the payload interval-of-validity.
 - the values here are: `int experimentLow, int runLow, int experimentHigh, int runHigh`

Using your MVA weightfile on grid

How to use a generic MVA weightfile with a basf2 job

It is rather straightforward to use the payload in a custom basf2 module (e.g.: a basf2 module written in Python):

```
import basf2

class ReadJSONFromCDB(basf2.Module):
    # Loading the database object in beginRun() ensures that, in case of run-dependent or
    # experiment-dependent payloads, the framework correctly loads them.

    def beginRun(self):
        import ROOT # noqa
        # The payload name must be 'ADictionaryJSON' as before
        self.a_dictionary_dbobject = ROOT.Belle2.DBAccessorBase(
            ROOT.Belle2.DBStoreEntry.c_RawFile, 'ADictionaryJSON', True
        )
        # This is the actual payload (namely: the path to the payload)
        self.a_dictionary_payload = self.a_dictionary_dbobject.getFilename()

    def event(self):
        # Here you simply use the object self.a_dictionary_payload as you wish
        # with open(self.a_dictionary_payload) as a_dictionary_json:
        ...
```


Using your MVA weightfile on grid

How to upload the MVA weightfile on the Conditions Database

You need to:

- `basf2_mva_upload/basf2_mva_add_to_local_database` if you have an XML weightfile created with our MVA package (e.g. from FastBDT)
- `b2conditionsdb-tag-create` for creating a globaltag: the globaltag must start with `user_USERNAME_` or `temp_USERNAME_`, where `USERNAME` is your b2mms username
- `b2conditionsdb-upload` for uploading your payloads to the globaltag created in the previous step
- `b2conditionsdb-tag-state` for changing the globaltag state to `TESTING` (basf2 crashes if you try to use an `OPEN` globaltag)