# Table of Contents

# LHCb Stripping FAQ

## Which DaVinci version & tags to use?

For the vast majority of user analysis (making ntuples from stripping output, etc), you should use the latest version of DaVinci and the latest CondDB and DDDB tags available for the data type that you're analysing. This way you get the the most recent versions of analysis tools with any improvements or bug fixes.

Note that, as analysis tools evolve, you may not get exactly the same values for observables with different DaVinci versions. Eg, the PID tuning or the vertex fitting algorithm may change. This just means that with the latest DaVinci you get the best possible measurement of the observables.

If you need to reproduce exactly the values of the observables that were used during the relevant stripping campaign you must use the same DaVinci version as was used for that campaign. This is the case if you're restripping the data for whatever reason, or if you're stripping MC. See here for details of which DaVinci version was used for each stripping campaign.

## How do I run some stripping line on MC data and fill the selected candidates into Tuple?

Hereafter is an example of running one stripping line, `FullDSTDiMuonJpsi2MuMuDetachedLine` of Stripping17 on MC10 data. You need to use the corresponding version of DaVinci, e.g., DaVinci v29r1 for Stripping17. Please **NOTE** the prefix "Stripping" when selecting the stripping line `"StrippingFullDSTDiMuonJpsi2MuMuDetachedLine"`, while there is no "Stripping" in the name of the output location `Phys/FullDSTDiMuonJpsi2MuMuDetachedLine/Particles`. You can find a full list of stripping lines of stripping17 here, you can take the line name in the first column "Line" to select your favorite line; and take the corresponding second column "Location" as the inputs of your Tuple, by removing the name of the stream inside.

```
from StrippingConf.Configuration import StrippingConf, StrippingStream
from StrippingSettings.Utils import strippingConfiguration
from StrippingArchive.Utils import buildStreams
from StrippingArchive import strippingArchive

# Standard stripping17
stripping='stripping17'
config  = strippingConfiguration(stripping)
archive = strippingArchive(stripping)
streams = buildStreams(stripping=config, archive=archive)

# Select my line
MyStream = StrippingStream("MyStream")
MyLines = [ 'StrippingFullDSTDiMuonJpsi2MuMuDetachedLine' ]

for stream in streams:
    for line in stream.lines:
        if line.name() in MyLines:
            MyStream.appendLines( [ line ] )

# Configure Stripping
from Configurables import ProcStatusCheck
filterBadEvents = ProcStatusCheck()

sc = StrippingConf( Streams = [ MyStream ],
                    MaxCandidates = 2000,
                    AcceptBadEvents = False,
                    BadEventSelection = filterBadEvents )
```

```
# Fill Tuple
from Configurables import DecayTreeTuple
MyTuple = DecayTreeTuple("MyTuple")
MyTuple.Inputs = [ "Phys/FullDSTDiMuonJpsi2MuMuDetachedLine/Particles"]
MyTuple.Decay = "J/psi(1S) -> ^mu+ ^mu-"
MyTuple.ToolList = [
    "TupleToolEventInfo",
    "TupleToolGeometry",
    "TupleToolKinematic",
    "TupleToolPid",
    "TupleToolPrimaries",
    "TupleToolPropertime",
    "TupleToolTrackInfo"
    ]

#
# DaVinci Configuration
#
from Configurables import DaVinci
DaVinci().InputType = 'DST'
DaVinci().DataType = "2010"
DaVinci().Simulation   = True
DaVinci().EvtMax = 5000                          # Number of events
DaVinci().HistogramFile = "DVHistos.root"
DaVinci().TupleFile = "Tuple.root"
DaVinci().appendToMainSequence( [ sc.sequence() ] )
DaVinci().UserAlgorithms = [ MyTuple ]

# database
DaVinci().DDDBtag = 'head-20101206'
DaVinci().CondDBtag  = "sim-20101210-vc-md100"
```

# How do I run some stripping line on MC data and save the selected events into DST?

Hereafter is an example of running one stripping line, `FullDSTDiMuonJpsi2MuMuDetachedLine` of Stripping17 on MC10 data. It generates a DST file named `MC10.MyStream.dst`, the selected candidates are saved on

`/Event/MyStream/Phys/FullDSTDiMuonJpsi2MuMuDetachedLine/Particles`

You need to use the corresponding version of DaVinci, e.g., DaVinci v29r1 for Stripping17.

```
from StrippingConf.Configuration import StrippingConf, StrippingStream
from StrippingSettings.Utils import strippingConfiguration
from StrippingArchive.Utils import buildStreams
from StrippingArchive import strippingArchive

# Standard stripping17
stripping='stripping17'
config  = strippingConfiguration(stripping)
archive = strippingArchive(stripping)
streams = buildStreams(stripping=config, archive=archive)

# Select my line
MyStream = StrippingStream("MyStream")
MyLines = [ 'StrippingFullDSTDiMuonJpsi2MuMuDetachedLine' ]

for stream in streams:
    for line in stream.lines:
        if line.name() in MyLines:
            MyStream.appendLines( [ line ] )
```

```
# Configure Stripping
from Configurables import ProcStatusCheck
filterBadEvents = ProcStatusCheck()

sc = StrippingConf( Streams = [ MyStream ],
                    MaxCandidates = 2000,
                    AcceptBadEvents = False,
                    BadEventSelection = filterBadEvents )


# Write DST
from DSTWriters.__dev__.microdstelements import *
from DSTWriters.__dev__.Configuration import (SelDSTWriter,
                                              stripDSTStreamConf,
                                              stripDSTElements
                                              )

#
# Configuration of SelDSTWriter
#
SelDSTWriterElements = {
    'default'              : stripDSTElements()
    }


SelDSTWriterConf = {
    'default'              : stripDSTStreamConf()
    }


dstWriter = SelDSTWriter( "MyDSTWriter",
                          StreamConf = SelDSTWriterConf,
                          MicroDSTElements = SelDSTWriterElements,
                          OutputFileSuffix ='MC10',
                          SelectionSequences = sc.activeStreams()
                          )

#
# DaVinci Configuration
#
from Configurables import DaVinci
DaVinci().InputType = 'DST'
DaVinci().DataType = "2010"
DaVinci().Simulation   = True
DaVinci().EvtMax = 5000                        # Number of events
DaVinci().HistogramFile = "DVHistos.root"
DaVinci().TupleFile = "Tuple.root"
DaVinci().appendToMainSequence( [ sc.sequence() ] )
DaVinci().appendToMainSequence( [ dstWriter.sequence() ] )

# database
DaVinci().DDDBtag = 'head-20101206'
DaVinci().CondDBtag  = "sim-20101210-vc-md100"
```

**NOTE** that we've moved to the ROOT Persistency since DaVinci v29r0, the output file is in ROOT format too. Before Moore can handle the ROOT Persistency, a workaround is that you can use an older version of DaVinci, e.g., v28r5, and getpack the corresponding version of `Phys/StrippingSelections`, e.g., v3r9 for Stripping17, then configure your lines "by hand" (the following will run all lines in the DiMuon linebuilder, you can select the one you want), you need to make sure the configuration dictionary in your stripping line file is the same as the one in the `Phys/StrippingSettings`.

```
MyStream = StrippingStream("MyStream")

from StrippingSelections.StrippingDiMuonNew import DiMuonConf
```

```
from StrippingSelections.StrippingDiMuonNew import config_default as config_FullDSTDiMuon
FullDSTDiMuonConf = DiMuonConf( name = "FullDSTDiMuon", config =config_FullDSTDiMuon  )
MyStream.appendLines( FullDSTDiMuonConf.lines() )
```

# How to re-strip the stripped MC data

**NOTE**, for real data and some MC, the stripping is run in filtering mode, i.e., only the events selected by some stripping line is written out, it may not make sense to re-strip. So we just focus on how to re-strip the MC data for which the stripping was run in flagging mode. Apart from the example shown above, one just need to kill the stripping banks at the very beginning of the job:

```
from Configurables import EventNodeKiller
eventNodeKiller = EventNodeKiller('Stripkiller')
eventNodeKiller.Nodes = [ '/Event/AllStreams', '/Event/Strip' ]
#...
DaVinci().appendToMainSequence( [ eventNodeKiller ] )   # Kill old stripping banks first
```

Re-stripping MC that was persisted in the Micro-DST format requires a bit more effort. See this awesome page here

# How to write a stripping line

Stripping is no more than (Pre)Selection of your favorite channels, once you have the (Pre)Selection at hand, it should be straightforward to add a Stripping line for that. Just have a look at some file in the directory python/StrippingSelections of `Phys/StrippingSelections` package. There is also a tutorial on how to write a stripping line here. In order to be able to test your line as suggested in here you have to follow what has been done in the $STRIPPINGSELECTIONSROOT/python/StrippingSelections/StrippingB2HHBDT.py. First of all you have to ensure that a default_config object is declared among the public objects defined with your line:

```
__all__ = ('B2HHBDTLines',
           'makeB2HHBDT',
           'applyBDT',
           'default_config')
```

NOTE: default_config is a mandatory name. Then you have to define a dictionary named default_config

```
default_config = {
    'NAME'        : 'B2HHBDT',
    'WGs'         : ['Charmless'],
    'BUILDERTYPE' : 'B2HHBDTLines',
    'CONFIG'      : { 'PrescaleB2HHBDT' : 1.,
                      'MinPT'           : 1000,
                      'MinIP'           : 0.12,
                      'TrChi2'          : 3,
                      'TrGhostProb'     : 0.5,
                      'CombMassLow'     : 4600,
                      'CombMassHigh'    : 6400,
                      'DOCA'            : 0.1,
                      'BPT'             : 1200,
                      'BIP'             : 0.12,
                      'BTAU'            : 0.0006,
                      'MassLow'         : 4800,
                      'MassHigh'        : 6200,
                      'BDTCut'          : -0.3,
                      'BDTWeightsFile'  : "$TMVAWEIGHTSROOT/data/B2HHBDT.xml"
                    },
    'STREAMS'     : { 'Bhadron' : ['StrippingB2HHBDTLine']}
    }
```

Note that this dictionary has the same structure of what will go in StrippingSettings package. The only difference is the presence of the key 'NAME'. The value to give to this default_config['NAME'] must be the same that you want to use to instantiate your stripping line. For example the StrippingB2HHBDTLines will be instantiated in the following way:

```
builder = B2HHBDTLines(default_config['NAME'],default_config['CONFIG'])
```

NOTE: Up to DaVinci v35r1 (included) you have to getpack also the head of Phys/StrippingUtils

# How to test my new stripping line?

Unless advertised otherwise, it is always recommended to use the latest DaVinci to do the test

```
SetupDaVinci --build-env
getpack Phys/StrippingSelections head
```

Add your new strippping line in the `python/StrippingSelections/` directory and in the `python/StrippingSelections/__init__.py` file, then

```
cd Phys/StrippingSelections/cmt
cmt make
```

Have a copy of `Phys/StrippingSelections/tests/TestMyStrippingLineOn2012Data_Reco14.py`, and change it accordingly to test your line

```
SetupDaVinci
gaudirun.py TestMyStrippingLineOn2012Data_Reco14.py
```

it will print out the CPU timing, retention rate, etc of your Strippping line.

# How to commit your new stripping line to svn?

Getpack the head of the `Phys/StrippingSelections` package

```
getpack Phys/StrippingSelections head
```

go to the directory

```
cd Phys/StrippingSelections
```

and add your stripping line file to

```
python/StrippingSelections/
```

then document your changes in the `release.notes` and commit your changes (under the directory `$HOME/../Phys/StrippingSelections`). **Note**, depending on when you check out the head of the package, it could happen that someone committed something in between (and changed the `release.notes`), so, it would be better to check whether it is the case by running `svn status -u`, if so, update to the head again by running `svn update`:

```
svn status -u
svn update
emacs doc/release.notes &
svn commit -m "some brief and meaningful comment"
```

you can find more info in the svn guidelines.

# Where is the selected candidates of a Stripping line stored?

All the locations for almost all of Strippings (since Stripping12) can be found here.

# How to run on MC sample flagged with some version of Stripping when I have a working script for data?

Just do the following:

- Change the Stream name to "AllStreams", e.g.,
  `/Event/Dimuon/Phys/BetaSBu2JpsiKDetachedLine/Particles`
  to
  `/Event/AllStreams/Phys/BetaSBu2JpsiKDetachedLine/Particles`
- Change database tags, how to find them can be found in RecommendedTags.
- Set DaVinci().Simulation=True, and add more simulation related information as needed.

# Options with test samples:

## 2011 data:

- $STRIPPINGSELECTIONSROOT/tests/data/Reco14_2011Data_MagUp_New.py
- $STRIPPINGSELECTIONSROOT/tests/data/Reco14_2011Data_MagDn_New.py

## 2012 data:

- $STRIPPINGSELECTIONSROOT/tests/data/Reco14_Run125113.py

# How to generate xml_catalog file for stripping test sample

## Option 1

- Use a script provided by Chris Jones, that uses a fixed version of `dirac-lhcb-generate-catalog` provided by Ricardo Graciani. If the default Dirac doesn't work, please use the one as shown below:

```
SetupLHCbDirac v7r7p5
lhcb-proxy-init
/afs/cern.ch/user/j/jhe/public/stripping/scripts/GetCERNXMLCatalogue /afs/cern.ch/use
```

## Option 2

- Change the datacard obtained from BK to the old syntax
- Default Dirac may not work, use v7r3
  `SetupLHCbDirac v7r3`
- Use a fixed version of genXMLCatalog provided by Zoltan Mathe

```
~jhe/public/stripping/scripts/genXMLCatalog -v --depth=2 /afs/cern.ch/user/j/jhe/public/st
```

-- JiboHE - 12-Oct-2011

This topic: LHCb > StrippingFAQ
Topic revision: r20 - 2018-02-20 - NiklasStefanNolte