Table of Contents

How to create a new stripping line

IMPORTANT

Please do not import anything explicitly from the StrippingSelections package, which is used for development. In production the LineBuilders are copied into StrippingArchive to freeze them. Explicit imports from StrippingSelections break this scheme and may cause your line to crash if there are differences with what is put in the StrippingArchive and what is in StrippingSelections.

Writing a basic selection

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 one of the sub-directories of python/StrippingSelections/Stripping/ of the Phys/StrippingSelections package. As an simple example we use the line StrippingHb2Charged2Body.py

```
"""
Stripping options for (pre-)selecting B -> hh'

Authors: Stefano Perazzini, Lars Eklumd
"""

#########################################################################
__author__ = ['Stefano Perazzini', 'Lars Eklund']
__date__ = '21/08/2012'
__version__ = '$Revision: 1.10 $'

__all__ = ('Hb2Charged2BodyLines',
           'makeB2Charged2Body',
           'default_config')

from Gaudi.Configuration import *

from GaudiConfUtils.ConfigurableGenerators import CombineParticles
from StandardParticles                     import StdNoPIDsPions, StdLooseProtons, StdLooseKaons

from PhysSelPython.Wrappers      import Selection
from StrippingConf.StrippingLine import StrippingLine
from StrippingUtils.Utils        import LineBuilder, checkConfig
from Configurables               import SubstitutePID, FilterDesktop


default_config = {
    'NAME'        : 'Hb2Charged2Body',
    'WGs'         : ['Charmless'],
    'BUILDERTYPE' : 'Hb2Charged2BodyLines',
    'CONFIG'      : {'PrescaleB2Charged2Body'   : 1,
                     'MinPTB2Charged2Body'      : 1000,
                     'MinIPChi2B2Charged2Body'  : 12,
                     'TrChi2'                   : 3,
                     'TrGhostProb'              : 0.5,
                     'MaxPTB2Charged2Body'      : 1400,
                     'MaxIPChi2B2Charged2Body'  : 40,
                     'CombMassLow'              : 4600,
                     'CombMassHigh'             : 6000,
                     'DOCA'                     : 0.08,
                     'BPT'                      : 1200,
                     'BIPChi2B2Charged2Body'    : 12,
                     'BTAU'                     : 0.0006,
                     'MassLow'                  : 4800,
                     'MassHigh'                 : 5800
```

```
                    },
    'STREAMS'     : ['BhadronCompleteEvent']
    }

class Hb2Charged2BodyLines( LineBuilder ) :
    """Class defining the Hb -> hh stripping lines"""

    __configuration_keys__ = ('PrescaleB2Charged2Body',
                              'MinPTB2Charged2Body',
                              'MinIPChi2B2Charged2Body',
                              'TrChi2',
                              'TrGhostProb',
                              'MaxPTB2Charged2Body',
                              'MaxIPChi2B2Charged2Body',
                              'CombMassLow',
                              'CombMassHigh',
                              'DOCA',
                              'BPT',
                              'BIPChi2B2Charged2Body',
                              'BTAU',
                              'MassLow',
                              'MassHigh'
                             )

    def __init__( self,name,config ) :

        LineBuilder.__init__(self, name, config)

        B2Charged2BodyName   = name + "B2Charged2Body"

        # make the various stripping selections
        self.B2Charged2Body = makeB2Charged2Body( B2Charged2BodyName,
                                                  config['TrChi2'],
                                                  config['TrGhostProb'],
                                                  config['MinPTB2Charged2Body'],
                                                  config['MinIPChi2B2Charged2Body'],
                                                  config['MaxPTB2Charged2Body'],
                                                  config['MaxIPChi2B2Charged2Body'],
                                                  config['CombMassLow'],
                                                  config['CombMassHigh'],
                                                  config['DOCA'],
                                                  config['BPT'],
                                                  config['BIPChi2B2Charged2Body'],
                                                  config['BTAU'],
                                                  config['MassLow'],
                                                  config['MassHigh']
                                                )

        self.lineB2Charged2Body = StrippingLine( B2Charged2BodyName+"Line",
                                                 prescale  = config['PrescaleB2Charged2Body'],
                                                 selection = self.B2Charged2Body,
                                                 EnableFlavourTagging = True,
                                                 RequiredRawEvents = ["Trigger","Muon","Calo","Ri

        self.registerLine(self.lineB2Charged2Body)

def makeB2Charged2Body( name,
                        trChi2,trGhostProb,minPT,minIPChi2,
                        maxPT,maxIPChi2,combMassLow,combMassHigh,doca,
                        bPT,bIPChi2,bTAU,massLow,massHigh ) :

    _daughters_cuts = "(TRGHOSTPROB < %(trGhostProb)s) & (TRCHI2DOF < %(trChi2)s) & (PT > %(minPT

    _combination_cuts = "(AMAXCHILD(MAXTREE('pi+'==ABSID,PT)) > %(maxPT)s ) & ( AMAXCHILD(MAXTREE
ow)s * MeV) & (AM < %(combMassHigh)s * MeV)" %locals()

    _mother_cuts = "(PT > %(bPT)s * MeV) & (M > %(massLow)s * MeV) & (M < %(massHigh)s * MeV) & (
```

```
    CombineHb2Charged2Body = CombineParticles( DecayDescriptor = 'B0 -> pi+ pi-',
                                               DaughtersCuts = { "pi+" : _daughters_cuts },
                                               CombinationCut = _combination_cuts,
                                               MotherCut = _mother_cuts )

    return Selection( name,
                      Algorithm = CombineHb2Charged2Body,
                      RequiredSelections = [ StdNoPIDsPions ] )

##################################################################
```

Let's have a look at the various parts:

Author names, date and code revision

The author names should be the ones responsible for the lines, which will be contacted if some issue arises with the stripping line:

```
__author__ = ['Stefano Perazzini', 'Lars Eklund']
__date__ = '21/08/2012'
__version__ = '$Revision: 1.10 $'
```

Declare public objects

Declare public objects, ensure that a default_config object is declared among the public objects defined with your line:

```
__all__ = ('Hb2Charged2BodyLines',
           'makeB2Charged2Body',
           'default_config')
```

Import needed modules

Import python modules needed by your line

```
from Gaudi.Configuration import *

from GaudiConfUtils.ConfigurableGenerators import CombineParticles
from StandardParticles                     import StdNoPIDsPions, StdLooseProtons, StdLooseKaons

from PhysSelPython.Wrappers      import Selection
from StrippingConf.StrippingLine import StrippingLine
from StrippingUtils.Utils        import LineBuilder, checkConfig
from Configurables               import SubstitutePID, FilterDesktop
```

Default configuration dictionary

The default configuration dictionary should normally be up to date and work with the lastest version of the line builder. This dictionary is normally used during testing. Once the testing is finished and the dictionary is final you should inform your WG liaison to copy the default_config into StrippingSettings so that it will run in the Stripping (NOTE: it will not be actuamatically run in the Stripping unless you explicitely ask for it)

```
default_config = {
    'NAME'        : 'Hb2Charged2Body',
    'WGs'         : ['Charmless'],
    'BUILDERTYPE' : 'Hb2Charged2BodyLines',
    'CONFIG'      : {'PrescaleB2Charged2Body'  : 1,
                     'MinPTB2Charged2Body'     : 1000,
```

```
                    'MinIPChi2B2Charged2Body' : 12,
                    'TrChi2'                  : 3,
                    'TrGhostProb'             : 0.5,
                    'MaxPTB2Charged2Body'     : 1400,
                    'MaxIPChi2B2Charged2Body' : 40,
                    'CombMassLow'             : 4600,
                    'CombMassHigh'            : 6000,
                    'DOCA'                    : 0.08,
                    'BPT'                     : 1200,
                    'BIPChi2B2Charged2Body'   : 12,
                    'BTAU'                    : 0.0006,
                    'MassLow'                 : 4800,
                    'MassHigh'                : 5800
                   },
    'STREAMS'     : ['BhadronCompleteEvent']
    }
```

The key 'CONFIG' usually contains all the list of cuts and pre/post scalings but not only (e.g. related info, raw event etc). We advise users to put as many settings as possible in this dictionary in order to minimise the modification to the LineBuilder itself. An alternative configuration of the above default configuration is the following

```
default_config = {
    'Hb2Charged2Body' :{
       'WGs'          : ['Charmless'],
       'BUILDERTYPE' : 'Hb2Charged2BodyLines',
        'CONFIG'      : {'PrescaleB2Charged2Body'  : 1,
                    'MinPTB2Charged2Body'     : 1000,
                    'MinIPChi2B2Charged2Body' : 12,
                    'TrChi2'                  : 3,
                    'TrGhostProb'             : 0.5,
                    'MaxPTB2Charged2Body'     : 1400,
                    'MaxIPChi2B2Charged2Body' : 40,
                    'CombMassLow'             : 4600,
                    'CombMassHigh'            : 6000,
                    'DOCA'                    : 0.08,
                    'BPT'                     : 1200,
                    'BIPChi2B2Charged2Body'   : 12,
                    'BTAU'                    : 0.0006,
                    'MassLow'                 : 4800,
                    'MassHigh'                : 5800
                   },
       'STREAMS'     : ['BhadronCompleteEvent']
       }
    }
```

In this case the key value in the key 'NAME' has become a key of another dictionary. In terms of functionality and testing procedure there is no difference between the two configurations. However, the last one is useful when more then one default dictionary needs to be specified (e.g. when the same LineBuilder is used with different configuration for different analyses), as follows

```
default_config = {
    'MyName1' :{
       'WGs'          : ['SomeWG'],
       'BUILDERTYPE' : 'BuilderName',
        'CONFIG'      : {'Cut1'   : 1000,
                    'Cut2'     : 500,
                   },
       'STREAMS'     : ['SomeStream1']
       },
     'MyName2' :{
       'WGs'          : ['SomeWG'],
       'BUILDERTYPE' : 'BuilderName',
        'CONFIG'      : {'Cut1'   : 2000,
```

```
                    'Cut2'      : 100,
                   },
        'STREAMS'      : ['SomeStream2']
        }
    }
```

Within a LineBuilder several stripping lines can be defined and registered, the ***configuration_keys*** must match those in the keys in default_config['CONFIG']

```
class Hb2Charged2BodyLines( LineBuilder ) :
    """Class defining the Hb -> hh stripping lines"""

    __configuration_keys__ = ('PrescaleB2Charged2Body',
                              'MinPTB2Charged2Body',
                              'MinIPChi2B2Charged2Body',
                              'TrChi2',
                              'TrGhostProb',
                              'MaxPTB2Charged2Body',
                              'MaxIPChi2B2Charged2Body',
                              'CombMassLow',
                              'CombMassHigh',
                              'DOCA',
                              'BPT',
                              'BIPChi2B2Charged2Body',
                              'BTAU',
                              'MassLow',
                              'MassHigh'
                             )

    def __init__( self,name,config ) :

        LineBuilder.__init__(self, name, config)

        B2Charged2BodyName   = name + "B2Charged2Body"

        # make the various stripping selections
        self.B2Charged2Body = makeB2Charged2Body( B2Charged2BodyName,
                                                  config['TrChi2'],
                                                  config['TrGhostProb'],
                                                  config['MinPTB2Charged2Body'],
                                                  config['MinIPChi2B2Charged2Body'],
                                                  config['MaxPTB2Charged2Body'],
                                                  config['MaxIPChi2B2Charged2Body'],
                                                  config['CombMassLow'],
                                                  config['CombMassHigh'],
                                                  config['DOCA'],
                                                  config['BPT'],
                                                  config['BIPChi2B2Charged2Body'],
                                                  config['BTAU'],
                                                  config['MassLow'],
                                                  config['MassHigh']
                                                )

        self.lineB2Charged2Body = StrippingLine( B2Charged2BodyName+"Line",
                                                 prescale  = config['PrescaleB2Charged2Body'],
                                                 selection = self.B2Charged2Body,
                                                 EnableFlavourTagging = True,
                                                 RequiredRawEvents = ["Trigger","Muon","Calo","Ri

        self.registerLine(self.lineB2Charged2Body)
```

Helper functions

Helper functions are used to make the code more readable and modular, they are very useful if several stripping lines defined in the same line builder share the same code:

```
def makeB2Charged2Body( name,
                        trChi2,trGhostProb,minPT,minIPChi2,
                        maxPT,maxIPChi2,combMassLow,combMassHigh,doca,
                        bPT,bIPChi2,bTAU,massLow,massHigh ) :

    _daughters_cuts = "(TRGHOSTPROB < %(trGhostProb)s) & (TRCHI2DOF < %(trChi2)s) & (PT > %(minPT

    _combination_cuts = "(AMAXCHILD(MAXTREE('pi+'==ABSID,PT)) > %(maxPT)s ) & ( AMAXCHILD(MAXTREE
ow)s * MeV) & (AM < %(combMassHigh)s * MeV)" %locals()

    _mother_cuts = "(PT > %(bPT)s * MeV) & (M > %(massLow)s * MeV) & (M < %(massHigh)s * MeV) & (

    CombineHb2Charged2Body = CombineParticles( DecayDescriptor = 'B0 -> pi+ pi-',
                                               DaughtersCuts = { "pi+" : _daughters_cuts },
                                               CombinationCut = _combination_cuts,
                                               MotherCut = _mother_cuts )

    return Selection( name,
                      Algorithm = CombineHb2Charged2Body,
                      RequiredSelections = [ StdNoPIDsPions ] )
```

Send lines created by the same to different streams

In general, linebuilders create more than one stripping line. It may be the case that not all of them go to the same Stream. This possibility is taken in accont in the stripping framework and can be done by properly setting the content of 'STREAMS' in the default_config, here is an example from S21

```
'STREAMS' :{ 'Charm'     : [ 'StrippingD02KpiForPromptCharm'          ,
                             'StrippingDstarForPromptCharm'           ,
                             'StrippingDForPromptCharm'               ,
                             'StrippingDsForPromptCharm'              ,
                             'StrippingDiCharmForPromptCharm'         ,
                             'StrippingChiAndCharmForPromptCharm'     ,
                             'StrippingCharmAndWForPromptCharm'       ,
                             'StrippingDiMuonAndCharmForPromptCharm'  ,
                             'StrippingLambdaCForPromptCharm'         ,
                             'StrippingLambdaC2pKKForPromptCharm'     ,
                             'StrippingSigmaCForPromptCharm'          ,
                             'StrippingLambdaCstarForPromptCharm'     ,
                             'StrippingD02KKForPromptCharm'           ,
                             'StrippingD02pipiForPromptCharm'         ,
                             'StrippingDstarCPForPromptCharm'         ,
                             'StrippingOmegaCstarForPromptCharm'
                             ] ,
                  ##
                  'Leptonic' : [ 'StrippingDoubleDiMuonForPromptCharm'    ,
                             'StrippingDiMuonAndWForPromptCharm'      ,
                             'StrippingChiAndWForPromptCharm'         ] }
```

So the instead of stream name, the value of "STREAMS" is a dictionary where the keys are the streams and the values are the names of the stripping lines that should go on that stream.

Request specific Raw Banks

From Stripping21 onwards, by default only trigger raw banks are stored in the stripping output since they are needed for TISTOSing. Specific raw banks can anyhow be requested on a line basis. The StrippingLine

defined in the LineBuilder taken as an example requests all the available banks using the RequiredRawEvent property of the StrippingLine class. Users are expected to only request the raw banks that will be effectively needed in the final analysis.

```
self.lineB2Charged2Body = StrippingLine( B2Charged2BodyName+"Line",
                                                    prescale  = config['PrescaleB2Charged2Body'],
                                                    selection = self.B2Charged2Body,
                                                    EnableFlavourTagging = True,
                                                    RequiredRawEvents = ["Trigger","Muon","Calo","Ri
```

Valid RawBanks are "Trigger", "HC", "Muon", "Calo", "Rich", "Velo" and "Tracker" (names are case-sensitive).

Request flavour tagging information

In order to run FlavourTagging and have it saved in microDST you just have to set the flag "EnableFlavourTagging" to "True" when creating your StrippingLine (as shown in the example above). NOTE: this is only applicable for lines going to microDST, for lines going to DST this option is forced to be False, regardless of what is specified by the user in the StrippingLine.

Request Related Info

Extra information, such as isolation variables, can be stored on microDST. Detailed information on how to add this extra information to your stripping line is available here

Re-fitted primary vertex

In previous strippings this was automatically done in microDST streams, but from stripping21 onward it is disabled. If your line goes to a microDST stream and you want to have a re-fitted PV you have to specify it. For example in StrippingB2JpsiXforBeta_s.py you can find

```
combiner = CombineParticles( DecayDescriptor = DecayDescriptor,
                                        DaughtersCuts = DaughterCuts,
                                        MotherCut = PostVertexCuts,
                                        CombinationCut = PreVertexCuts,
                                        ReFitPVs = ReFitPVs)
```

What is the MDST.DST stream? How can I have my line stored also in MDST.DST stream?

The MDST.DST stream is devoted to the use case where the line in question is going to uDST. In MDST.DST will be stored the full reconstructed event and not only the candidate selected by the stripping line. MDST.DST stream will allow fast regeneration of certain things, such as new Flavour tagging, or isolation etc., for those selected candidates, that requires the full reconstructed event information. The regeneration of microDST will have to be coordinated according with the computing team (MDST.DST will not be accessible to normal users). If you want to specify that you want also the MDST.DST for your line you have to:

```
self.lineB2HHBDT = StrippingLine( B2HHBDTName+"Line",
                                             prescale  = config['PrescaleB2HHBDT'],
                                             selection = self.CutBDT,
                                             EnableFlavourTagging = True,
                                             MDSTFlag = True )
```

Reduce the time consumed by my multi-body stripping line

Vanya Beylaev developed the NBodyDecay tool. It proved to give an important gain in time consumption. Details can be found in this presentation. The use of NBodyDecay is strongly advised for multi-body lines. An example of a line using NBodyDecay to make 3-particle combinations in available here

Usage of multi-variate discriminators in the stripping

The usage of multivariate discriminators in the stripping in possible using the generic MVATool in gaudi. Other private tools are not allowed for new lines. A description of the tool is available here. An good example of a StrippingLine which cuts on a MVA discriminant is available here

How to test my stripping line

After you have written your new stripping line, it has to be tested as described here

Before I commit my stripping line

Once you have successfully tested your line inform your WG group liaison about the results of the testing and ask for permission to commit to svn. For new lines, please inform also the coordinators before committing the new code.

-- AndreaContu - 2015-02-13

This topic: LHCb > CreateNewStrippingLine
Topic revision: r10 - 2016-04-12 - ThomasLatham