Table of Contents

Gaudi Python FAQ

Introduction

Welcome to the Gaudi Python FAQ page. If you have questions (and answers) please add them to the page. The goal of this page is to create documentation for the Gaudi Python package.

Getting Started

Have a look at the following tutorials:

- Introduction to GaudiPython (Ulrich Kerzel, 35th software week, 9th December 2008)
- Seeing a DST as a Big Ntuple (Thomas Ruf)
- Package Tutorial/GaudiPythonTutor (Juan Palacios). This contains exercises and example solutions geared at simple analysis on LHCb DSTs and MicroDSTs

FAQs

How to get access and instantiate an LHCb Event class

The LHCb Event classes now live in the namespace "LHCb". In order to get access you have to use this namespace. `GaudiPythin.Bindings.gbl` represents the C++ global namespace (::) You can abbreviate the namespace with assignments:

```
from GaudiPython.Bindings import gbl
std  = gbl.std
LHCb = gbl.LHCb
```

Then you can create your own instances of LHCb event model types:

```
myMCParticle = LHCb.MCParticle()
myRecVertex = LHCb.RecVertex()
```

How to access common classes (e.g. STL containers, ROOT::Math classes etc.)

We get access to the global namespace and create shortcuts as explained above. Then we can instantiate some STL types

```
from GaudiPython.Bindings import gbl
std  = gbl.std
myvector = std.vector('double')()
particleVector = std.vector('const LHCb::Particle *')(50, 0)
```

Root::Math Generic Vector and Transformation classes, as well as LHCb-specific geometry classes and functions are available in the LHCbMath module:

```
import LHCbMath

vXYZ = LHCbMath.XYZVector(0.,1.,45)
print vXYZ.x(), " ", vXYZ.y(), " ", vXYZ.z()
```

How to run the new MC associators

From DaVinci v23r0 onwards, new MC association is available. This is implemented in terms of simple tool interfaces that lend themselves very well to interactive use in Python. GaudiPython examples are here.

How to access to DoxyGen documentation for C++ class/instances?

Starting from LHCb v22r9 it is very easy to get an easy access to the DoxyGen documentation pages for the certain classes, types, objects and instances:

```
>>> import LoKiCore.doxygenurl as doxy
>>> o = ...     ## arbitrary object
>>> doxy.browse ( o )      ## ask DoxyGen for the objects
>>> doxy.browse ( "LHCb::MCVertex")      ## ask doxyGen for the class by name
```

The idea from Thomas Ruf has been used.

How to run using a job options file several iterations

This is a minimalistic example on how to run a gaudi job (DaVinci in this case) for a number of events several times. In between the runs the user can access any information or change the algorithms or their properties.

```
from GaudiPython import AppMgr
gaudi = AppMgr( outputlevel = 3,
                            joboptions   = '$DAVINCIROOT/options/DVDC06MCParticleMaker.opts')
gaudi.initialize()

gaudi.run(10)
#---You can change the algorimths or other parameters here
gaudi.evtsel().rewind()
gaudi.run(10)
```

How to access Linker tables

The LHCb linker tables can be accessed in Python via the *eventassoc.py* module in *Event/LinkerInstances*.

First, do not forget to add in the requirements file the line

```
use LinkerInstances v* Event
```

Then the usage is standard. For the sake of example, let's assume that the variable *track* contains a VELO Track you are interested in, from the 'Rec/Track/Velo' container. A simple manipulation is:

```
>>> from eventassoc import linkedTo
>>> location = 'Rec/Track/Velo'
>>> Track = gbl.LHCb.Track
>>> MCParticle = LHCb.MCParticle
>>> LT = linkedTo(MCParticle,Track,location)
>>> LT.notFound() == False
True
>>> range = LT.range(track)
>>> range
<ROOT.vector<LHCb::MCParticle*> object at 0xfe27e90>
>>> range.size()
```

```
1L
>>> mcp = range[0]
>>> mcp
{ momentum :   (-65.73,-63.69,1777.86,1785.68)
particleID :   { pid :   211
 }
 }
```

How to do a vertex fit

First get the vertex fitter tool:

```
appMgr = gaudimodule.AppMgr(outputlevel=3)
OflineVertexFitter = appMgr.toolsvc().create('OfflineVertexFitter', interface='IVertexFit')
```

Then create the mother particle and vertex along the lines of

```
pidMother    = gaudimodule.gbl.LHCb.ParticleID(pidOfMother)
MotherVertex = gaudimodule.gbl.LHCb.Vertex()
MotherCand = gaudimodule.gbl.LHCb.Particle(pidMother)
```

Put the daughters which you want to fit into a vector of LHCb::Particle* :

```
particleVector = gaudimodule.gbl.std.vector('LHCb::Particle *')
daughterVect = particleVector()
daughterVect.push_back(dau1)
daughterVect.push_back(dauN)
```

Finally, perform the vertex fit:

```
sc = OfflineVertexFitter.fit(daughterVect,MotherCand, MotherVertex)
```

How to deal with Gaudi/AIDA histograms in GaudiPython ?

There are three major ways of dealing with Gaudi/AIDA histograms in GaudiPython

- Direct manipulation with histogram service
- Usage of functionality offered by **`HistoUtils`** module
- Access to "the nice" histogramming through the base-class inheritance (OO-spirit)

Direct manipulation with the histogram service allows to book and fill hisrogram from the simple (Gaudi)Python scripts in a relatively intuitive way:

```
# get the service (assume that gaudi is the ApplMgr object):
svc = gaudi.histosvc()

# book the histogram
h1 = svc.book('some/path','id','title',100,0.0,1.0)

# fill it  (e.g. in a loop):
for i in range(0,100): h1.fill( math.sin( i ) )
```

The module **`HistoUtils`** (available from Gaudi v19r5) provides couple of functions, which simplified a bit the booking of histograms and profiles:

```
from HistoUtils import book

```

```
# book the histogram
h1 = book('some/path','id','title',100,0.0,1.0)

# fill it  (e.g. in a loop):
for i in range(0,100): h1.fill( math.sin( i ) )
```

Also it provides "powerful fill" with implicit loop and selection:

```
from HistoUtils import fill

# book the histogram
histo = ...

# fill it  with single value:
value = ...
fill ( histo , value )

# fill it with arbitrary sequence of objects, convertible to double:
fill ( histo , [1,2,3,4,5,6,7] )

# use the sequence and apply the function:
#  fill histogram with 1*1, 2*2 , 3*3, 4*4
fill ( histo , [1,2,3,4,5] , lambda x : x*x )

# use the sequence and apply the function:
# for each track in sequence evaluate "pt()" and fill the histo
tracks = ...
fill ( histo , tracks , lambda t : t.pt() )

# use the sequnce, applythe function, but filter out even values:
fill ( histo , [1,2,3,4,5,6,7] , lambda x : x*x , lambda y : 0==y%2 )


# use the sequence and apply the function:
# for each track in sequence evaluate "p()" and fill the histogram with track momentum,
# keepingonly track with small transverse momentum:
tracks = ...
fill ( histo , tracks , lambda t : t.p() , lambda t : t.pt() < 1000 )

```

Also the module exports two functions which provides the access to the histogram by their location in Histogram Transient Store:

```
import HistoUtils

path = 'some/path/to/my/histo/ID'

# get as AIDA:
aida = HistoUtils.getAsAIDA ( path )

# get as native ROOT:
root = HistoUtils.getAsROOT( path )
```

`HistoUtils` are partly inspired by Tadashi Maeno' API from ATLAS' **`PyKernel`**

OO-spirit is described in detail here and it allows to reuse the whole functionality of easy-and-friendly histograms, including booking-on-demand. Also it is a recommended way for prototyping, since the resulting code is very easy to be converted into C++ lines using almost "1->1" correspondence.

How to deal with Gaudi N-tuples in GaudiPython ?

The direct manipulation (booking&filling of columns) with the native Gaudi N-tuples in Python seems to be a very difficult task. Up to now no good and easy disprove of this statement are known. Therefore one needs to find an alternative way. Three relatively easy options have been demonstrated

1. A direct manipulation with ROOT (T)-trees&N-tuples
2. Use of "smart-and-easy" N-tuples via **`TupleUtils`** module (starting from Gaudi version v19r5)
3. Access to "the nice" N-tuples through the base-class inheritance (OO-spirit)

Please consult ROOT manual for the first way, here we describe only the second way. The third way (OO-spirit) is described in detail here and it allows to reuse the whole functionality of easy-and-friendly N-tuples, including booking-on-demand. It is nice, simple, safe and it represents the recommended way for prototyping, since the resulting code is very easy to be converted into C++ lines using almost "1->1" correspondence.

The module **`GaudiPython.TupleUtils`** (appears in Gaudi v19r5) contains essentially one important function **`nTuple`** :

```
import GaudiPython
import GaudiPython.TupleUtils as TupleUtils
nTuple = TupleUtils.nTuple

gaudi = GaudiPython.AppMgr()
gaudi.HistogramPersistency = 'ROOT'
ntSvc = GaudiPython.iService ( 'NTupleSvc' )
ntSvc.Output = [ "MYLUN1 DATAFILE='TupleEx4_1.root' OPT='NEW'"]
gaudi.config()
gaudi.initialize()
# get N-tuple (book-on-demand)
t = nTuple ( 'the/path/to/the/directory' , ## path
             'MyNtupleID' ,  ## the literal ID of the tuple
             'It is the title for my N-tuple ', ## title
              LUN = 'MYLUN1' ) ## logical unit

# fill it with data e.g. trivial scalar columns:
import math
for i in range ( 1 , 1000 ) :
   t.column ( 'i' , i )               ## integer scalar
   t.column ( 'a' , math.sin(i) )     ## float scalar
   t.column ( 'b' , math.cos(i) )     ## one more float scalar
   t.write ()                         ## commit the row

```

It is important at the end of the job to release all implicitely aquired n-tuples:

```
import TupleUtils

# get the list of "active" N-tuples:
print TupleUtils.activeTuples()

# release all "active" N-tuples:
TupleUtils.releaseTuples ()
```

How to access Relation tables in (Gaudi)Python?

There are many Relation tables flying around in Gaudi/LHCb software. They are used in many areas:

1. As representationof Monte Carlo truth links for Calorimeter objects

2. As the dynamic extension of recontruction classes, e.g. χ^2-matching of Calorimeter clusters with recontructed tracks
3. As the main representation of Monte Carlo links for (Proto)Particles for LoKi

The relation tables provides easy, intuitive and efficient way for relation between any objects in Gaudi. The python interface looks very similar to C++ interface. E.g. the following example shows how one can use the relation table of C++ type **Relation::RelationWeighted<LHCb::CaloCluster,LHCb::MCParticle,float>** for selection of Monte Carlo "merged" neutral pions.

```
#!/usr/bin/env python2.4
# =============================================================================
## import everything from bender
from Bender.MainMC import *
# =============================================================================
## Simple examples of manipulations with relation tables
#  @author Vanya BELYAEV ibelyaev@physics.syr.edu
#  @date 2007-09-26
class MergedPi0(AlgoMC) :
    """ simple class to plot dikaon mass peak """

    ## standard constructor
    def __init__ ( self , name = 'MergedPi0' ) :
        """ Standard constructor """
        return AlgoMC.__init__ ( self , name )

    ## standard mehtod for analyses
    def analyse( self ) :
        """ Standard method for Analysis """

        finder = self.mcFinder(" pi0->2gamma MC-finder")

        mcpi0 = finder.find ( "pi0 -> gamma gamma" ) ;

        if mcpi0.empty() : return self.Warning("No MC-pi0 is found (1)", SUCCESS )

        #get only pi0s, which satisfy the criteria:
        #1) large Pt
        mc1 = MCPT > 500                    # * Gaudi.Units.MeV
        # 2) valid origin vertex
        mc2 = MCOVALID
        # 2) vertex near the primary vertex
        mc3 = abs ( MCVFASPF( MCVZ ) ) < 500 # * Gaudi.Units.mm

        mccut = mc1 & mc2 & mc3

        mcpi = self.mcselect ( "mcpi" , mcpi0 , mccut )

        if mcpi.empty() : return self.Warning ( "No MC-pi0 are found (2)", SUCCESS )

        # get the relation table from TES
        table = self.get( "Relations/" + "Rec/Calo/Clusters" ) # LHCb::CaloClusterLocation::

        #invert the table(create the inverse table) for local usage
        iTable = cpp.Relations.RelationWeighted(LHCb.MCParticle,LHCb.CaloCluster,'float')
        itable = iTable( table , 1 )

        # consruct "Ecal-acceptance" cuts
        outer  = ( abs(MCPY/MCPZ) < 3.00/12.5 ) & ( abs(MCPX/MCPZ) < 4.00/12.5 )
        inner  = ( abs(MCPY/MCPZ) > 0.32/12.5 ) | ( abs(MCPX/MCPZ) > 0.32/12.5 )
        accept = outer &inner

        # loop over mcpi0:
        for pi0 in mcpi :

            # get daughter MC-photons
```

```
1560            gamma1 = pi0.child(1)
1570            if not gamma1 : continue
1580            gamma2 = pi0.child(2)
1590            if not gamma2 : continue
1600
1610            # both MC-photons in Ecal acceptance
1620            if not accept ( gamma1 ) : continue
1630            if not accept ( gamma2 ) : continue
1640
1650            pt   = MCPT ( pi0 ) / 1000
1660            mnpt = min( MCPT ( gamma1 ) , MCPT ( gamma2 ) ) / 1000
1670
1680            self.plot (   pt , "ALL pi0->gamma gamma                      " , 0 ,  5 )
1690            self.plot ( mnpt , "ALL pi0->gamma gamma : min pt of photon " , 0 ,  5 )
1700
1710            clus1 = itable.relations ( gamma1 )
1720            clus2 = itable.relations ( gamma1 )
1730
1740            # each photon have some associated cluster(s)  in ECAL
1750            if clus1.empty() or clus2.empty() : continue
1760
1770            self.plot (   pt , "ECAL pi0->gamma gamma                      " , 0 ,  5 )
1780            self.plot ( mnpt , "ECAL pi0->gamma gamma : min pt of photon " , 0 ,  5 )
1790
1800            # select only 1 or 2-cluster pi0s
1810            clus0 = itable.relations ( pi0 )
1820            if 1 != clus0.size() and 2 != clus0.size() : continue
1830
1840            self.plot (   pt , " 1||2 pi0->gamma gamma                      " , 0 ,  5 )
1850            self.plot ( mnpt , " 1||2 pi0->gamma gamma : min pt of photon " , 0 ,  5 )
1860
1870            # select only true "2-cluster" pi0
1880            if 2 == clus0.size() and 1 == clus1.size() and 1 == clus2.size() and clus1.front
1890                self.plot (   pt , " 2 pi0->gamma gamma                      " , 0 ,  5 )
1900                self.plot ( mnpt , " 2 pi0->gamma gamma : min pt of photon " , 0 ,  5 )
1910
1920            # select only true "1-cluster" pi0
1930            if 1 == clus0.size() and 1 == clus1.size() and 1 == clus2.size() and clus1.front
1940                self.plot (   pt , " 1 pi0->gamma gamma                      " , 0 ,  5 )
1950                self.plot ( mnpt , " 1 pi0->gamma gamma : min pt of photon " , 0 ,  5 )
1960
1970        return SUCCESS
1980# =============================================================================
1990
```

The example illustrate:

- retrival the relation table from Transient Event Store (the line `01410`)
- inversion of table (on-flight conversion to the C++ type **`Relations::RelationWeigted<LHCb::MCParticle,LHCb::CaloCluster,float>`** , see the lines `01440-01450`)
- selection of $\pi^0 \rightarrow \gamma\gamma$, which:
    - satisfy the decay pattern "pi0 -> gamma gamma" (the line `01220`)
    - have an origin vertex within +-50 centimeters in z-direction around the primary vertex (the line `01320`)
    - each of the photon is in the *geometrical* acceptance of Ecal (the lines `01620-01630`)
- Retrieve from the relation table the number of associated Ecal clusters for π^0 and each of the daughter photons (the lines `01710-01720&01810`)
- makes the plots of the transverse momentum of the π^0 and the minimal pt of dauhter photons for each case.

In addition one can make an explict loop e.g. through all associated clusters e.g. compare with lines `01800-01810` in the previous listing:

```
# select only 1 or 2-cluster pi0s
            clus0 = itable.relations ( pi0 )
            # make explicit loop over related clusters:
            for link in clus0 :
                     #get the related cluster  of type LHCb::CaloCluster
                     cluster = link.to()
                     #get the weigth for the relation (cumulated energy deposition from the
                     weight = link.weight()

```

It is worth to compare these lines with corrresponding C++ example from **Ex/LoKiExample** package, see the file **$LOKIEXAMPLEROOT/src/LoKi_MCMergedPi0s.cpp**

How to access `LHCb::Track` -> MC truth Relation tables in (Gaudi)Python?

To access Relation tables for `LHCb::Track`-> MC Truth in python one needs to activate "on-demand" conversion of Linker objects into Relation tables. it can be done just in one line:

```
# activate  automatic "on-demand" converison of  LHcbTrack->MC Linker objects into Relation
import LoKiPhysMC.Track2MC_Configuration

# OPTIONALLY: decorate the relation tables,e.g. for easy iteration
import Relations.Rels

# OPTIONALLY: decorate MC-particles for "nice" methods
import LoKiMC.MC
```

As soon as it is done, the rest is trivial. e.g. exploiting "direct" relations (`LHCb::Track` -> MC) :

```
# get the tracks for Transient Event Store
tracks = evt['Rec/Track/Best']
print ' #number of tracks: ', tracks.size()

# get the relation table from the Transient Event Store
table = evt['Relations/Rec/Track/Default']
print ' Relation table, # of links', table.relations().size()

# loop over the tracks
for track in tracks :

 # for each track get related MC-particles:
 mcps = table.relations ( track )

  # check the number of related particles:
  if mcps.empty() : continue
  print ' # of related MC-Particles ', mcps.size()

  # get the first related particle:
  mcp = mcps[0]._to()

  # get the associetd weight
  weight = mcps[0].weight()

  print ' the first associated particle ', mcp.pname() , weight
```

The inverse relations (MC -> `LHCb::Track`) are also trivial:

```
# get MC-particles for TES
mcparticles  = evt['MC/Particles']
print ' #number of tracks: ', tracks.size()

# get the relation table from the Transient Event Store
table = evt['Relations/Rec/Trac/Default']
```

```
print ' Relation table, # of links', table.relations().size()

# get 'inverse'-view for the relation table:
itable = table.inverse()

# loop over mc-particles
for mcp in mcparticles :

    trks = itable.relations ( mcp )

    print ' # of related tracks: ', trks.size()

    if trks.empty() : continue

    # get the first related track:

    track = trks[0].to()
    weight = trks[0].weight()

    print ' the first associated track ', track.key() , weight
```

The corresponding example in Bender is attached here

Note that very similar examle in C++ is provided here.

-- Vanya Belyaev - 06-May-2k+10

-- Vanya Belyaev - 30-Apr-2k+10

-- StefanRoiser - 13 Apr 2006

-- Vanya Belyaev - 25 Sep 2k+7

-- JuanPalacios - 27 Jun 2009

This topic: LHCb > GaudiPython
Topic revision: r27 - 2013-06-25 - JaapPanman