11/2018

AGATA Data Analysis User's guide
Global Level Processing

AGATA Data Analysis Team

This document provides a guide to help the users analyzing the AGATA data produced at the global level processing i.e. for building or merging events and store root Trees.

The last version of this document, provided by the AGATA Data Analysis Team, can be found on ATRIUM (https://atrium.in2p3.fr/). Just click on WORKSPACE (ARBORESCENCE) and follow the tree on the left side (see the following picture) to reach the DataProcessing entry point.

Contents

People involved in this document:

D. Bazzacco, A. Boston, E. Clement, N. Dosmes, J. Dudouet, A. Gadea, F. Holloway, L. Hongjie, A. Korichi, N. Lalovic, E. Legay, J. Ljungvall, C. Michelagnoli, R. Perez, D. Ralet, M. Siciliano and O. Stezowski

NOTE: Please for any comment/question/suggestion contact agata{at}ipnl.in2p3.fr

Some general statements

AGATA Forum

The AGATA Forum http://agata.in2p3.fr/forum/ is a dedicated place for any question, issue, or discussion concerning the AGATA data analysis.

NARVAL, actors and the AGAPRO package (including Femul)

See the local level processing user's guide

The different actors processing the data flow

The EventBuilder actor

This actor is in charge to build AGATA events in a given time window.

The following topology file allows to build AGATA events from a list of 12 crystals, and store the result in Builder.*adf files:

```
LOOP CRY 00B 00C 01B 01C 04B 04C 05A 05B 05C 06A 06B 06C

Chain 3     CRY
Producer    CrystalProducerATCA
Filter      PreprocessingFilterPSA
Dispatcher  EventBuilder
ENDLOOP

Chain 2     Builder/
Builder     EventBuilder
Consumer    BasicAFC
```

To configure the EventBuilder actor, use the "gen_conf.py" file, here is an example of the part dedicated to the EventBuilder:

```
#########################
EventBuilder=(
"ActualClass            EventBuilder",         # name of the used daughter class
"SaveDataDir            $SAVEDIR/$BUILDER",    # Out/Builder
"Window                 10",                   # Coincidence window in timestamp unit
"keyIn                  data:psa",             # key of 1st queue.
"keyIn                  data:psa",             # key of 2nd queue.
"keyOut                 event:data:psa",       # key of the output frame default is event:data.
"MinFold                1",                    # Minimum number of AGATA crystals in the event
"Verbose",                                     # more verbose terminal-output
)
```

The BasicATSB actor

Similarly to the EventBuilder, this actor is in charge of building events in a giving time window. But on the contrary to the EventBuilder, this actor is not a Dispatcher but a Filter. This means that it requires as input a single data flow, where successive frames are stored but not built into events. This actor has been used for example for the NEDA/DIAMANT campaign.

The following topology file allows to build NEDA events and store the result in Builder_neda_.*adf files:

```
Chain 2   neda/
Producer  BasicAFP
Filter    BasicATSB
Consumer  BasicAFC
```

To configure the BasicATSB actor, use the “gen_conf.py” file, here is an example of the part dedicated to the BasicATSB:

```
#########################
BasicATSB_NEDA=(
"ActualClass           BasicATSB",              # name of the used daughter class
"timewindow            20",                     # Coincidence window in timestamp unit
"ikey                  data:ranc0  1  152",     # input key, Min multiplicity, Max multiplicity
"okey                  event:data:ranc0",       # output key
)
```

For the NEDA/DIAMANT campaign, it has been decided to attribute to NEDA the key “anc0”, and for DIAMANT, the key “anc1”. This needs to be adapted according to that for diamant.

The EventMerger actor

This actor is in charge to build global events, including AGATA and ancillaries. Actually, the actor is the same than for the EventBuilder, this is only its configuration that needs to be adapted.

The following topology file allows to merger AGATA events from a list of 12 crystals, neda and diamant data, apply the tracking (more details in the following) and store the result in a root tree (more details in the following):

```
LOOP CRY 00B 00C 01B 01C 04B 04C 05A 05B 05C 06A 06B 06C

Chain 3     CRY
Producer    CrystalProducerATCA
Filter      PreprocessingFilterPSA
Dispatcher  EventBuilder
ENDLOOP

Chain 2     Builder/
Builder     EventBuilder
Dispatcher  EventMerger

Chain 3     neda/
Producer    BasicAFP
Filter      BasicATSB
Dispatcher  EventMerger

Chain 4     diamant/
Producer    BasicAFP
Filter      BasicATSB
```

```
Dispatcher  EventMerger

Chain 3     Merger/
Builder     EventMerger
Filter      TrackingFilterOFT
Consumer    TreeBuilder
```

To configure the EventMerger actor, use the “gen_conf.py” file, here is an example of the part dedicated to the EventMerger:

```
#########################
EventBuilder=(
"ActualClass            EventMerger",           # name of the used daughter class
"SaveDataDir            $SAVEDIR/$MERGER",      # Out/Merger
"Window                 30",                    # Coincidence window in timestamp unit
"keyIn                  event:data:psa",        # key of 1st queue.
"keyIn                  event:data:ranc0",      # key of 2nd queue.
"keyIn                  event:data:ranc1",      # key of 3rd queue.
"keyOut                 event:data",            # key of the output frame default is event:data.
"MandatoryKey           event:data:psa",        # For Merger, key that needs to be triggered to
validate the event
"MandatoryKey           event:data:ranc0",      # For Merger, key that needs to be triggered to
"MinFold                2",                     # Minimum number of AGATA crystals in the event
"Verbose",                                      # more verbose terminal-output
)
```

Note: In this case, it is required to have at least agata and neda frames in the event to pass the EventMerger, using the option “MandatoryKey”.

The TrackingFilter actor

This actor is in charge perform the tracking algorithm. It can be done directly after the EventBuilder, if no EventMerger is used, or after the EventMerger. Two tracking algorithm are available:

- The Orsay Forward Tracking (default): A. Lopez-Martens, et al., Nuclear Instruments and Methods in Physics Research Section A 533 (2004) 454.
- The Mars Gamma-Ray Tracking: D. Bazzacco, Nuclear Physics A 746 (2004) 248 (Proceedings of the 2029 Sixth International Conference on Radioactive Nuclear Beams (RNB6)).

More information on the tracking algorithms can be found on ATRIUM, in the materials of the AGATA Analysis workshops (https://atrium.in2p3.fr/f8d6f1ef-46db-43ab-b0ad-fed404f3c784)

To configure the TrackingFilter actor, use the “gen_conf.py” file, here is an example of the part dedicated to the Tracking:

```
#########################
TrackingFilter=(
"ActualClass            TrackingFilterOFT", # TrackingFilterOFT or TrackingFilterMGT
"SaveDataDir            $SAVEDIR/$MERGER",  # $Merger or $BUILDER
"EnergyGain             4",                 # channels/keV of the calibrated energy spectra
"OftParams              0.05 0.02 0. 0.",   # minprobtrack minprobsing sigma_thet (no more used)
cluster_max_angle_reduction_factor
"SourcePosition         0 0 0",             # position of source with respect to the center of AGATA
)
```

Note: In the first version of OFT, the three parameters were minprobtrack, minprobsing and sigma_theta. The new version is now calculating automatically the sigma_theta value. This option is then no more used, but for backward compatibilities with old gen_conf.py files, the parameter is still present. A 4th parameter is now available, corresponding to a reduction factor of the cluster angle (default value is 0).

The TreeBuilder actor

In the past, the replay was performed up to the tracking and then stored in Tracked.*adf files. The ROOT Trees were then generated using the GammaWare toolkit (still possible by putting a BasicAFC consumer instead of the TreeBuilder in the previous Topology). But to avoid this two step method and gain a factor 2 in time processing, it has been decided to build the ROOT Trees directly at the end of the replay process, using the TreeBuilder actor.

Note: This actor has been developed during the NEDA/DIAMANT 2018 campaign at GANIL and has been upgraded after the campaign to be generic to any ancillary detector. The updated code, and the configuration of the actor from the software is then different to the version used on the GANIL site during the campaign.

To configure the TreeBuilder actor, use the “gen_conf.py” file, here is an example of the part dedicated to the TreeBuilder (example for the NEDA/DIAMANT campaign):

```
#########################
TreeBuilder=(
"ActualClass  TreeBuilder",                              # name of the used daugther class
"SaveDataDir  $SAVEDIR/$ANALYSIS AgNedDiam_ TreeMaster", # Saved in Out/Analysis, Files named
AgNedDiamTree_*.root, TreeName is TreeMaster"
"AddDetector  AGATA_BUILDER  event:data:psa   1",        # Add PSA hits in the Tree
"AddDetector  AGATA_TRACKING data:tracked     0",        # Add tracked gamma-rays in the Tree
"AddDetector  NEDA           event:data:ranc0 1",        # Add NEDA branches in the Tree
"AddDetector  DIAMANT        event:data:ranc1 0",        # Add DIAMANT branches in the Tree
)
```

The AddDetector parameter is used to add in the Tree the different part of the data analysis. It is used as follows: “AddDetector DetectorType InputKey Mode”:

- The Detector type can be:
  - AGATA_BUILDER: all data related to psa hits information (Builder output)
  - AGATA_TRACKING: all data related to the tracking output
  - NEDA: NEDA data
  - DIAMANT: DIAMANT data
- The input key corresponds to the type of ADF key that needs to be triggered:
  - AGATA_BUILDER: event:data:psa
  - AGATA_TRACKING: data:tracked
  - NEDA: event:data:ranc0
  - DIAMANT: event:data:ranc1

- The Mode corresponds to the trigger parameters:
    - -1: the frame MUST NOT BE PRESENT (we discard all events related to this key)
    - 0: the frame can be present
    - 1: the frame MUST BE PRESENT (event not written if not)

If you need to modify the code to add, remove or modify ROOT branches information, the C++ classes (in the agapro package) are located in agapro/consumers/TreeBuilder/