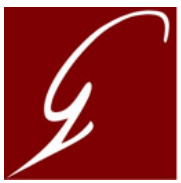
The distributed Acquisition system used at LNL: XDAQ

Alain Goasduff

University of Padova - INFN Padova

Agata Week 2019
Sept. 19, 2019

UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Dipartimento
di Fisica
e Astronomia
Galileo Galilei

XDAQ Generalities

- Software platform developed at CERN

XDAQ Generalities

- Software platform developed at CERN
- XDAQ provides:
  - Platform independent services,
  - Peer-to-peer communication between the actors
  - SOAP for configuration and control.

XDAQ Generalities

- Software platform developed at CERN
- XDAQ provides:
  - Platform independent services,
  - Peer-to-peer communication between the actors
  - SOAP for configuration and control.
- Main Features:
  - Device access
  - Configuration, control and monitoring
  - Maintainability and portability (releases are now available via YUM)
  - Scalability.

XDAQ Generalities

- Software platform developed at CERN
- XDAQ provides:
  - Platform independent services,
  - Peer-to-peer communication between the actors
  - SOAP for configuration and control.
- Main Features:
  - Device access
  - Configuration, control and monitoring
  - Maintainability and portability (releases are now available via YUM)
  - Scalability.
- GALILEO DAQ based on the release 11 $\rightarrow$ **SLC6**

XDAQ Generalities

- Software platform developed at CERN
- XDAQ provides:
  - Platform independent services,
  - Peer-to-peer communication between the actors
  - SOAP for configuration and control.
- Main Features:
  - Device access
  - Configuration, control and monitoring
  - Maintainability and portability (releases are now available via YUM)
  - Scalability.
- GALILEO DAQ based on the release 11 $\rightarrow$ **SLC6**
- Starting from release 14.0 $\rightarrow$ compatible **CentOS7**

Practical example: The data acquisition of the GALILEO γ-ray array

The GALILEO array in few words

- 25 Compton-Suppressed HPGe $\Longrightarrow\sim$ 300 kHz
- Ancillaries: EUCLIDES, SPIDER, $LaBr_3$, ... $\Longrightarrow\sim$ 50 kHz - 200 kHz ...

Practical example: The data acquisition of the GALILEO γ-ray array

The GALILEO array in few words

- 25 Compton-Suppressed HPGe $\Longrightarrow\sim$ 300 kHz
- Ancillaries: EUCLIDES, SPIDER, $LaBr_3$, ... $\Longrightarrow\sim$ 50 kHz - 200 kHz ...

- Readout: [DigiOpt-12 + GGPs]

Practical example: The data acquisition of the GALILEO γ-ray array

The GALILEO array in few words

- 25 Compton-Suppressed HPGe $\Longrightarrow\sim$ 300 kHz
- Ancillaries: EUCLIDES, SPIDER, $LaBr_3$, ... $\Longrightarrow\sim$ 50 kHz - 200 kHz ... , NeutronWall $\Longrightarrow\sim$ 300 kHz
- Readout: [DigiOpt-12 + GGPs] + [VME + AGAVA]

Practical example: The data acquisition of the GALILEO γ-ray array

The GALILEO array in few words

- 25 Compton-Suppressed HPGe $\Longrightarrow\sim$ 300 kHz
- Ancillaries: EUCLIDES, SPIDER, $LaBr_3$, ... $\Longrightarrow\sim$ 50 kHz - 200 kHz ... , NeutronWall $\Longrightarrow\sim$ 300 kHz
- Readout: [DigiOpt-12 + GGPs] + [VME + AGAVA] + [V1730]
- Synchronization: GTS System using IPBUS

Practical example: The data acquisition of the GALILEO γ-ray array

The GALILEO array in few words

- 25 Compton-Suppressed HPGe $\Longrightarrow\sim$ 300 kHz
- Ancillaries: EUCLIDES, SPIDER, $LaBr_3$, ... $\Longrightarrow\sim$ 50 kHz - 200 kHz ... , NeutronWall $\Longrightarrow\sim$ 300 kHz
- Readout: [DigiOpt-12 + GGPs] + [VME + AGAVA] + [V1730]
- Synchronization: GTS System using IPBUS

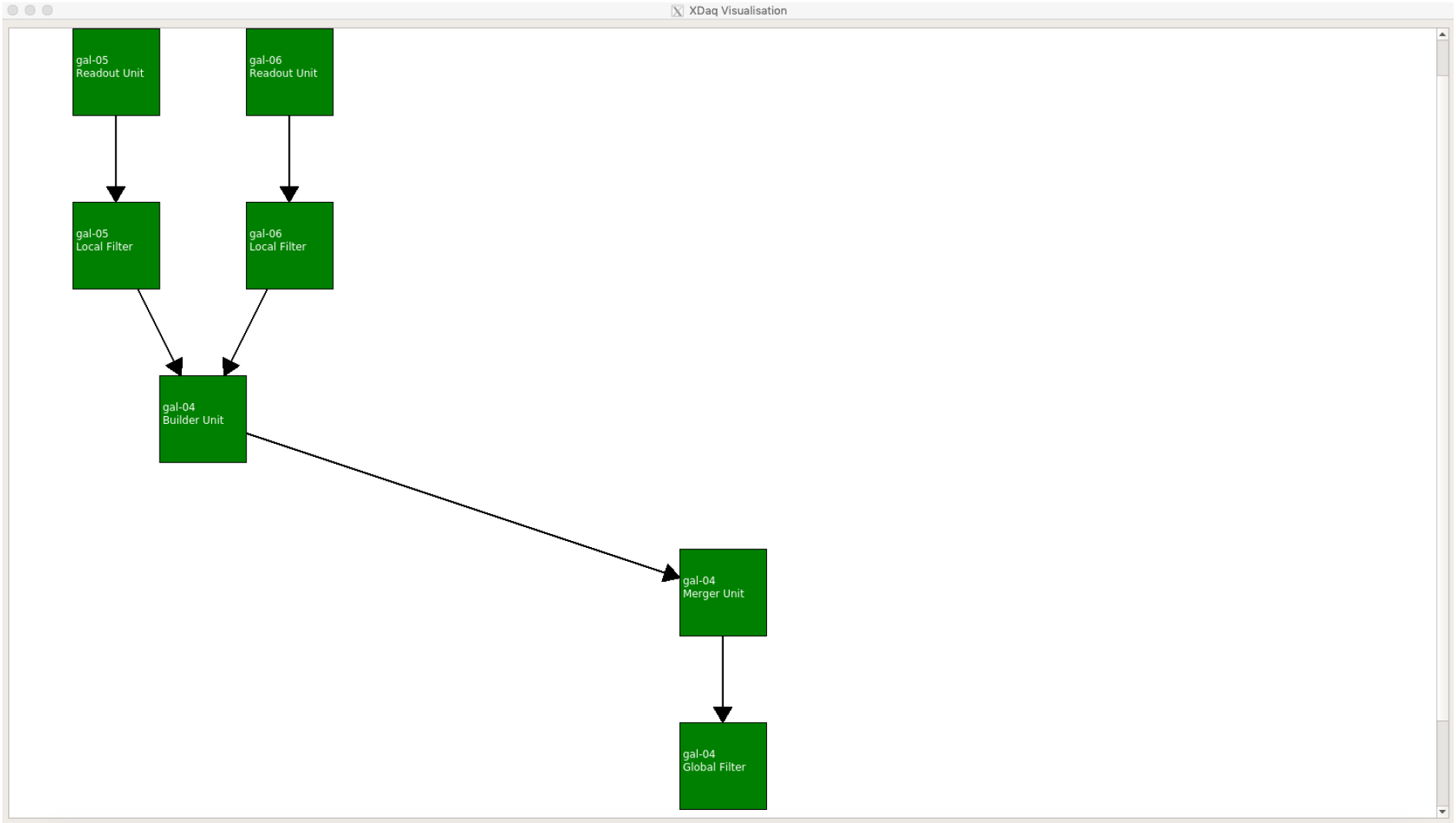
All of this translates into ONE single XML files (similar to Narval/DCOD approach)

- Definition of the different actors $\rightarrow$ RU, LF, BU, MU, GF
- Network connections between the actors
- Parameters (buffer size, data file, spy, ...)

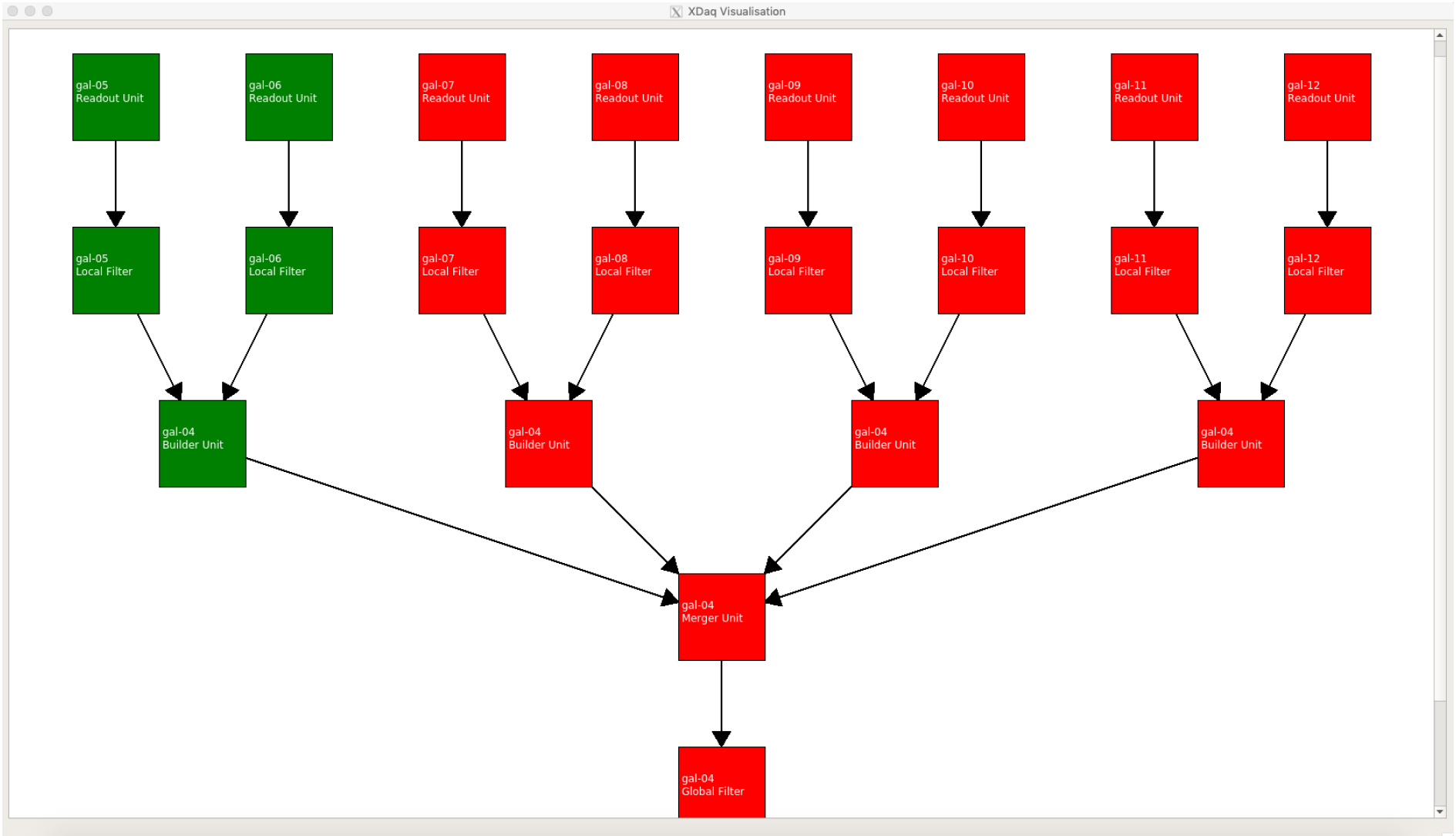
The GALILEO Topology file

```
<?xml version="1.0"?>
<xc:Partition xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
              xmlns:xc="http://xdaq.web.cern.ch/xdaq/xsd/2004/XMLConfiguration-30"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

        <i2o:protocol xmlns:i2o="http://xdaq.web.cern.ch/xdaq/xsd/2004/I2OConfiguration-30">
       <i2o:target class="rubuilder::ru::Application" instance="0" tid="100"/>
              <i2o:target class="rubuilder::ru::Application" instance="1" tid="110"/>
              <i2o:target class="LocalFilter" instance="0" tid="200"/>
              <i2o:target class="LocalFilter" instance="1" tid="210"/>
              <i2o:target class="rubuilder::bu::Application" instance="0" tid="500"/>
              <i2o:target class="rubuilder::merger::Application" instance="0" tid="600"/>
              <i2o:target class="GlobalFilter" instance="0" tid="610"/>
        </i2o:protocol>

        <xc:Context url="http://gal-05.lnl.infn.it:50000">
              <xc:Module>$XDAQ_ROOT/lib/libxdaq2rc.so</xc:Module>
              <xc:Endpoint protocol="atcp_00" service="i2o" hostname="192.168.45.7" port="40000" network="net_atcp0"/>
              <xc:Application class="pt::atcp::rcvl_00::PeerTransportATCP" id="10" instance="0" network="local">
                    <properties xmlns="urn:xdaq-application:pt::atcp::rcvl_00::PeerTransportATCP" xsi:type="soapenc:Struct">
                          <syncronousSend xsi:type="xsd:boolean">true</syncronousSend>
                          <blockingReceive xsi:type="xsd:boolean">true</blockingReceive>
                          <nonBlockingConnection xsi:type="xsd:boolean">false</nonBlockingConnection>
                          <maxPacketSize xsi:type="xsd:unsignedInt">262140</maxPacketSize>
                    </properties>
              </xc:Application>
              <xc:Application class="rubuilder::ru::Application" id="30" instance="0" network="local">
                    <properties xmlns="urn:xdaq-application:rubuilder::ru::Application" xsi:type="soapenc:Struct">
                          <runNumber xsi:type="xsd:unsignedInt">0</runNumber>
                          <inputFilepath xsi:type="xsd:string">/galileodisks/rData/rudata_gal-05</inputFilepath>
                          <detInputMask xsi:type="xsd:string">ff</detInputMask>
                          <dataOutput soapenc:arrayType="xsd:ur-type[2]" xsi:type="soapenc:Array">
                                <item soapenc:position="[0]" xsi:type="xsd:string">network</item>
                                <item soapenc:position="[1]" xsi:type="xsd:string">local</item>
                          </dataOutput>
                          <packetLength xsi:type="xsd:unsignedInt">464</packetLength>
                          <driverBuffLength xsi:type="xsd:unsignedInt">928000</driverBuffLength>
                          <writeDataFile xsi:type="xsd:boolean">false</writeDataFile>
                          <idleEventOnFile xsi:type="xsd:boolean">false</idleEventOnFile>
                          <dfile xsi:type="xsd:boolean">false</dfile>
                          <lfile xsi:type="xsd:boolean">false</lfile>
                          <sfile xsi:type="xsd:boolean">false</sfile>
                          <outputFilepath xsi:type="xsd:string">/galileodisks/xData/rudata</outputFilepath>
                          <LocalFilterInstances soapenc:arrayType="xsd:ur-type[1]" xsi:type="soapenc:Array">
                                <item soapenc:position="[0]" xsi:type="xsd:unsignedInt">0</item>
                          </LocalFilterInstances>
                    </properties>
                    <xc:Unicast instance="0" class="LocalFilter" network="net_atcp0"/>
              </xc:Application>
              <xc:Module>$XDAQ_ROOT/lib/libptatcprcvl00.so</xc:Module>
              <xc:Module>$XGAL_ROOT/lib/librubuilderutils.so</xc:Module>
              <xc:Module>$XGAL_ROOT/lib/librubuilderru.so</xc:Module>
        </xc:Context>

        ...

</xc:Partition>
```

The GALILEO Topology



The GALILEO Topology



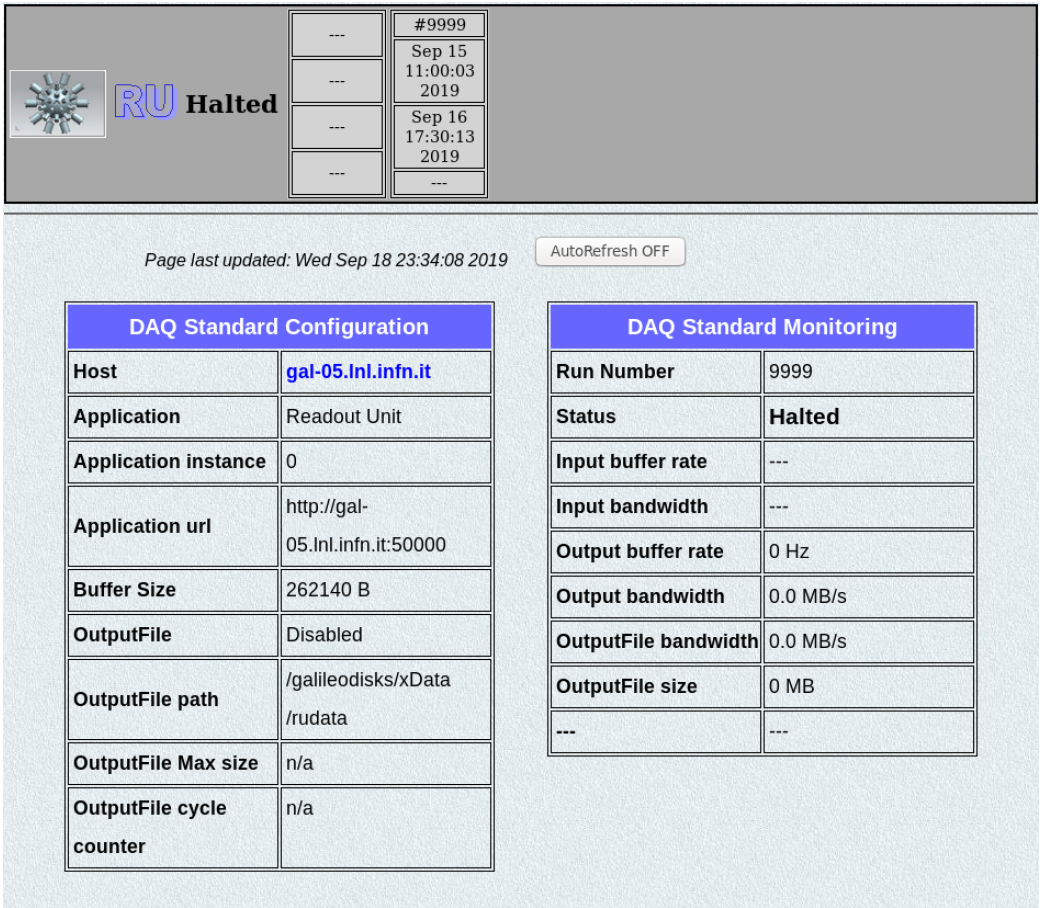
Readout units (RU)

Interface to the front-end electronic:

- GGP readout
- VME readout for ancillaries

$\Longrightarrow$ Application developed in C++

Accessible web interface with the important parameters



RU Halted

---	#9999
---	Sep 15 11:00:03 2019
---	Sep 16 17:30:13 2019
---	---

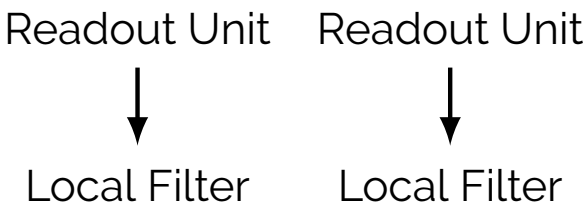
Page last updated: Wed Sep 18 23:34:08 2019 AutoRefresh OFF

DAQ Standard Configuration	
Host	gal-05.lnl.infn.it
Application	Readout Unit
Application instance	0
Application url	http://gal-05.lnl.infn.it:50000
Buffer Size	262140 B
OutputFile	Disabled
OutputFile path	/galileodisks/xData/rudata
OutputFile Max size	n/a
OutputFile cycle counter	n/a

DAQ Standard Monitoring	
Run Number	9999
Status	**Halted**
Input buffer rate	---
Input bandwidth	---
Output buffer rate	0 Hz
Output bandwidth	0.0 MB/s
OutputFile bandwidth	0.0 MB/s
OutputFile size	0 MB
---	---

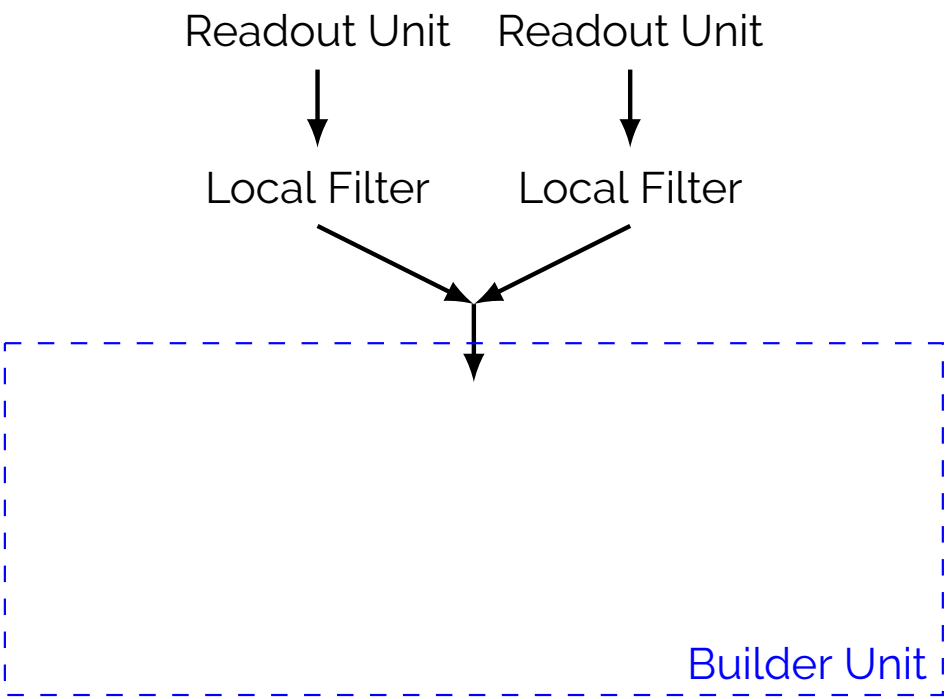
Builder units (BU, MU)

Event reconstruction based on timestamp information



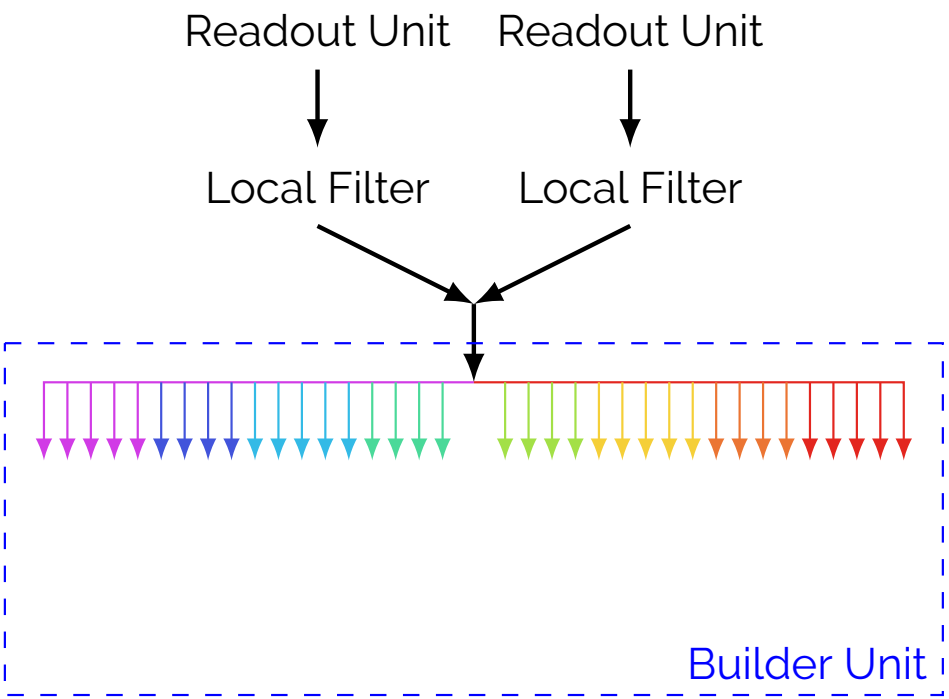
Builder units (BU, MU)

Event reconstruction based on timestamp information



Builder units (BU, MU)

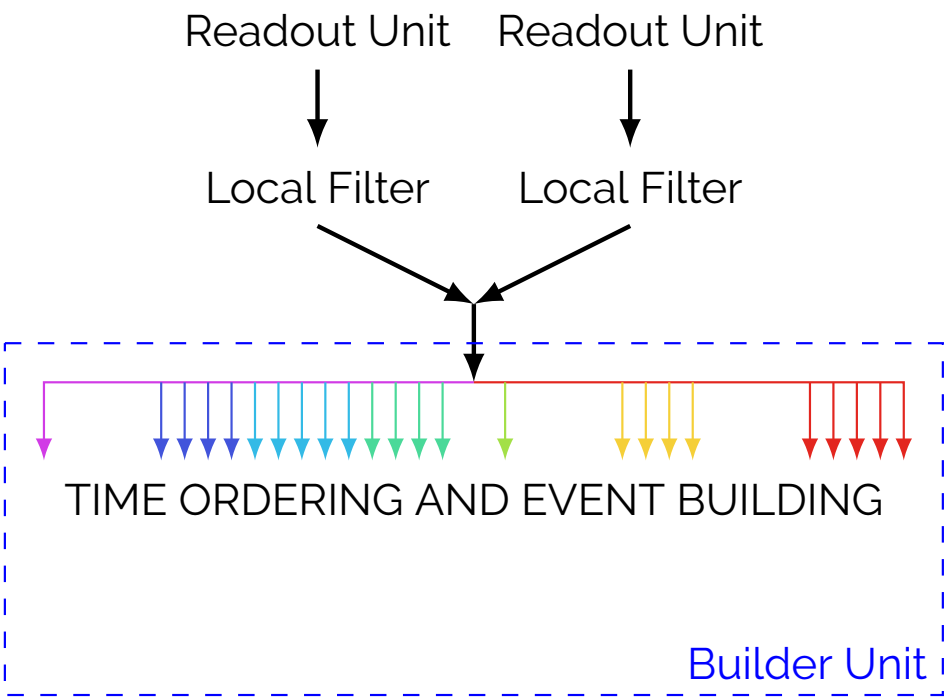
Event reconstruction based on timestamp information



- Configurable inputs

Builder units (BU, MU)

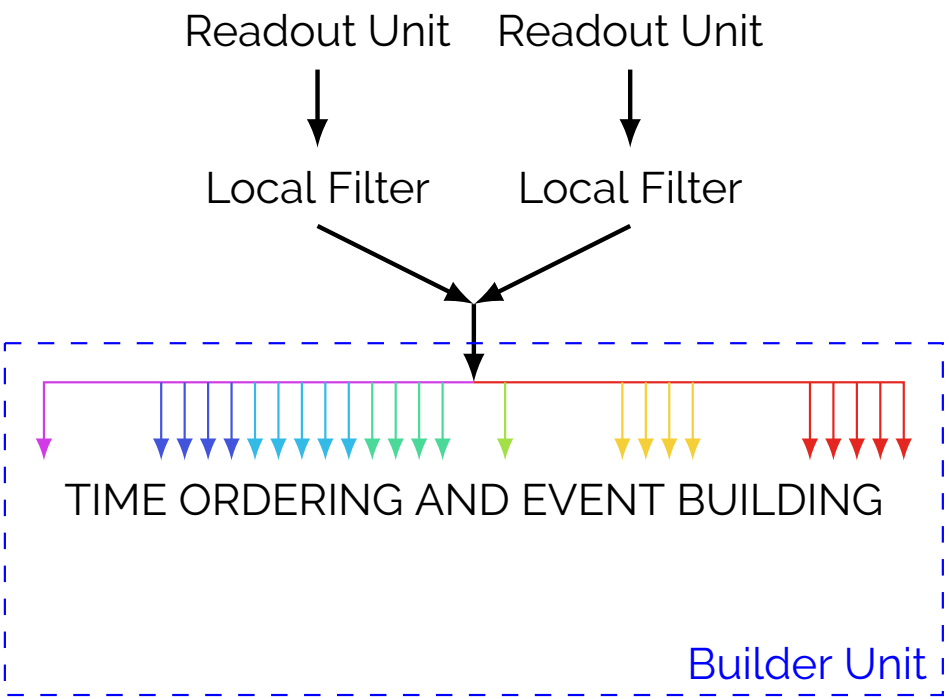
Event reconstruction based on timestamp information



- Configurable inputs

Builder units (BU, MU)

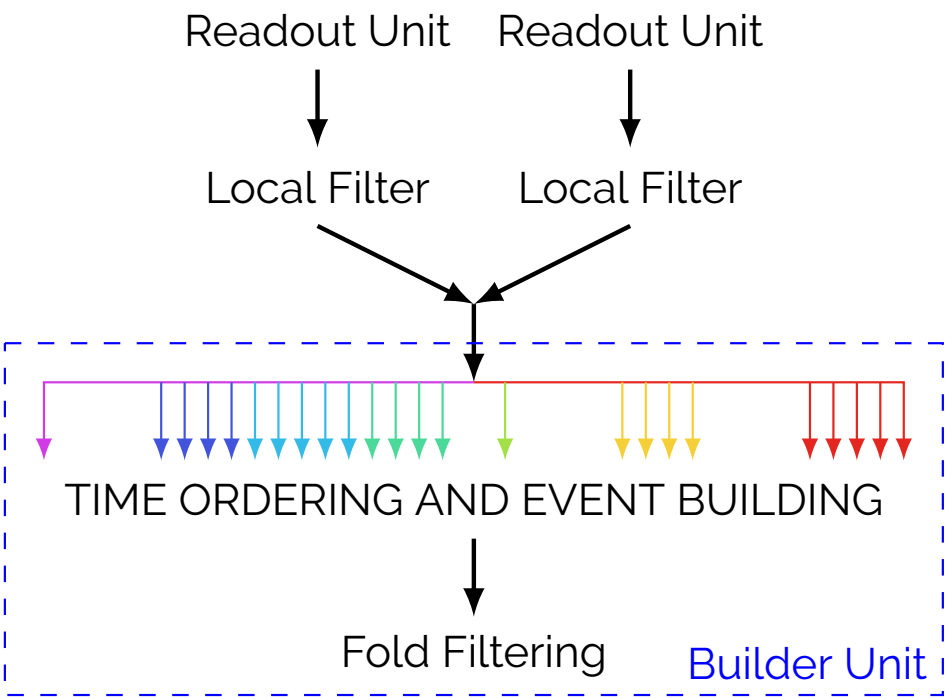
Event reconstruction based on timestamp information



- Configurable inputs
- Configurable time window via XML file

Builder units (BU, MU)

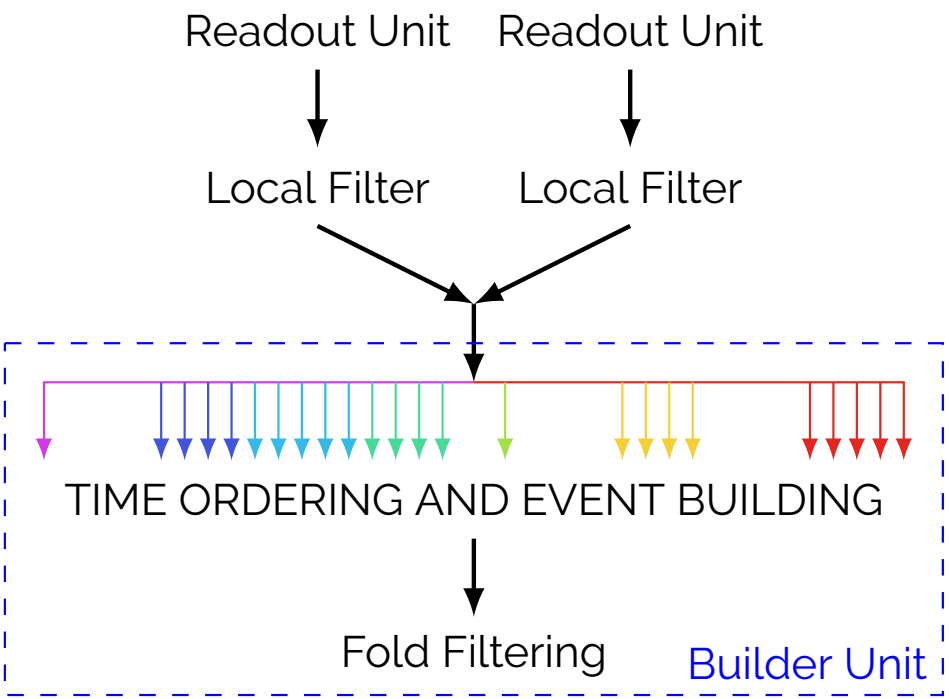
Event reconstruction based on timestamp information



- Configurable inputs
- Configurable time window via XML file

Builder units (BU, MU)

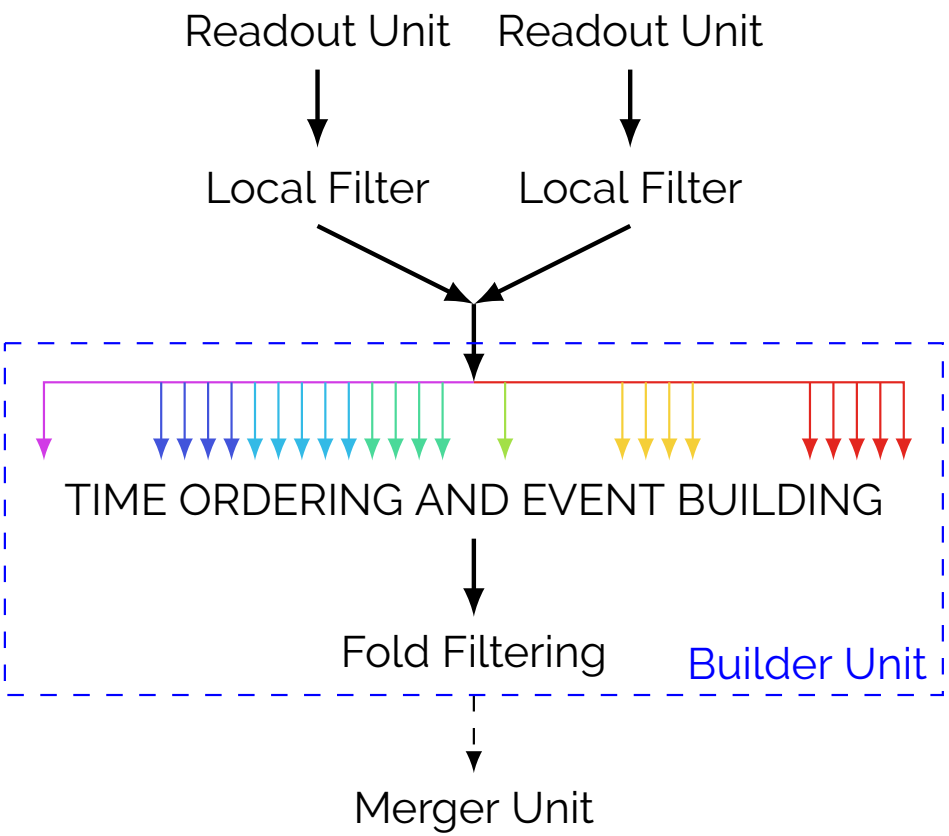
Event reconstruction based on timestamp information



- Configurable inputs
- Configurable time window via XML file
- Minimum fold requirement for the output

Builder units (BU, MU)

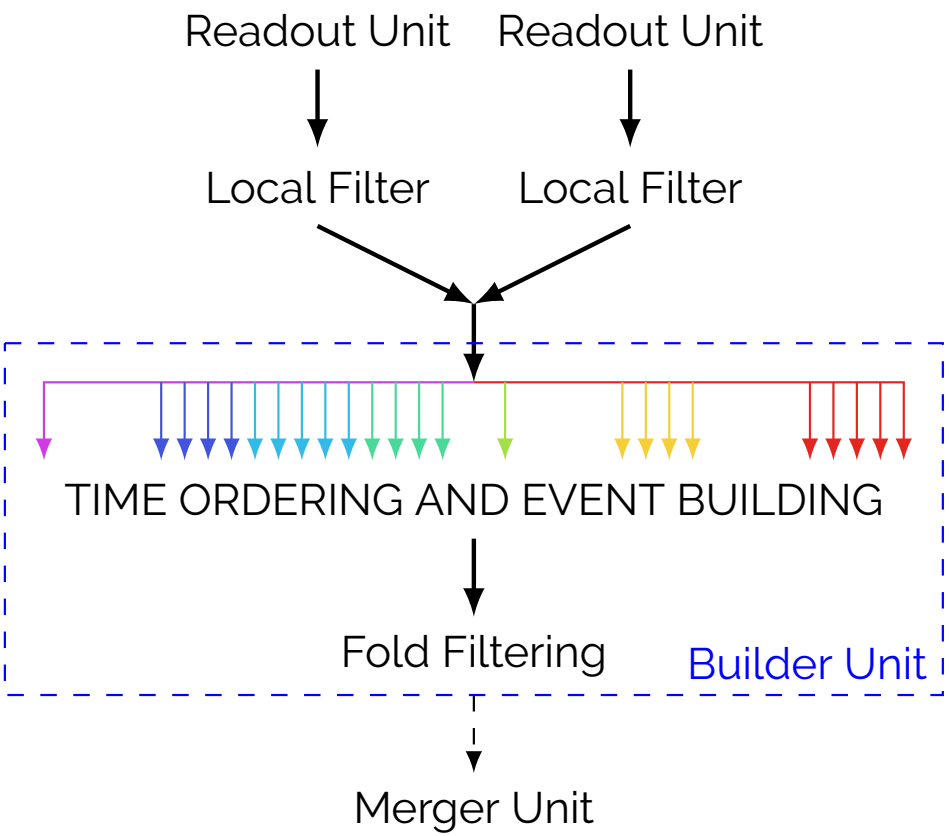
Event reconstruction based on timestamp information



- Configurable inputs
- Configurable time window via XML file
- Minimum fold requirement for the output

Builder units (BU, MU)

Event reconstruction based on timestamp information



- Configurable inputs
- Configurable time window via XML file
- Minimum fold requirement for the output
- Composite frames with configurable keyword

User Supplied Actors: The Local (LF) and Global (GF) Filters

Separation of Data Flow and Data Treatment:

- Generic application ensuring the communication between the actor and the DAQ
  $\Longrightarrow$ Common to all data filters

User Supplied Actors: The Local (LF) and Global (GF) Filters

Separation of Data Flow and Data Treatment:

- Generic application ensuring the communication between the actor and the DAQ
  $\Longrightarrow$ Common to all data filters
- Skeleton of data treatment actor in C/C++

```
/**
* User supplied function implementing the action to be executed on the
* Halted->Ready transition of the application's finite state machine.
*/
void process_config(const char* file, int *error_code);

/**
* User supplied function implementing the action to be executed on the
* Ready->Enabled transition of the application's finite state machine.
*/
void process_start(uint32_t run, int *error_code);

/**
* User supplied function implementing the action to be executed on the
* Enabled->Ready transition of the application's finite state machine.
*/
void process_stop(int *error_code);

/**
* User supplied function that is invoked on the filter application when
* a buffer is received.
*/
void process_block(
  void *input_buffer, int input_size, int packet_ID,
  void *output_buffer, int output_size,
  int *used_size_of_output_buffer,
  int *error_code);
```

User Supplied Actors: The Local (LF) and Global (GF) Filters

Separation of Data Flow and Data Treatment:

- Generic application ensuring the communication between the actor and the DAQ
  $\Longrightarrow$ Common to all data filters
- Skeleton of data treatment actor in C/C++
- For GALILEO:
  - Compton-Suppression
  - CFD
  - Energy calculation using the Short Traces
  - Zero suppression
  - Data formatting: ADF-like key, ...

```
/**
* User supplied function implementing the action to be executed on the
* Halted->Ready transition of the application's finite state machine.
*/
void process_config(const char* file, int *error_code);

/**
* User supplied function implementing the action to be executed on the
* Ready->Enabled transition of the application's finite state machine.
*/
void process_start(uint32_t run, int *error_code);

/**
* User supplied function implementing the action to be executed on the
* Enabled->Ready transition of the application's finite state machine.
*/
void process_stop(int *error_code);

/**
* User supplied function that is invoked on the filter application when
* a buffer is received.
*/
void process_block(
  void *input_buffer, int input_size, int packet_ID,
  void *output_buffer, int output_size,
  int *used_size_of_output_buffer,
  int *error_code);
```

User Supplied Actors: The Local (LF) and Global (GF) Filters

Separation of Data Flow and Data Treatment:

- Generic application ensuring the communication between the actor and the DAQ
  $\Longrightarrow$ Common to all data filters
- Skeleton of data treatment actor in C/C++
- For GALILEO:
  - Compton-Suppression
  - CFD
  - Energy calculation using the Short Traces
  - Zero suppression
  - Data formatting: ADF-like key, ...
  - "Software trigger"

```
/**
* User supplied function implementing the action to be executed on the
* Halted->Ready transition of the application's finite state machine.
*/
void process_config(const char* file, int *error_code);

/**
* User supplied function implementing the action to be executed on the
* Ready->Enabled transition of the application's finite state machine.
*/
void process_start(uint32_t run, int *error_code);

/**
* User supplied function implementing the action to be executed on the
* Enabled->Ready transition of the application's finite state machine.
*/
void process_stop(int *error_code);

/**
* User supplied function that is invoked on the filter application when
* a buffer is received.
*/
void process_block(
  void *input_buffer, int input_size, int packet_ID,
  void *output_buffer, int output_size,
  int *used_size_of_output_buffer,
  int *error_code);
```

Slow control & Readout for the CAEN digitizers

Still on-going development:
$\Longrightarrow$ Coupling of GALILEO with $LaBr_3$ detectors (High energy or Fast timing)
$\Longrightarrow$ Detectors with fast signals $\rightarrow$ Sampling from 100 to 500 Msps

Slow Control

- SOAP server on PCIe/USB host
- SOAP client on remote PC
- **DPP-PSD** firmware registers
- Support USB and Optical Fiber

Slow control & Readout for the CAEN digitizers

Still on-going development:
$\Longrightarrow$ Coupling of GALILEO with $LaBr_3$ detectors (High energy or Fast timing)
$\Longrightarrow$ Detectors with fast signals $\rightarrow$ Sampling from 100 to 500 Msps

Slow Control

- SOAP server on PCIe/USB host
- SOAP client on remote PC
- **DPP-PSD** firmware registers
- Support USB and Optical Fiber

Slow control & Readout for the CAEN digitizers

Still on-going development:
$\Longrightarrow$ Coupling of GALILEO with $LaBr_3$ detectors (High energy or Fast timing)
$\Longrightarrow$ Detectors with fast signals $\rightarrow$ Sampling from 100 to 500 Msps

Slow Control

- SOAP server on PCIe/USB host
- SOAP client on remote PC
- **DPP-PSD** firmware registers
- Support USB and Optical Fiber

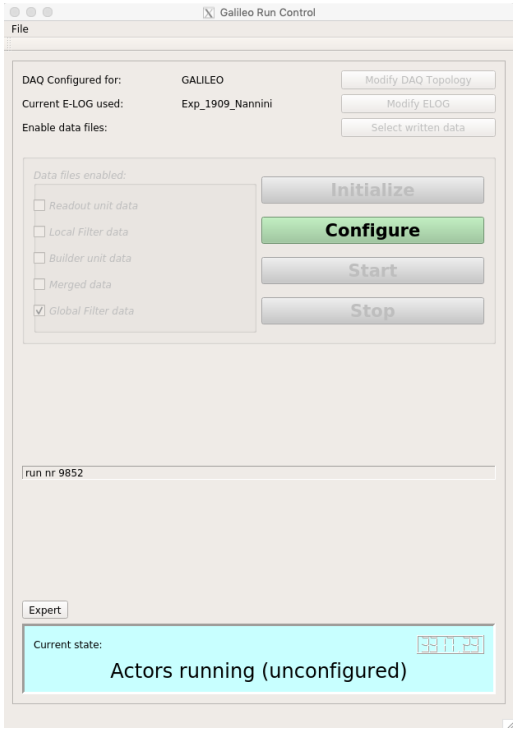
Readout

- Architecture for all available firmwares
- Preliminary test for **DPP-PSD**
- Support of list-mode (LM)
- Support of mixed-mode (MM)
- Preliminary test with Fast-Timing detectors
  $\rightarrow \sim$ 230 ps resolution
- PLL at 100 MHz available
- On-going coupling with the GTS

Run Control

Based originally on the run control developed by J. Grebosz

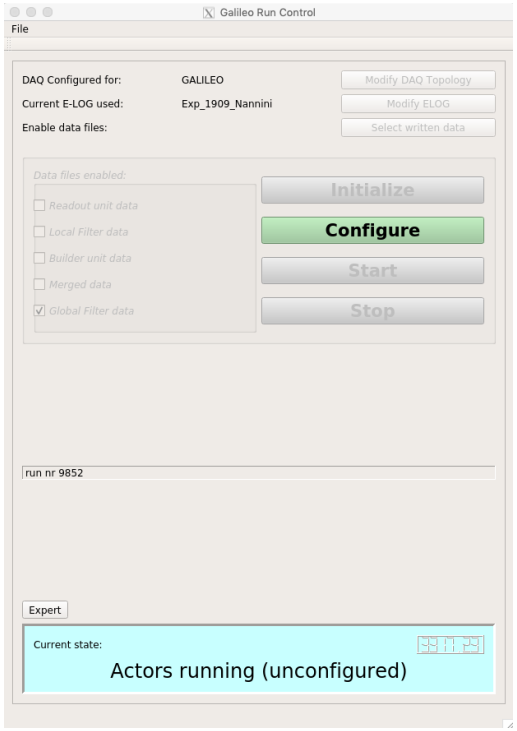
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

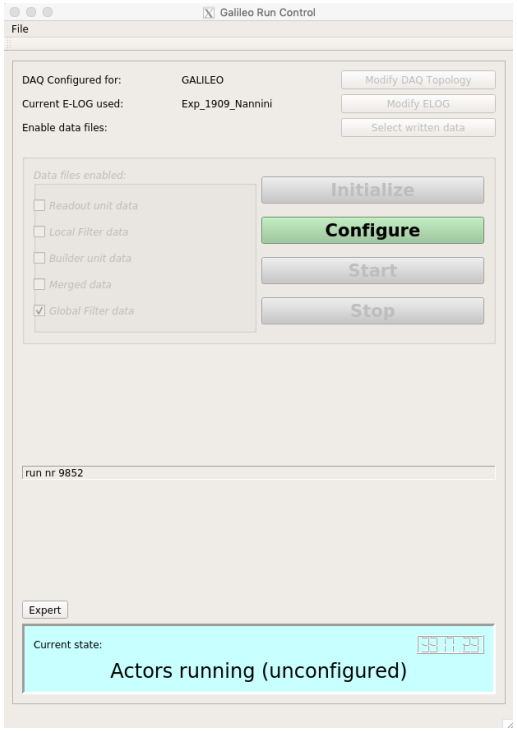
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

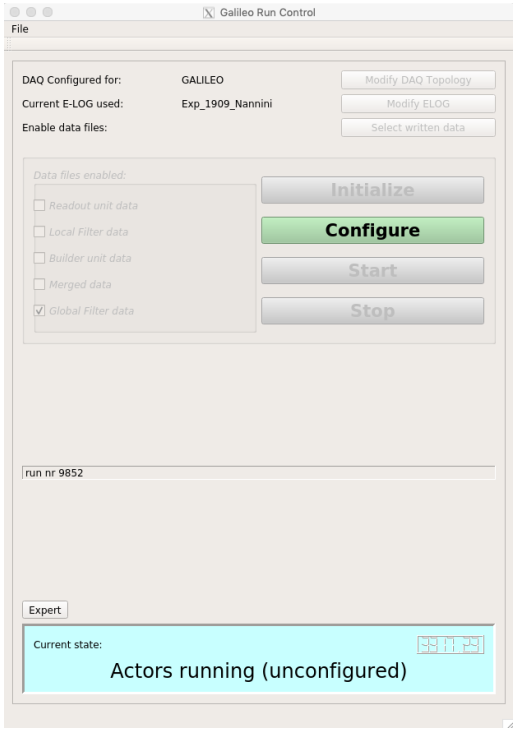
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

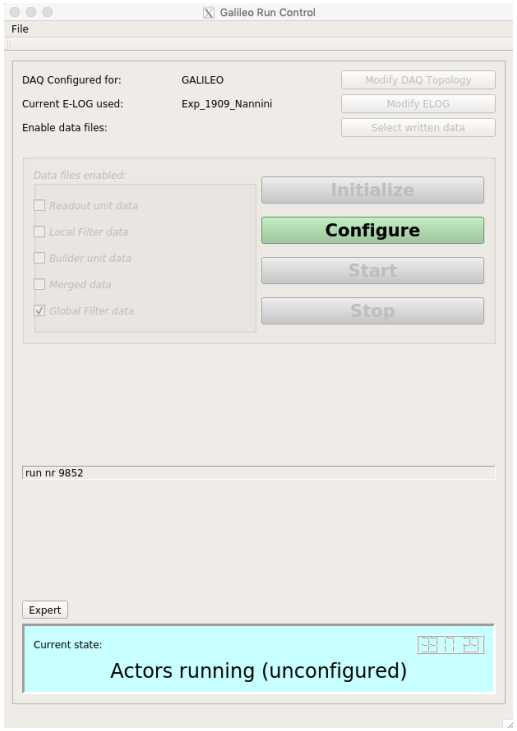
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors
- Direct communication with the XDAQ actors using SOAP messages



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

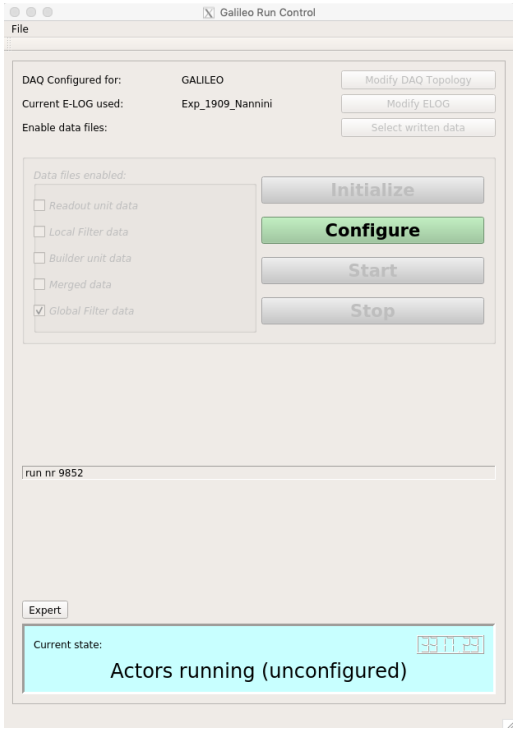
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors
- Direct communication with the XDAQ actors using SOAP messages
- Selection of files written on disk



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

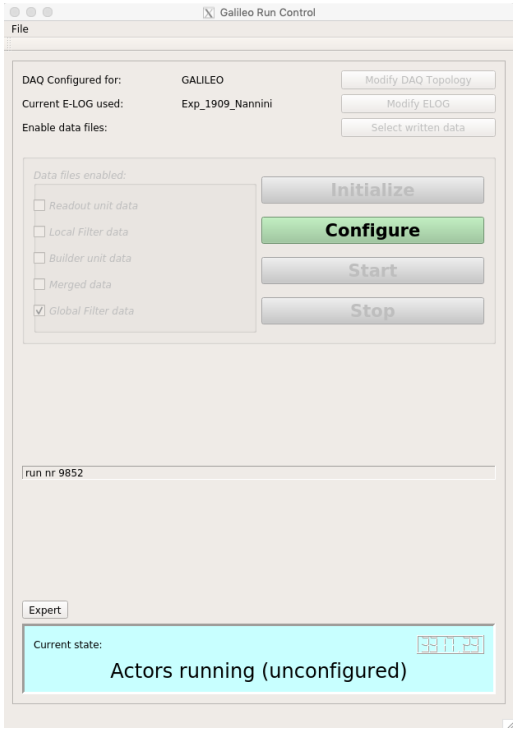
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors
- Direct communication with the XDAQ actors using SOAP messages
- Selection of files written on disk
- E-LOG capable $\Longrightarrow$ automatic post at Start/Stop



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

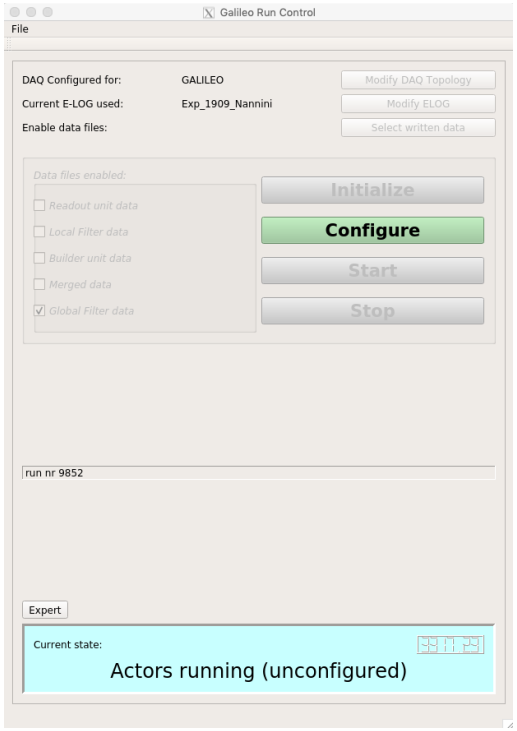
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors
- Direct communication with the XDAQ actors using SOAP messages
- Selection of files written on disk
- E-LOG capable $\Longrightarrow$ automatic post at Start/Stop
- Rate Monitoring



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

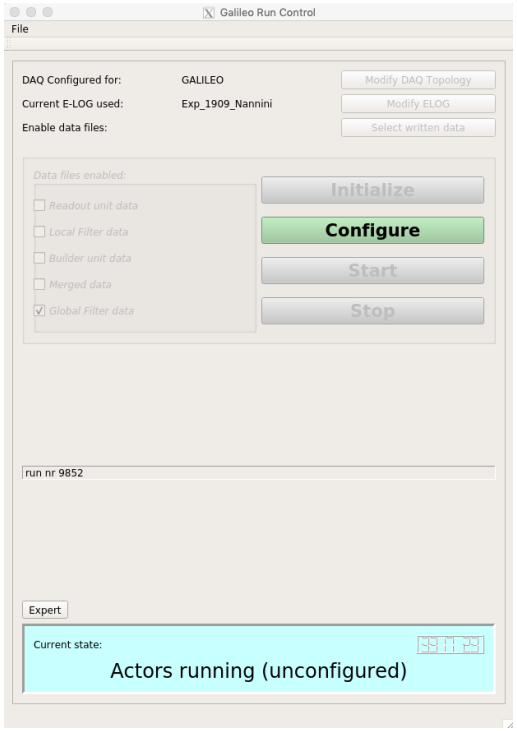
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors
- Direct communication with the XDAQ actors using SOAP messages
- Selection of files written on disk
- E-LOG capable $\Longrightarrow$ automatic post at Start/Stop
- Rate Monitoring
- ...



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

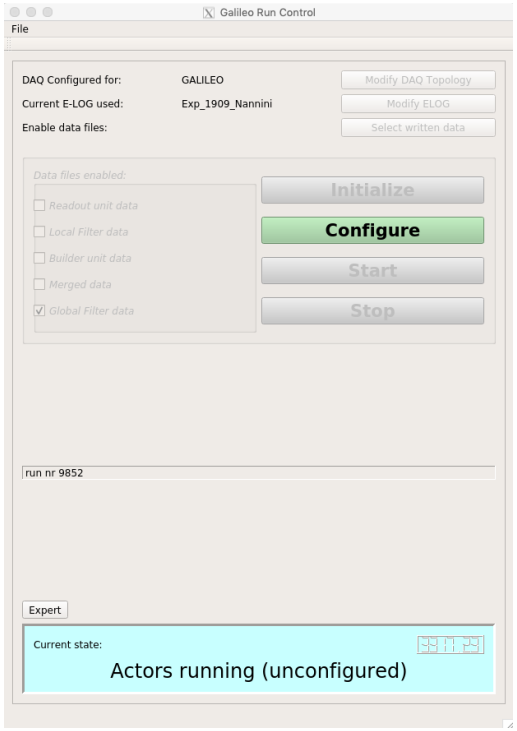
- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors
- Direct communication with the XDAQ actors using SOAP messages
- Selection of files written on disk
- E-LOG capable $\Longrightarrow$ automatic post at Start/Stop
- Rate Monitoring
- ...
- Always willing to improve it



Run Control

Based originally on the run control developed by J. Grebosz **relying on bash scripts :(**

- Simple graphical interface with **limited** possible actions
- Developed in C++ / Qt5
- Reads XML file to determine the list of actors
- Direct communication with the XDAQ actors using SOAP messages
- Selection of files written on disk
- E-LOG capable $\Longrightarrow$ automatic post at Start/Stop
- Rate Monitoring $\Longrightarrow$ Interface Grafana
- ...
- Always willing to improve it



Conclusion

- All ancillaries presented (except NEDA at the moment) are integrated in XDAQ
- Local know-how and support

Conclusion

- All ancillaries presented (except NEDA at the moment) are integrated in XDAQ
- Local know-how and support
- On-going development for new ancillaries and possible PRISMA readout upgrade

Conclusion

- All ancillaries presented (except NEDA at the moment) are integrated in XDAQ
- Local know-how and support
- On-going development for new ancillaries and possible PRISMA readout upgrade
- Monitoring tools and run control are existing ... a lot of cosmetic to be done

Conclusion

- All ancillaries presented (except NEDA at the moment) are integrated in XDAQ
- Local know-how and support
- On-going development for new ancillaries and possible PRISMA readout upgrade
- Monitoring tools and run control are existing ... a lot of cosmetic to be done

Personal opinions:

- Unique framework for all the ancillaries
- One dedicated network for the ancillaries with 1 bridge with AGATA DAQ.

Conclusion

- All ancillaries presented (except NEDA at the moment) are integrated in XDAQ
- Local know-how and support
- On-going development for new ancillaries and possible PRISMA readout upgrade
- Monitoring tools and run control are existing ... a lot of cosmetic to be done

THANK YOU FOR YOUR ATTENTION

Personal opinions:

- Unique framework for all the ancillaries
- One dedicated network for the ancillaries with 1 bridge with AGATA DAQ.