DPS Data Model

R.I. INAF IASF Bologna n. xx, 2014

Custodian:	Name:	Andrea Zoli	Signature:		Date:	06/05/2014
Prepared by:	Name:	Andrea Zoli	Signature:		Date:	06/05/2014
Reviewed by:	Name:	Andrea Bulgarelli	Signature:		Date:	06/05/2014
Approved by:	Name:	Massimo Trifoglio	Signature:		Date:	06/05/2014

AUTHOR LIST	
Andrea Zoli, Andrea Bulgarelli	INAF/IASF Bologna, Italy

DISTRIBUTION LIST	
CIWS e-mail list	ciws@iasfbo.inaf.it

DOCUMENT HISTORY		
Version	**Date**	**Modification**
1.0	06 May 2014	First version

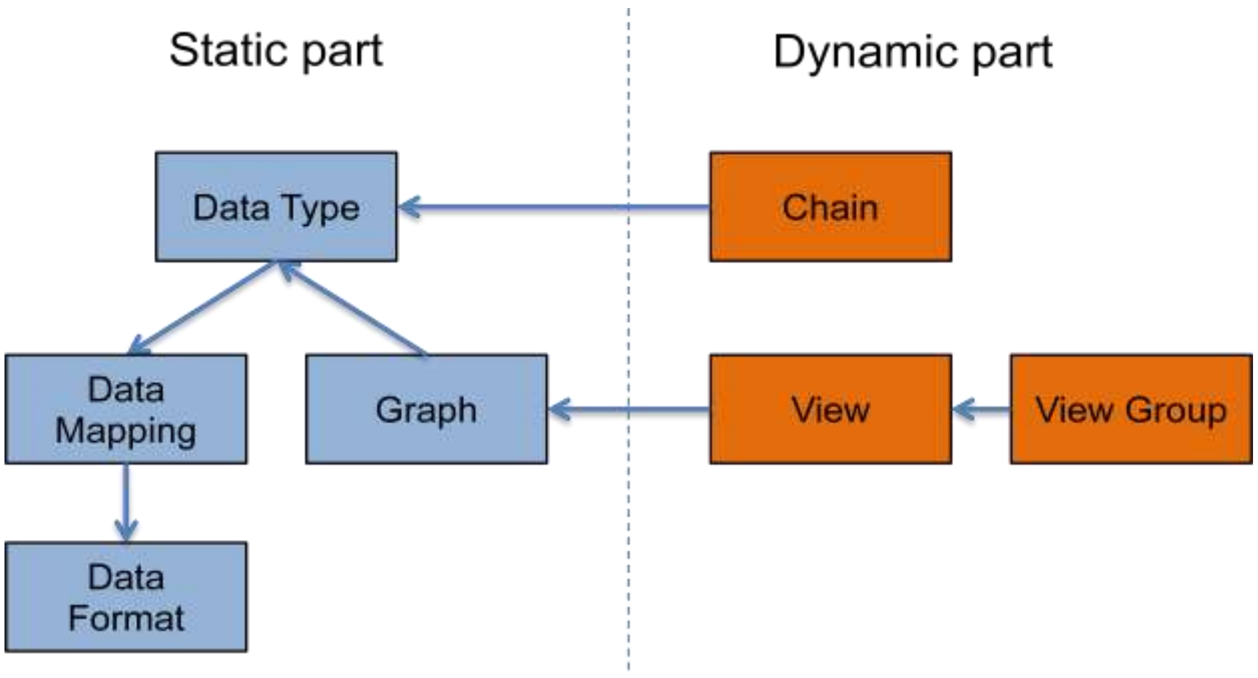
TABLE OF CONTENTS

1. Introduction

This document gives you a description of the DPS Data Model necessary to configure most of the Quick-Look component using a series of XML configuration files. It gives also an overview of the capabilities and the flexibility of the Quick-Look component. Finally, it shows you some examples of the AGILE data model porting to the CIWS-FW DPS data model. The AGILE port project it is called IW2, while the IW1 refers to the ASTRI porting that we are not going to show you with this document. We shows you the IW2 because it touch all the aspects of the DPS Data Model, so it is a complete example.

2. Data Model definition

The Data Model used by the CIWS-FW framework is defined using the XML Schema format. There are different files that are conceptually linked together allowing the CIWS components to operate from data acquisition to data visualization. It is straightforward that you need to model the domain using XML as instances of this Data Model.



As shown in the image above, the Data Model could be divided conceptually into two groups:

- the **static part** of the model is generally defined by the CIWS-FW developer, because it requires knowledge of the framework internal operations, and programming skills to extend the framework, in order to support additional data formats. However, in the simpler scenario even a CIWS-FW User could do the entire job, just using built-in data formats.
- the **runtime part** of the model could be changed during the execution of the pipeline and it is generally updated by the CIWS-FW User through a GUI.

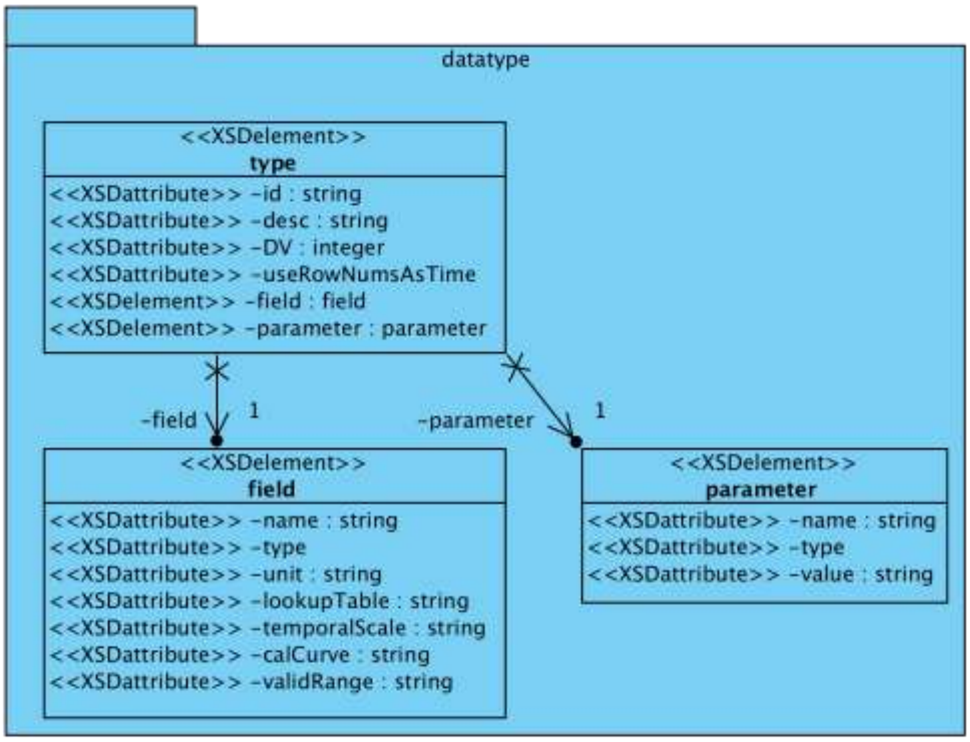
Starting from the static part the model is divided into:

- a **Data Type** that defines a generalization of a data structure. Because it represents an abstraction it is named Type and has a fixed number of fields. However, you can easily see it as a table with a fixed number of columns.
- a **Data Format** that describes how a specific file format it is structured. The only built-in format supported by the framework is the FITS file format. It can be extended easily by the CIWS-FW developers.
- a **Data Mapping** that defines how to map the data type to a specific data format.
- **Graphs** describes how to visualize data types during the Quick Look, for example you can bind a specific data type to an histogram 2d or a temporal axis graph.

For the run-time part the model is divided into:

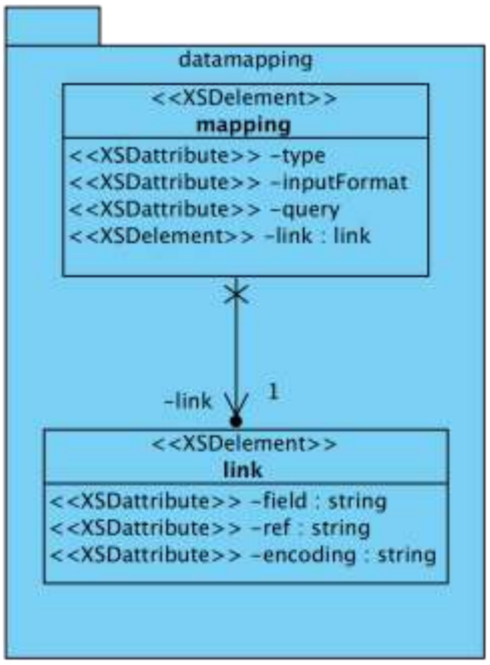
- a **Chain** that controls which and how data types are processed by the framework. This allows for example to enable/disable a specific data type without modifying its static definition.
- a **View** that defines which list of graphs are going to be drawn on a window using a specific layout.
- a **View Group** that defines a group of views. The grouping formation automate view opening avoiding the user to open individual windows.

2.1 Data Type


datatype
<<XSDelement>>
type
<<XSDattribute>> -id : string
<<XSDattribute>> -desc : string
<<XSDattribute>> -DV : integer
<<XSDattribute>> -useRowNumsAsTime
<<XSDelement>> -field : field
<<XSDelement>> -parameter : parameter
-field
1
-parameter
1
<<XSDelement>>
field
<<XSDattribute>> -name : string
<<XSDattribute>> -type
<<XSDattribute>> -unit : string
<<XSDattribute>> -lookupTable : string
<<XSDattribute>> -temporalScale : string
<<XSDattribute>> -calCurve : string
<<XSDattribute>> -validRange : string
<<XSDelement>>
parameter
<<XSDattribute>> -name : string
<<XSDattribute>> -type
<<XSDattribute>> -value : string

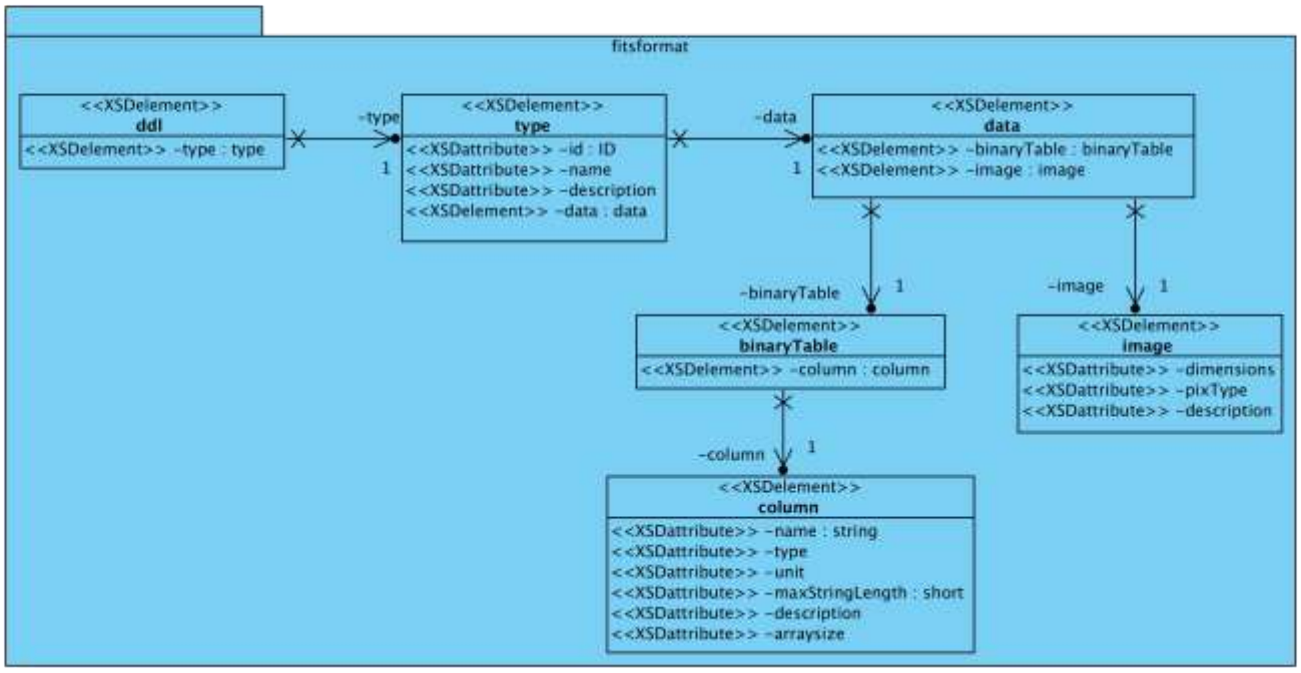
A data type has fields and parameters like DV or useRowNumAsTime. DV stands for DeltaV and identify the size of the chunk of data to be handled. The operational parameter userRowNumAsTime identify whenever use or not the row number instead of a field to handle the time. Each fields has a name, a basic type like int, float or string, a measurement unit and some additional parameters like a lookupTable, a temporal Scale, a calibration curve or a valid range of values. The CIWS-FW developer could also extend the parameter for its own purpose without modifying the data type XSD interface defining a list of parameters with generic definition name, simple type and value.

2.2 Data Mapping


datamapping
<<XSDelement>>
mapping
<<XSDattribute>> -type
<<XSDattribute>> -inputFormat
<<XSDattribute>> -query
<<XSDelement>> -link : link
-link
1
<<XSDelement>>
link
<<XSDattribute>> -field : string
<<XSDattribute>> -ref : string
<<XSDattribute>> -encoding : string

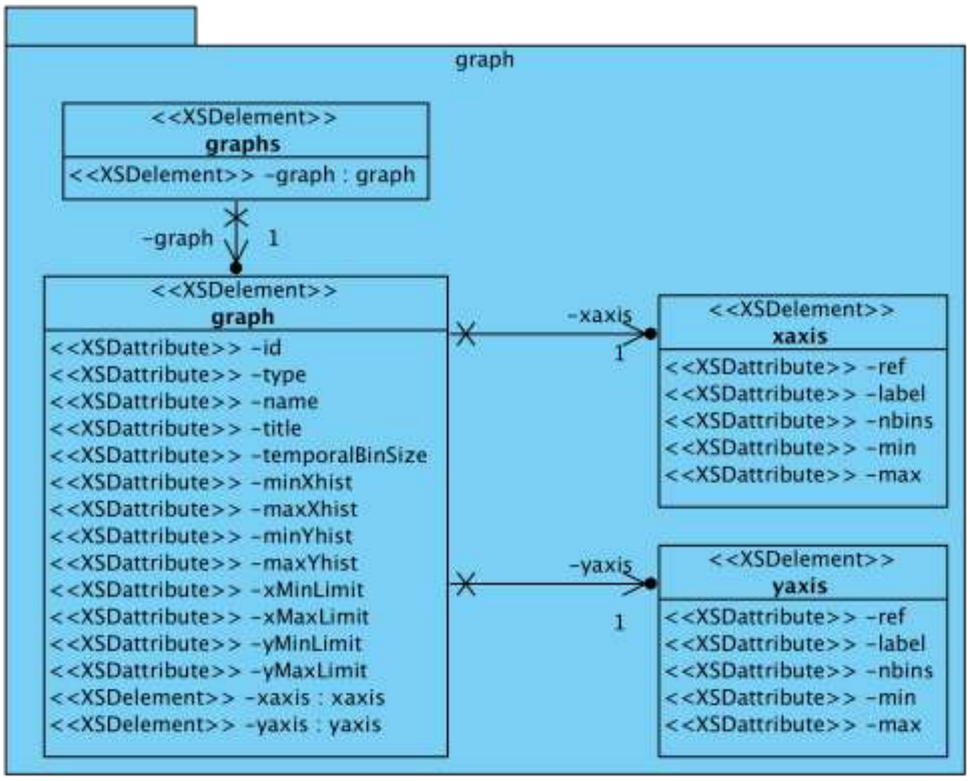
The data mapping refers to a type and to a specific inputFormat and has multiple links. The query attribute is useful to query the data format like a database to do things like innerjoin between tables. The links are mandatory in order to bind a data type to a data format. Hereafter we use the name **input data type** to define this type of data types. If there is no binding it is assumed that the data type will be generated using builders and we call this kind of data type **intermediate data type**. The ref attribute is the key attribute of a field and has a fixed grammar that is handled during the Quick-Look loading of an input data type. For FITS we used simply a “TABLE/COLUMN” string that map the field to a specific column.

2.3 Data Format (FITS)


fitsformat
<<XSDelement>>
ddl
<<XSDelement>> -type : type
-type
1
<<XSDelement>>
type
<<XSDattribute>> -id : ID
<<XSDattribute>> -name
<<XSDattribute>> -description
<<XSDelement>> -data : data
-data
1
<<XSDelement>>
data
<<XSDelement>> -binaryTable : binaryTable
<<XSDelement>> -image : image
-binaryTable
1
-image
1
<<XSDelement>>
binaryTable
<<XSDelement>> -column : column
<<XSDelement>>
image
<<XSDattribute>> -dimensions
<<XSDattribute>> -pixType
<<XSDattribute>> -description
-column
1
<<XSDelement>>
column
<<XSDattribute>> -name : string
<<XSDattribute>> -type
<<XSDattribute>> -unit
<<XSDattribute>> -maxStringLength : short
<<XSDattribute>> -description
<<XSDattribute>> -arraysize

The FITS data format is similar to the DAS ddl. The DPS subsystem needs only to handle data, by now, and doesn't load FITS metadata. In order to map to a data type we need an identifier for each data set. This design allows to treat the FITS like a database and develop queries on it.

2.4 Graphs



The Graph file define a list of graphs. Each graph binds a graph type provided with the Quick-Look component and the fields of the data type to be used as input. The graph types provided are:

- Temporal graph
- Histograms 1d, 2d and 3d
- Label-like graph for sinoptic values
- Temporal graph with error bars
- Image 2d

The graph, xaxis and yaxis attributes allows to set the graphs parameters based on the graph type. The xaxis and yaxis ref attribute, in order to link graphs axis to the fields of the data type, uses a linking grammar similar to the mapping between data type and data format: “TYPE/FIELD”.

2.5 Chains, views and view groups

The Quick-Look component use configurations files to select input data, set the layout and group the views.

More in depth the chain component allows to enable/disable data types and to configure them using a xml config file. The DK, DT and DS xml attributes are used to specify respectively the refresh rate (ms), the step (number of rows/events) and the time step (ms). The chain procChannel value has to be set in order to run the Quick-Look component in online-mode, more precisely it represents the shared memory channel to be used for reading the data in realtime. Prior to that the data is generally written in the shared memory by a Processor. The inputDataInstance value, finally, allows to use a custom input data class to read a the data type with no extension to the data model definition.

The view configuration files allow to layout graphs on a windows in a grid fashion like 2x2 or 3x4, etc. It allows also to override graphic type defined in the graph files.

The viewgroup configuration, instead, permit grouping of views. For views and viewgroups we still use a Cern ROOT config file.

3. Examples

We have provided two Instrument Workstation together with the CIWS-FW so you can see the entire data model. In this section we focus on the static part of the data model showing you how we have converted part of the AGILE data model into the CIWS-FW one, in order to process and visualize FITS files with the Quick Look component.

3.1 AGILE 3914_0 data model

We started modeling the data format (FITS) from the XML Schema of the DAS component. We used the *fits2xml* converter to generate the XML from and an existing 3914 FITS file. You can find the converter in the libQLBase component, within the scripts directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<ddl>
  <type id="3014_packets" name="PACKETS">
    <data>
      <binaryTable>
        <column name="PAKTNUMB" type="int32" />
        ...
        <column name="MMPAGEID" type="int16" />
        <column name="MMPKTCNT" type="int16" />
        <column name="APIDSEQC" type="byte" />
        <column name="OBTMSETT" type="int32" unit="us" />
        <column name="OBTSECTT" type="float64" unit="s" />
        <column name="OBTTIMFL" type="byte" />
        <column name="PLOPMODE" type="byte" />
        <column name="PKTSTRID" type="byte" />
        <column name="REDUNDID" type="byte" />
        <column name="CRCVALUE" type="int16" />
        <column name="TIME" type="float64" unit="us" />
        <column name="NEVENTS" type="int16" unit="event" />
      </binaryTable>
    </data>
  </type>
  <type id="3014_packets" name="EVENTS">
    <data>
      <binaryTable>
        <column name="PAKTNUMB" type="int32" />
        <column name="EVNUMBER" type="int16" />
        <column name="SSID" type="byte" />
        <column name="MSGVALI" type="byte" />
        <column name="TIMEDAY" type="int16" unit="d" />
        <column name="TIMEMSEC" type="int32" unit="ms" />
        <column name="Q1" type="float32" />
        <column name="Q2" type="float32" />
        <column name="Q3" type="float32" />
        <column name="Q4" type="float32" />
        <column name="TIME" type="float64" unit="ms" />
      </binaryTable>
    </data>
  </type>
</ddl>
```

The 3914 data format (3914_format.xml)

As you can see from the data format there are 2 tables inside the FITS format and there is a relation 1-to-Many between them, where PAKTNUMB is the foreign key.

We defined only a subset of the entire data format. We want to to visualize only the quaternions Q1, Q2, Q3, Q4 using the Quick Look component. We have chosen to use the same names of the FITS columns but you can name fields as you prefer, the data type is a data generalization. Note that the field types are float, this mean that there is a conversion from the original column types.

```
<?xml version="1.0" encoding="UTF-8"?>
<type id="3914_0" DV="100">
  <field name="SSID" type="float" />
  <field name="MSGVALI" type="float" />
  <field name="Q1" type="float" />
  <field name="Q2" type="float" />
  <field name="Q3" type="float" />
  <field name="Q4" type="float" />
  <field name="TIME" type="float" unit="ms" />
</type>
```

The 3914_0 data type (3914_0_type.xml)

Once we have the data format and the data type we need to create the mapping between them. The mapping file is fairly straightforward in the case of FITS data format. A table is generalized to a type and a column is generalized to a field. So, we link a field of the data type with a column using the ref. The **reference** grammar is fixed as **table/column**.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping type="3914_0" inputFormat="fits" query="select 3914_events.* from
3014_events inner join 3014_packets ON 3914_packets.PAKTNUMB ==
3914_events.PAKTNUMB">
  <link field="SSID" ref="3914_events/SSID" />
  <link field="MSGVALI" ref="3914_events/MSGVALI" />
  <link field="Q1" ref="3914_events/Q1" />
  <link field="Q2" ref="3914_events/Q2" />
  <link field="Q3" ref="3914_events/Q3" />
  <link field="Q4" ref="3914_events/Q4" />
  <link field="TIME" ref="3914_events/TIME" />
</mapping>
```

The 3914_0 data mapping (3914_0_mapping.xml)

You can note, also, that we specified a query to resolve inner join between the two tables, telling the Quick Look component to handle it.

Then we define how to visualize the data modeling the Graphs xml. We choose to visualize the quaternions and the MSGVALI fields using temporal graphs.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphs>
<!--####################################################################-->
<!-- File: PDHU.param -->
  <graph id="3914_0_MSGVALI" type="temporal" name="3914_0:MSGVALI"
title="3914_0: MSGVALI">
    <xaxis ref="3914_0/MSGVALI" label="VALIDITY" />
  </graph>
  <graph id="3914_0_Q1" type="temporal" name="QUAT1" title="3914_0: QUAT1">
    <xaxis ref="3914_0/Q1" label="quat" />
  </graph>
  <graph id="3914_0_Q2" type="temporal" name="QUAT2" title="3914_0: QUAT2">
    <xaxis ref="3914_0/Q2" label="quat" />
  </graph>
  <graph id="3914_0_Q3" type="temporal" name="QUAT3" title="3914_0: QUAT3">
    <xaxis ref="3914_0/Q3" label="quat" />
  </graph>
  <graph id="3914_0_Q4" type="temporal" name="QUAT4" title="3914_0: QUAT4">
    <xaxis ref="3914_0/Q4" label="quat" />
  </graph>
  ...
</graphs>
```

The 3914_0 graphs (PDHU.xml)

Note: the Quick Look load only the graphs defined in the parameters.list file, so you need to add the PDHU.xml file to it in order to use it.

Finally, a last configuration file it's needed for the Quick Look in order to load a data type. You have to configure the chain.

```
<?xml version="1.0" encoding="UTF-8"?>
<chains>
  ...
  <chain name="3914_0" type="3914_0" uniqueKey="3914" procChannel="14">
    <config DK="6000" DT="100" DS="100" qlrows="10" connect="yes"/>
  </chain>
  ...
</chains>
```

chains.conf

Some notes for the chain file:

- the chain type refers to a data type, the chain name is used only for the visualization.
- the uniqueKey is an identifier for the data type that must be present in the FITS filename. This allow the Quick Look to associate the data within the FITS file with a specific data type.
- The procChannel refers to the DISCoS shared memory channel where to read data during online mode

The <config> element is generally updated using the Quick Look GUI. The connect="yes" means that the chain is enabled

3.2 AGILE 3903 data model

The 3903 data model is a bit more complex then the 3014. We have a single table of input. The only attribute not already seen is the arraysize one. It means that a cell of the table contains an array of size "arraysize". If not specified the default value it is 1.
The data model 3903 requires a processing of the data type in order to visualize values like error bars on the graphs. All the job of processing is done by the Quick Look component. The CIWS-FW developer needs only to extend a Builder class and define the output as a new data type. We show you how this can be done in few steps. The data format is defined as follow.

```
<?xml version="1.0" encoding="UTF-8"?>
<ddl>
  <type id="3903_packets" name="PACKETS">
    <data>
      <binaryTable>
        <column name="PAKTNUMB" type="int32" />
        <column name="PKTIDVNM" type="byte" />
        <column name="PKTIDTYP" type="byte" />
        <column name="PKTIDDFF" type="byte" />
        <column name="SEGMFLAG" type="byte" />
        <column name="PKTSEQCN" type="int16" />
        <column name="PKTLENGT" type="int16" />
        <column name="32BITSYN" type="int32" />
        <column name="CRCFLAG" type="byte" />
        <column name="MMPAGEID" type="int16" />
        <column name="MMPKTCNT" type="int16" />
        <column name="APIDSEQC" type="byte" />
        <column name="OBTMSETT" type="int32" unit="us" />
        <column name="OBTSECTT" type="float64" unit="s" />
        <column name="OBTTIMFL" type="byte" />
        <column name="PLOPMODE" type="byte" />
        <column name="PKTSTRID" type="byte" />
        <column name="REDUNDID" type="byte" />
        <column name="CRCVALUE" type="int16" />
        <column name="TIME" type="float64" unit="s" />
        <column name="PACKETID" type="int16" />
        <column name="TPKT3903" type="int16" />
        <column name="T1EVPEMS" type="int32" unit="ms" />
        <column name="T1EVPESE" type="int32" unit="s" />
        <column name="EVFTBADD" type="byte" />
        <column name="EVFEBADD" type="byte" />
        <column name="EVTA1ADD" type="byte" />
        <column name="CONSIGMA" type="float32" />
        <column name="PEDMEANS" type="float32" arraysize="128" />
        <column name="PEDSIGMA" type="float32" arraysize="128" />
        <column name="TIMEEVENT" type="float64" unit="ms" />
      </binaryTable>
    </data>
  </type>
</ddl>
```

The 3903 data format (3903_format.xml)

We want to handle only few fields of the entire data format, the data type is the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<type id="3903" desc="3903_Pedestral" DV="288">
  <field name="PEDMEANS" type="float" size="128" />
  <field name="PEDSIGMA" type="float" size="128" />
  <field name="CONSIGMA" type="float" />
  <field name="EVFTBADD" type="float" />
  <field name="EVFEBADD" type="float" />
  <field name="EVTA1ADD" type="float" />
  <field name="TIME" type="float" unit="ms" />
</type>
```

The 3903 data type (3903_type.xml)

The mapping is a simple association type/field → table/column, nothing different from the previous example.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping type="3903" inputFormat="fits">
  <link field="PEDMEANS" ref="3903_packets/PEDMEANS" size="128" />
  <link field="PEDSIGMA" ref="3903_packets/PEDSIGMA" size="128" />
  <link field="CONSIGMA" ref="3903_packets/CONSIGMA" />
  <link field="EVFTBADD" ref="3903_packets/EVFTBADD" />
  <link field="EVFEBADD" ref="3903_packets/EVFEBADD" />
  <link field="EVTA1ADD" ref="3903_packets/EVTA1ADD" />
  <link field="TIME" ref="3903_packets/TIME" />
</mapping>
```

The 3903 data mapping (3903_mapping.xml)

The chains.xml has, obviously an entry for the 3903.

```
<?xml version="1.0" encoding="UTF-8"?>
<chains>
  ...
  <chain name="3903" type="3903" uniqueKey="3903" procChannel="3">
    <config DK="5000" DT="20" DS="288" qlrows="10" connect="yes"/>
  </chain>
  ...
</chains>
```

The 3903 entry (chains.xml)

Then we create a Builder class that build the intermediate 3903_DIFF data type from our 3903 data type as follow.

```
#ifndef EIDBUILDER_3903_DIFF_H
#define EIDBUILDER_3903_DIFF_H
#include "QLExternalInputDataBuilder.h"
#include "QLInputDataParameter.h"
#include "MIOFile.h"

class Data_3903 {
public:
    Data_3903() {
        p3903 = 0;
    }
    void SetData(QLInputDataParameter* p3903) {
        this->p3903 = p3903;
    }
    DOUBLE_T GetMean(unsigned short ftb, unsigned short feb, unsigned short chip, unsigned
short strip);
    DOUBLE_T GetSigma(unsigned short ftb, unsigned short feb, unsigned short chip, unsigned
short strip);

protected:
    QLInputDataParameter* p3903;
    int SearchIndex(int ftb, int feb, int chip);
};

class EIDBuilder_3903_DIFF : public QLExternalInputDataBuilder {
public:
    EIDBuilder_3903_DIFF(TString inName, TString outName);

    virtual ~EIDBuilder_3903_DIFF();

    virtual QLInputData* Build(QLInputData* in);

protected:

    QLInputDataParameter* c3903;
    Bool_t dataLoaded;
    TString filenamePed;
    MIOFile fileLog;
    MIOFile fileLogCheck;
    Bool_t firstTime;
    Data_3903* d3903;
};

#endif
```

EIDBuilder_3903_DIFF.h header file

```
#include "EIDBuilder_3903_DIFF.h"
#include "InputData_GRID.h"
#include "QLDataList.h"
#include "QLInputFileFITS.h"
#include "PDHUGlobalMemory.h"
#include "QLGlobalMemory.h"
#include "QLMain.h"

DOUBLE_T Data_3903::GetMean(unsigned short ftb, unsigned short feb, unsigned short chip,
unsigned short strip) {
    if(p3903 == 0)
        return -1;
    if(ftb > 2 || feb > 6 || chip > 24 || strip > 128)
        return -1;
    int index;
    index = SearchIndex(ftb, feb, chip);
    if(index != -1) {
        TObjArray* ar = p3903->GetDataArray("PEDMEANS");
        QLInputDataParameterGraphData* dg;
        dg = (QLInputDataParameterGraphData*) ar->At(index);
        return dg->fY[strip];
    } else
        return -1;

}

DOUBLE_T Data_3903::GetSigma(unsigned short ftb, unsigned short feb, unsigned short chip,
unsigned short strip) {
    if(p3903 == 0)
        return -1;
    if(ftb > 2 || feb > 6 || chip > 24 || strip > 128)
        return -1;
    int index;
    index = SearchIndex(ftb, feb, chip);
    if(index != -1) {
        TObjArray* ar = p3903->GetDataArray("PEDSIGMA");
        QLInputDataParameterGraphData* dg;
        dg = (QLInputDataParameterGraphData*) ar->At(index);
        return dg->fY[strip];
    } else
        return -1;

}

int Data_3903::SearchIndex(int ftb, int feb, int chip) {
    int ne = p3903->GetTotalNumberOfEvents();
    QLTARRAYD* evftb = p3903->GetData("EVFTBADD");
    QLTARRAYD* evfeb = p3903->GetData("EVFEBADD");
    QLTARRAYD* evta1 = p3903->GetData("EVTA1ADD");
    int i;
    Bool_t found = kFALSE;
    for(i=0; i<ne; i++) {
        int eftb, efeb, eta1;
        eftb = evftb->At(i);
        efeb = evfeb->At(i);
        eta1 = evta1->At(i);
        if(eftb == ftb && efeb == feb && eta1 == chip) {
            found = kTRUE;
            break;
        }
    }
    if(found)
        return i;
    else
        return -1;
}

EIDBuilder_3903_DIFF::EIDBuilder_3903_DIFF(TString inName, TString outName)
```

```
    : QLExternalInputDataBuilder(inName, outName) {
    char tmp[256];
    c3903 = new QLInputFileFITS();
    TString fnc = gm.GetBaseDirConf();
    Int_t index = 0;
    fnc += "vconf/3903.conf";
    //cout << fnc <<endl;
    c3903->Load("3903");
    //carico gli altri parametri
    TEnv* env = new TEnv((const char*) fnc);
    sprintf(tmp, ".IDP.%d.FileSource", index);
    filenamePed = gm.GetBaseDirConf();
    filenamePed += "vconf/";
    filenamePed += env->GetValue(tmp, "No filename");
//cout << "Pedestal reference file name: " << filenamePed << endl;
    sprintf(tmp, ".IDP.%d.MaxMean", index);
    pdhugm.pedMeans = (Double_t) env->GetValue(tmp, 2);
    sprintf(tmp, ".IDP.%d.MaxSigma", index);
    pdhugm.pedSigma = (Double_t) env->GetValue(tmp, 2);
    sprintf(tmp, ".IDP.%d.MaxMeanCheck", index);
    pdhugm.pedMeansCheck = (Double_t) env->GetValue(tmp, 2);
    sprintf(tmp, ".IDP.%d.MaxSigmaCheck", index);
    pdhugm.pedSigmaCheck = (Double_t) env->GetValue(tmp, 2);

    dataLoaded = kFALSE;
    firstTime = kTRUE;
    //gm.dataList->AddInputData(c3903);
    d3903 = new Data_3903;
    pdhugm.data3903 = d3903;
}


EIDBuilder_3903_DIFF::~EIDBuilder_3903_DIFF()
{
    fileLog.Close();
    delete c3903;

}


QLInputData* EIDBuilder_3903_DIFF::Build(QLInputData* in) {
    //0) verifico se il dato è di questo EID
    if(this->in != in)
        return 0;
    SyncIO(in);
    //si settano i pedestal con cui effettuare i calcoli (gli ultimi arrivati)
    d3903->SetData((QLInputDataParameter*) in);
    //0 bis) carica il pedestal da confrontare
    if(dataLoaded == kFALSE) {
        if(c3903->OpenInputSource(filenamePed)) {
            c3903->Read(1, -1, 0);
            c3903->CloseInputSource();
            dataLoaded = kTRUE;
        } else {
            gm.PrintLogMessage("It is impossible to open the pedestal reference file.");
            dataLoaded = kFALSE;
            return 0;
        }
    }

    //1) casting per passare ai tipi corretti
    QLInputDataParameter* ind = (QLInputDataParameter*) in;
    QLInputDataParameter* outd = (QLInputDataParameter*) out;

    if(ind->GetResetFlag()) {
        fileLog.Close();
        firstTime = kTRUE;
        TString filename = gm.GetOutputLogFileName();
```

```
            TString filenameCheck = gm.GetOutputLogFileName();
            filename += ".pedestal.compare";
            filenameCheck += ".pedestal.check";
            fileLog.Open(filename, "w");
            fileLogCheck.Open(filenameCheck, "w");
            cout << "Log pedestal file created in " << filename << endl;
            pdhugm.logPedestalFileName = filename;
            pdhugm.logPedestalFileNameCheck = filenameCheck;
        } else {
            fileLog.Open(pdhugm.logPedestalFileName, "a");
            fileLogCheck.Open(pdhugm.logPedestalFileNameCheck, "a");
        }

        //2) prendo i valori per la scansione del dato in input
        Long_t in_nte = ind->GetTotalNumberOfEvents();
        if(last_in_nte == in_nte)
            return out;
        else
            last_in_nte = in_nte;

        Long_t in_dimAddedData = ind->GetDimAddedData(); //il numero di dati aggiunti dall'input
    = il numero di eventi
        Long_t in_dimOldData = ind->GetDimOldData();
        Double_t* in_fTime = ind->fTime.GetArray();
        Long_t out_dimOldData = outd->GetDimOldData();
        Long_t out_nte = outd->GetTotalNumberOfEvents();
        Long_t rindex;

        //3) costruzione dell'array del tempo
        //3.1) primo giro per determinare quante righe nuove
        Long_t dimAddedData = 0;
        for(Long_t i = in_dimOldData; i < in_nte; i++) {
            dimAddedData++;
        }
        //3.2) creazione della memoria e assegnazione dei valori del tempo
        Double_t* lTime = new Double_t[dimAddedData];
        rindex = 0;
        for(Long_t i = in_dimOldData; i < in_nte; i++) {
            lTime[rindex] = in_fTime[i];
            rindex++;
        }
        outd->fTime.Add(dimAddedData, lTime);

        //4) costruzione degli array dei dati
        //4.1) alloco la memoria necessaria per le colonne
        UInt_t dimfData = outd->GetDataDimension();
        for(UInt_t col = 0; col < dimfData; col++) {
            DOUBLE_T* lCol = new DOUBLE_T[dimAddedData];
            outd->fData[col].Add(dimAddedData, lCol);
            delete[] lCol;
        }
        //4.2) alloco la memoria per le colonne di tipo fDataArray
        UInt_t dimfDataArray = outd->GetDataArrayDimension();
        for(UInt_t col = 0; col < dimfDataArray; col++) {
            for(UInt_t row = 0; row < dimAddedData; row++) {
                QLInputDataParameterGraphData* dataGraph = new QLInputDataParameterGraphData;
                dataGraph->fX.resize(0);
                dataGraph->fY.resize(128);
                for(UInt_t ix = 0; ix<128; ix++) dataGraph->fY[ix] = 0;
                dataGraph->nelements = 128;
                outd->fDataArray[col].Add(dataGraph);
            }
        }

        //4.2) assegno i valori alle colonne
        rindex = out_nte;
        QLTARRAYD* evftb_o = outd->GetData("EVFTBADD");
        QLTARRAYD* evfeb_o = outd->GetData("EVFEBADD");
        QLTARRAYD* evta1_o = outd->GetData("EVTA1ADD");
        QLTARRAYD* eccsi_o = outd->GetData("CONSIGMA");
        QLTARRAYD* resul_o = outd->GetData("RESULT"); //0 OK, 1 means out of range, 2 sigma oor,
    3 mean & sigma oor
        TObjArray* mean_o = outd->GetDataArray("PEDMEANSD");
        TObjArray* sigma_o = outd->GetDataArray("PEDSIGMAD");
        QLTARRAYD* rn = outd->GetData("ROWCOUNTER");
        TObjArray* mean2_o = outd->GetDataArray("PEDMEANS");
        TObjArray* sigma2_o = outd->GetDataArray("PEDSIGMA");
        TObjArray* pedanalvar_o = outd->GetDataArray("PEDANALVAR");
        QLTARRAYD* sev = outd->GetData("QLSEVNUMBER");

        QLTARRAYD* evftb_c = c3903->GetData("EVFTBADD");
        QLTARRAYD* evfeb_c = c3903->GetData("EVFEBADD");
        QLTARRAYD* evta1_c = c3903->GetData("EVTA1ADD");
        QLTARRAYD* eccsi_c = c3903->GetData("CONSIGMA");
        TObjArray* mean_c = c3903->GetDataArray("PEDMEANS");
        TObjArray* sigma_c = c3903->GetDataArray("PEDSIGMA");

        QLTARRAYD* evftb_i = ind->GetData("EVFTBADD");
        QLTARRAYD* evfeb_i = ind->GetData("EVFEBADD");
        QLTARRAYD* evta1_i = ind->GetData("EVTA1ADD");
        QLTARRAYD* eccsi_i = ind->GetData("CONSIGMA");
        TObjArray* mean_i = ind->GetDataArray("PEDMEANS");
        TObjArray* sigma_i = ind->GetDataArray("PEDSIGMA");
        TString line;

        if(firstTime) {
            firstTime = kFALSE;
            line += "Pedestal compare (input - nominal) at ";
            line += QLGlobalMemory::GetCurrentDateTime();
            line += "\n";
            fileLog.WriteString(line);
            line = "Mean threshold ";
            line += pdhugm.pedMeans;
            line += "\n";
            fileLog.WriteString(line);
            line = "Sigma threshold ";
            line += pdhugm.pedSigma;
            line += "\n";
            fileLog.WriteString(line);

            line = "Pedestal check at ";
            line += QLGlobalMemory::GetCurrentDateTime();
            line += "\n";
            fileLogCheck.WriteString(line);
            line = "Mean for check ";
            line += pdhugm.pedMeansCheck;
            line += "\n";
            fileLog.WriteString(line);
            line = "Sigma for check ";
            line += pdhugm.pedSigmaCheck;
            line += "\n";
            fileLog.WriteString(line);

        }


        for(Long_t i = in_dimOldData; i < in_nte; i++) {
            evftb_o->AddAt(evftb_i->At(i), rindex);
            evfeb_o->AddAt(evfeb_i->At(i), rindex);
            evta1_o->AddAt(evta1_i->At(i), rindex);
            QLInputDataParameterGraphData* dg_i, *dg_o, *dg_c, *dc_o, *ds_i, *dm_i, *dana_o;
            dg_i = (QLInputDataParameterGraphData*) mean_i->At(i);
            dm_i = dg_i;
            dc_o = (QLInputDataParameterGraphData*) mean2_o->At(i);
            dg_o = (QLInputDataParameterGraphData*) mean_o->At(i);
            dg_c = (QLInputDataParameterGraphData*) mean_c->At(i);
```

```
for(int j=0; j<128; j++) {
    dc_o->fY[j] = dg_c->fY[j];
    dg_o->fY[j] = (dg_i->fY[j] - dg_c->fY[j]);
}
dg_i = (QLInputDataParameterGraphData*) sigma_i->At(i);
ds_i = dg_i;
dc_o = (QLInputDataParameterGraphData*) sigma2_o->At(i);
dg_o = (QLInputDataParameterGraphData*) sigma_o->At(i);
dg_c = (QLInputDataParameterGraphData*) sigma_c->At(i);
for(int j=0; j<128; j++) {
    dc_o->fY[j] = dg_c->fY[j];
    if(abs(dg_i->fY[j]) < 50 &&  abs(dg_c->fY[j]) < 50)
        dg_o->fY[j] = (dg_i->fY[j] - dg_c->fY[j]);
    else
        dg_o->fY[j] = 0;
}
dana_o = (QLInputDataParameterGraphData*) pedanalvar_o->At(i);
for(int j=0; j<128; j++) {
    //analisi Alberto Pellizzoni - strip noisy finder
    double sig = ds_i->fY[j];
    double mea = dm_i->fY[j];
    if(mea > 0)
        dana_o->fY[j] = sig / TMath::Sqrt(mea);
    else
        dana_o->fY[j] = 0;
}


eccsi_o->AddAt(eccsi_i->At(i) - eccsi_c->At(i), rindex);
//si calcola adesso il risultato della valutazione
dg_o = (QLInputDataParameterGraphData*) mean_o->At(i);
dg_i = (QLInputDataParameterGraphData*) mean_i->At(i);
Int_t result = 0;
for(int j=0; j<128; j++) {
    if(abs(dg_o->fY[j]) > pdhugm.pedMeans) {
        line = "MEANS: ";
        line += "FTB ";
        line += evftb_i->At(i);
        line += "\t - FEB ";
        line += evfeb_i->At(i);
        line += "\t - CHIP ";
        line += evta1_i->At(i);
        line += "\t - STRIP ";
        line += j;
        line += "\t - Difference: ";
        line += dg_o->fY[j];
        line += "\n";
        fileLog.WriteString(line);
        if(result == 0) result = 1;
        if(result == 2) result=3;
    }
    if(dg_i->fY[j] > pdhugm.pedMeansCheck) {
        line = "MEANS: ";
        line += "FTB ";
        line += evftb_i->At(i);
        line += "\t - FEB ";
        line += evfeb_i->At(i);
        line += "\t - CHIP ";
        line += evta1_i->At(i);
        line += "\t - STRIP ";
        line += j;
        line += "\t - Value: ";
        line += dg_i->fY[j];
        line += "\n";
        fileLogCheck.WriteString(line);
    }

}
```

```
            dg_o = (QLInputDataParameterGraphData*) sigma_o->At(i);
            dg_i = (QLInputDataParameterGraphData*) sigma_i->At(i);
            for(int j=0; j<128; j++) {
                if(abs(dg_o->fY[j]) > pdhugm.pedSigma) {
                    if(result == 0) result = 2;
                    if(result == 1) result = 3;
                    line = "SIGMA: ";
                    line += "FTB ";
                    line += evftb_i->At(i);
                    line += "\t - FEB ";
                    line += evfeb_i->At(i);
                    line += "\t - CHIP ";
                    line += evta1_i->At(i);
                    line += "\t - STRIP ";
                    line += j;
                    line += "\t - Difference: ";
                    line += dg_o->fY[j];
                    line += "\n";
                    fileLog.WriteString(line);
                }
                if(dg_i->fY[j] > pdhugm.pedSigmaCheck) {
                    line = "SIGMA: ";
                    line += "FTB ";
                    line += evftb_i->At(i);
                    line += "\t - FEB ";
                    line += evfeb_i->At(i);
                    line += "\t - CHIP ";
                    line += evta1_i->At(i);
                    line += "\t - STRIP ";
                    line += j;
                    line += "\t - Value: ";
                    line += dg_i->fY[j];
                    line += "\n";
                    fileLogCheck.WriteString(line);
                }

            }
            resul_o->AddAt(result, rindex);
            rindex++;
        }
        fileLog.Close();
        fileLogCheck.Close();
        //6) set delle variabili del dato di output
        outd->SetDimOldData(out_nte);
        outd->SetDimAddedData(dimAddedData);
        outd->SetTotalNumberOfEvents(out_nte + dimAddedData);
        //7) free
        delete[] lTime;
        //outd->SaveData("a.txt");
        return outd;
}
```

EIDBuilder_3903_DIFF.cpp source file

In order to process a data type you only need to extend the QLExternalInputDataBuilder class. The method Build() is essentially a processor. The output is a new data type described that you need to describe in a different xml.

The output data type we will generate with the EIDBuilder is the following.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<type id="3903_DIFF" desc="3903 Pedestral Difference" DV="10">
  <field name="EVFTBADD" type="float" />
  <field name="EVFEBADD" type="float" />
  <field name="EVTA1ADD" type="float" />
  <field name="CONSIGMA" type="float" />
  <field name="RESULT" type="float" />
  <field name="PEDMEANSD" type="float" />
  <field name="PEDSIGMAD" type="float" />
  <field name="PEDMEANS" type="float" size="128" />
  <field name="PEDSIGMA" type="float" size="128" />
  <field name="PEDANALVAR" type="float" />
</type>
```

3903_DIFF data type (3903_DIFF_type.xml)

In order to work we need, also, to define the 3903_DIFF data type as a built type in the built-types.xml file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<built-types>
  ...
  <type name="3903_DIFF" />
</built-types>
```

Built data types (built-types.xml)

Finally, here we are with the graph part. We defined the *PEDMEANS* and *PEDSIGMA* fields as data array 2d and we can easily show them using a "graph_dataarray" type. The *QLNOCOL* field is a simple event counter and the Quick Look component has a graph type for that. There are other graphs that uses the 3903_DIFF data type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<graphs>
<!--###################################################################-->
<!-- File: 3903.param -->
<graph id="3903_F_PEDMEANS_0_0" type="graph_dataarray" name="3903:
PEDMEANS_0_0" title="Pedestal Mean Value: FTB(0) FEB(0)">
  <xaxis ref="3903/PEDMEANS" label="Strip" nbins="128" min="0" max="128" />
  <yaxis label="Noise value" />
</graph>
...
<graph id="3903_F_PEDSIGMA_0_0" type="graph_dataarray" name="3903:
PEDSIGMA_0_0" title="Pedestal Sigma Value: FTB(0) FEB(0)">
  <xaxis ref="3903/PEDSIGMA" label="Strip" nbins="128" min="0" max="128" />
  <yaxis label="Noise value" />
</graph>
...
<graph id="3903_COUNT" type="eventcount" name="3903: Event count"
title="3903 Event count" temporalBinSize="1">
```

```
  <xaxis ref="3903/QLNOCOL" label="Time (steps = 1 ms)" />
  <yaxis label="Count" />
</graph>
<graph id="3903_F_PEDMERR_0_0" type="grapherror_dataarray" name="3903:
PEDMEANS_ERR_0_0" title="Pedestal Mean Value: FTB(0) FEB(0)">
  <xaxis ref="3903/PEDMEANS" label="Strip" nbins="128" min="0" max="128" />
  <yaxis errorType="3903/PEDSIGMA" label="Noise value" />
</graph>
...
<graph id="3903_PEDMEANS_HISTO" type="histo_dataarray" name="3903 PEDMEANS
Histo" title="3903 Pedestal means histo">
  <xaxis ref="3903/PEDMEANS" label="ADC count" nbins="4906" min="0"
max="4096" />
  <yaxis label="Number of strips" />
</graph>
...
<graph id="3903_COHEN_PLOT" type="graph" name="3903 Coher. noise sigma
plot" title="3903 Coher. noise sigma" temporalBinSize="1">
  <xaxis ref="3903/ROWCOUNTER" label="Row number" />
  <yaxis ref="3903/CONSIGMA" label="Sigma" />
</graph>
<graph id="3903_FTB_QV" type="histogram" name="3903 FTB address histo (QV)"
title="3903 FTB address histo (QV)">
  <xaxis ref="3903/EVFTBADD" label="FTB" nbins="2" min="0" max="2" />
  <yaxis label="Count" />
</graph>
<graph id="3903_EVCOUNT" type="synoptic_single_value" name="3903 Number of
events" title="3903 Number of events" temporalBinSize="1">
  <xaxis ref="3903/ROWCOUNTER" label="events" />
</graph>
...
</graphs>
```

The 3903 graphs (3903_graph.xml)

For completeness we show you, also, the 3903 viewgroup definition and views that are still in the old format (not xml).

```
...
.View.Group.Filename.2   3903.views
.View.Group.Name.2    Pedestal views
...
```

Part of the viewgroups.conf file

```
.Views.number: 25
.Views.0   3903_DIFF_FTB0_FEB0_MEAN.view
.Views.1   3903_DIFF_FTB0_FEB1_MEAN.view
.Views.2   3903_DIFF_FTB0_FEB2_MEAN.view
.Views.3   3903_DIFF_FTB0_FEB3_MEAN.view
.Views.4   3903_DIFF_FTB0_FEB4_MEAN.view
.Views.5   3903_DIFF_FTB0_FEB5_MEAN.view
.Views.6   3903_DIFF_FTB1_FEB0_MEAN.view
.Views.7   3903_DIFF_FTB1_FEB1_MEAN.view
.Views.8   3903_DIFF_FTB1_FEB2_MEAN.view
.Views.9   3903_DIFF_FTB1_FEB3_MEAN.view
.Views.10  3903_DIFF_FTB1_FEB4_MEAN.view
.Views.11  3903_DIFF_FTB1_FEB5_MEAN.view
.Views.12  3903_DIFF_FTB0_FEB0_SIGMA.view
.Views.13  3903_DIFF_FTB0_FEB1_SIGMA.view
.Views.14  3903_DIFF_FTB0_FEB2_SIGMA.view
.Views.15  3903_DIFF_FTB0_FEB3_SIGMA.view
.Views.16  3903_DIFF_FTB0_FEB4_SIGMA.view
.Views.17  3903_DIFF_FTB0_FEB5_SIGMA.view
.Views.18  3903_DIFF_FTB1_FEB0_SIGMA.view
.Views.19  3903_DIFF_FTB1_FEB1_SIGMA.view
.Views.20  3903_DIFF_FTB1_FEB2_SIGMA.view
.Views.21  3903_DIFF_FTB1_FEB3_SIGMA.view
.Views.22  3903_DIFF_FTB1_FEB4_SIGMA.view
.Views.23  3903_DIFF_FTB1_FEB5_SIGMA.view
.Views.24  3903_DIFF_Full_Compare.view
```

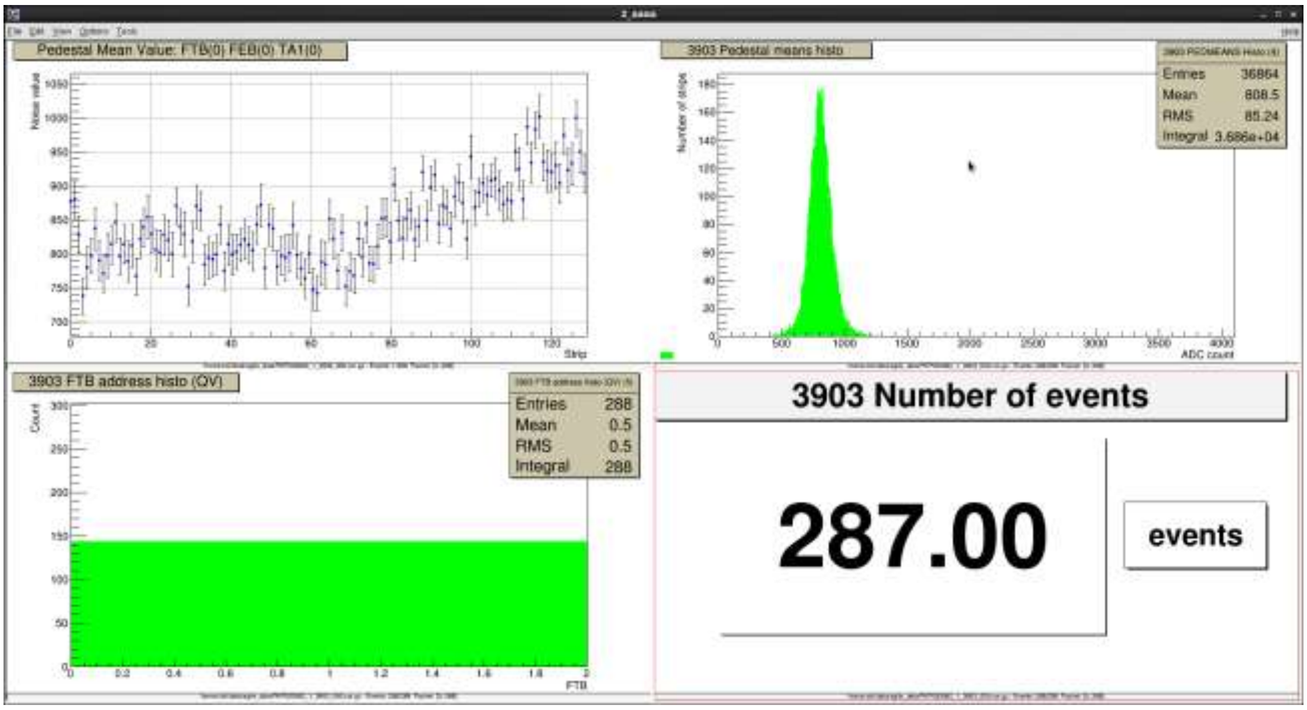
Views group defined using the file 3903.views

The view for the 3903_FTB0_FEB0_MEAN is the following.

```
.View.Name:                               3903_FTB0_FEB0_MEAN
.View.Title:                              3903_FTB0_FEB0 TAA1 noise (mean)
.View.id:                                 10000
.View.idDestination:                      1
.View.M:                                  6
.View.N:                                  4
.View.graphType.0:                        2048
.View.parameterName.0:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.0:       0
.View.graphType.1:                        2048
.View.parameterName.1:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.1:       1
.View.graphType.2:                        2048
.View.parameterName.2:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.2:       2
.View.graphType.3:                        2048
.View.parameterName.3:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.3:       3
.View.graphType.4:                        2048
.View.parameterName.4:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.4:       4
.View.graphType.5:                        2048
.View.parameterName.5:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.5:       5
.View.graphType.6:                        2048
.View.parameterName.6:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.6:       6
.View.graphType.7:                        2048
.View.parameterName.7:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.7:       7
.View.graphType.8:                        2048
.View.parameterName.8:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.8:       8
.View.graphType.9:                        2048
.View.parameterName.9:                    3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.9:       9
.View.graphType.10:                       2048
.View.parameterName.10:                   3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.10:      10
.View.graphType.11:                       2048
.View.parameterName.11:                   3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.11:      11
.View.graphType.12:                       2048
.View.parameterName.12:                   3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.12:      12
.View.graphType.13:                       2048
.View.parameterName.13:                   3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.13:      13
.View.graphType.14:                       2048
```

```
.View.parameterName.14:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.14:     14
.View.graphType.15:                      2048
.View.parameterName.15:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.15:     15
.View.graphType.16:                      2048
.View.parameterName.16:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.16:     16
.View.graphType.17:                      2048
.View.parameterName.17:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.17:     17
.View.graphType.18:                      2048
.View.parameterName.18:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.18:     18
.View.graphType.19:                      2048
.View.parameterName.19:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.19:     19
.View.graphType.20:                      2048
.View.parameterName.20:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.20:     20
.View.graphType.21:                      2048
.View.parameterName.21:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.21:     21
.View.graphType.22:                      2048
.View.parameterName.22:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.22:     22
.View.graphType.23:                      2048
.View.parameterName.23:                  3903_F_PEDMERR_0_0
.View.filterDynamicConfiguration.23:     23
.View.parametesNumber:                   24
```

Here it is a graphical result of some 3903 data visualization.



A screenshot of a custom 3903 data visualization. In order from left to right, top to bottom,

The grapherror_dataarray, histo_dataarray, histogram, and synoptic_single_value graph types

3.3 AGILE 3901 data model

The 3901 is a particular AGILE data model that do not rely on an input data model description. The handling of the input data type is carried out by the class InputData_GRID, and not through mapping/format files description for many reasons:

1. The original data format it was too complex to manage
2. For performance purpose
3. There was a lot of algorithms that works event-by-event. For this reason it was not convenient to translate a set of event into a set of tables (due to the complexity of data format) and convert them back to a single event.

For these reasons the 3901 input data type is handled using a specific loader called InputData_GRID.

Anyway we have to take care of the following steps in order to keeps things working:

1. Define an empty input data type.
2. The class InputData_GRID must inherit from QLInputData in order to be used by the chain manager and by the builders.
3. Assign an instance identifier for the QLInputData and register with the QLMain method:
   ```
   virtual void QLMain::registerInputData(std::string instanceName,
   QLInputData* instance);
   ```
4. Define a chain for the 3901 data type and specify the instance identifier to use in the chains.xml. The Quick Look component during the initialization phase will bind the class instance you have registered before to the chain.

```
<?xml version="1.0" encoding="UTF-8"?>
<type id="3901" DV="1" MCALThres="0" />
```

The 3901 input data type (3901_type.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<chains>
  <chain name="3901" type="3901" uniqueKey="3901" procChannel="1"
inputDataInstance="3901_IDP_Instance">
    <config DK="1000" DT="1000" DS="1000" qlrows="10" connect="yes"/>
  </chain>
</chains>
```

chains.conf

```
class InputData_GRID : public QLInputData {
public:

    InputData_GRID(TString chainName, qlbase::InputFile *inputFile);

    virtual ~InputData_GRID() {};
    virtual bool Load(const std::string& dataType);

    virtual Int_t Read(Long_t frow, Long_t lrow, ULong_t deltaS);

    virtual void Reset();

    void Print(Option_t *) const;

    TObjArray fData;

protected:

    GRID_Build_2DSingleTrack* track2DSingle;
    InputData_GRID_Filter filter1;

};
#endif
```

The 3901 input data type (3901_type.xml)

Once the input data is managed by the InputData_GRID class, it can be used by data builders and algorithms in order to obtain the intermediate data type that can be, finally, visualized. The way how the intermediate data types can be generated and visualized is the same of the 3903 described erarlier.

Just as example we report the 3901_CLUSTER intermediate data type.

```
<?xml version="1.0" encoding="UTF-8"?>
<type id="3901_CLUSTER" DV="10000">
  <field name="FTB" type="float" />
  <field name="FEB" type="float" />
  <field name="HDI" type="float" />
  <field name="TAA1" type="float" />
  <field name="CENTRAL_STRIP" type="float" />
  <field name="TOTCHARGE" type="float" />
  <field name="TOTWIDTH" type="float" />
  <field name="C1" type="float" />
  <field name="C2" type="float" />
  <field name="C3" type="float" />
  <field name="C4" type="float" />
  <field name="C5" type="float" />
  <field name="PLAN" type="float" />
  <field name="TAA1_FTB" type="float" />
  <field name="CENTSTRIP_PLAN" type="float" />
  <field name="ETA" type="float" />
  <field name="PULL" type="float" />
  <field name="BETADX" type="float" />
  <field name="BETASX" type="float" />
  <field name="THETA" type="float" />
  <field name="PHI" type="float" />
  <field name="CLUSTER_SELECTED" type="float" />
  <field name="SNR" type="float" />
  <field name="SCATTERRATIO" type="float" />
  <field name="TRACKDIM" type="float" />
  <field name="R" type="float" />
  <field name="PTYPE" type="float" />
  <field name="TIME" type="float" unit="us" />
</type>
```

The 3901_CLUSTER data type (3901_CLUSTER_type.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<graphs>
...
<graph id="3901_CLUS_PLAN" type="histogram" name="3901_CLUSTER - PLAN histo
(calculated) (QV)" title="Plan Histo for 3901 Cluster Events (calculated)">
    <xaxis ref="3901_CLUSTER/PLAN" label="Plan" nbins="12" min="0" max="12"
/>
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_FTB" type="histogram" name="3901_CLUSTER - FTB histo
(QV)" title="FTB Histo for 3901 Cluster Events">
    <xaxis ref="3901_CLUSTER/FTB" label="FTB number" nbins="2" min="0"
max="2" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_FEB" type="histogram" name="3901_CLUSTER - FEB histo
(QV)" title="FEB Histo for 3901 Cluster Events">
    <xaxis ref="3901_CLUSTER/FEB" label="FEB number" nbins="6" min="0"
max="6" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_HDI" type="histogram" name="3901_CLUSTER - HDI histo
(calculated) (QV)" title="HDI Histo for 3901 Cluster Events (calculated)">
    <xaxis ref="3901_CLUSTER/HDI" label="HDI number" nbins="4" min="0"
max="4" />
    <yaxis label="Count" />
</graph>
```

```
<graph id="3901_CLUS_TAA1" type="histogram" name="3901_CLUSTER - TAA1 histo
(QV)" title="TAA1 Histo for 3901 Cluster Events">
    <xaxis ref="3901_CLUSTER/TAA1" label="TAA1 address" nbins="24" min="0"
max="24" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_CENTRAL_STRIP" type="histogram" name="3901
CENTRAL_STRIP histo (QV)" title="Central Strip histo for 3901 Events">
    <xaxis ref="3901_CLUSTER/CENTRAL_STRIP" label="Central strip"
nbins="128" min="0" max="128" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_TOTCHARGE" type="histogram" name="3901 TOTCHARGE"
title="TOTCHARGE Histo for Grid Event 3901">
    <xaxis ref="3901_CLUSTER/TOTCHARGE" label="ADC count" nbins="20000"
min="0" max="40000" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_TOTWIDTH" type="histogram" name="3901 TOTWIDTH"
title="TOTWIDTH Histo for Grid Event 3901">
    <xaxis ref="3901_CLUSTER/TOTWIDTH" label="ADC count" nbins="16" min="0"
max="16" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_C1" type="histogram" name="3901 Charge histo of C1"
title="3901 Charge histo of C1" yMinLimit="32" yMaxLimit="4096">
    <xaxis ref="3901_CLUSTER/C1" label="ADC count (12 bit)" nbins="256"
min="0" max="4096" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_C2" type="histogram" name="3901 Charge histo of C2"
title="3901 Charge histo of C2" yMinLimit="32" yMaxLimit="4096">
    <xaxis ref="3901_CLUSTER/C2" label="ADC count (12 bit)" nbins="256"
min="0" max="4096" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_C3" type="histogram" name="3901 Charge histo central
strip of cluster (C3)" title="3901 Histo charge of cluster central strip">
    <xaxis ref="3901_CLUSTER/C3" label="ADC count (12 bit)" nbins="256"
min="0" max="4096" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_C4" type="histogram" name="3901 Charge histo of C4"
title="3901 Charge histo of C4" yMinLimit="32" yMaxLimit="4096">
    <xaxis ref="3901_CLUSTER/C4" label="ADC count (12 bit)" nbins="256"
min="0" max="4096" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_C5" type="histogram" name="3901 Charge histo of C5"
title="3901 Charge histo of C5" yMinLimit="32" yMaxLimit="4096">
    <xaxis ref="3901_CLUSTER/C5" label="ADC count (12 bit)" nbins="256"
min="0" max="4096" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_TAA1FTB" type="histogram" name="3901 Chip address (0-
143)" title="3901 Chip address (0-143)">
```

```
    <xaxis ref="3901_CLUSTER/TAA1_FTB" label="TAA1 address" nbins="144"
min="0" max="144" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_CENTSTRIP_PLAN" type="histogram" name="3901 Strip
address (0-1535)" title="3901 Strip address (0-1535)">
    <xaxis ref="3901_CLUSTER/CENTSTRIP_PLAN" label="Strip address"
nbins="1536" min="0" max="1536" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ETA" type="histogram" name="3901 Eta" title="3901 Eta"
yMinLimit="0.03" yMaxLimit="1" minYhist="0.01" maxYhist="1">
    <xaxis ref="3901_CLUSTER/ETA" label="Eta function" nbins="100" min="0"
max="1" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_C1C5" type="histogram" name="3901 Charge histo of C1
and C5" title="3901 Charge histo of C1 and C5" yMinLimit="32"
yMaxLimit="4096">
    <xaxis ref="3901_CLUSTER/C1" label="ADC count (12 bit)" nbins="256"
min="0" max="4096" />
    <yaxis ref="3901_CLUSTER/C5" label="Count" nbins="256" min="0"
max="4096" />
</graph>
<graph id="3901_CLUS_C2C4" type="histogram" name="3901 Charge histo of C2
and C4" title="3901 Charge histo of C2 and C4" yMinLimit="32"
yMaxLimit="4096">
    <xaxis ref="3901_CLUSTER/C2" label="ADC count (12 bit)" nbins="256"
min="0" max="4096" />
    <yaxis ref="3901_CLUSTER/C4" label="Count" nbins="256" min="0"
max="4096" />
</graph>
<graph id="3901_CLUS_PULL" type="histogram" name="3901 Pull" title="3901
Pull">
    <xaxis ref="3901_CLUSTER/PULL" label="Pull - strip max" nbins="320"
min="0" max="160" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_BETADX" type="histogram" name="3901 Beta dx"
title="3901 Beta dx" yMinLimit="0.01" yMaxLimit="1.5">
    <xaxis ref="3901_CLUSTER/BETADX" label="Beta dx function" nbins="1500"
min="0" max="1.5" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_BETASX" type="histogram" name="3901 Beta sx"
title="3901 Beta sx" yMinLimit="0.01" yMaxLimit="1.5">
    <xaxis ref="3901_CLUSTER/BETASX" label="Beta sx function" nbins="1500"
min="0" max="1.5" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_SNR" type="histogram" name="3901 SNR" title="3901
SNR">
    <xaxis ref="3901_CLUSTER/SNR" label="SNR" nbins="320" min="0" max="160"
/>
    <yaxis label="Count" />
</graph>
```

```
<graph id="3901_CLUS_PTYPE" type="histogram" name="3901 PTYPE" title="3901
PTYPE">
    <xaxis ref="3901_CLUSTER/PTYPE" label="PTYPE" nbins="51" min="0"
max="51" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_PLAN" type="histogram" name="3901_CLUSTER_STRIP -
PLAN histo (calculated) (QV)" title="Plan Histo for 3901 Cluster Events
(calculated)">
    <xaxis ref="3901_CLUSTER_STRIP/PLAN" label="Plan" nbins="12" min="0"
max="12" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_FTB" type="histogram" name="3901_CLUSTER_STRIP -
FTB histo (QV)" title="FTB Histo for 3901 Cluster Events">
    <xaxis ref="3901_CLUSTER_STRIP/FTB" label="FTB number" nbins="2"
min="0" max="2" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_FEB" type="histogram" name="3901_CLUSTER_STRIP -
FEB histo (QV)" title="FEB Histo for 3901 Cluster Events">
    <xaxis ref="3901_CLUSTER_STRIP/FEB" label="FEB number" nbins="6"
min="0" max="6" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_HDI" type="histogram" name="3901_CLUSTER_STRIP -
HDI histo (calculated) (QV)" title="HDI Histo for 3901 Cluster Events
(calculated)">
    <xaxis ref="3901_CLUSTER_STRIP/HDI" label="HDI number" nbins="4"
min="0" max="4" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_TAA1" type="histogram" name="3901_CLUSTER_STRIP -
TAA1 histo (QV)" title="TAA1 Histo for 3901 Cluster Events">
    <xaxis ref="3901_CLUSTER_STRIP/TAA1" label="TAA1 address" nbins="24"
min="0" max="24" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_STRIP" type="histogram" name="3901_CLUSTER_STRIP
STRIP histo (QV)" title="3901 Full Strip histo">
    <xaxis ref="3901_CLUSTER_STRIP/STRIP" label="Central strip" nbins="128"
min="0" max="128" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_CHARGE" type="histogram" name="3901_CLUSTER_STRIP
Charge histo" title="3901 Full Cluster Charge histo" yMinLimit="32"
yMaxLimit="4096">
    <xaxis ref="3901_CLUSTER_STRIP/CHARGE" label="ADC count (12 bit)"
nbins="256" min="0" max="4096" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_TAA1FTB" type="histogram" name="3901_CLUSTER_STRIP
Chip address (0-143)" title="3901 Full Chip address (0-143)">
    <xaxis ref="3901_CLUSTER_STRIP/TAA1_FTB" label="TAA1 address"
nbins="144" min="0" max="144" />
    <yaxis label="Count" />
</graph>
```

```
<graph id="3901_CLUS_ST_STRIP_PLAN" type="histogram"
name="3901_CLUSTER_STRIP Strip address (0-1535)" title="3901 Full Strip
address (0-1535)">
    <xaxis ref="3901_CLUSTER_STRIP/STRIP_PLAN" label="Strip address"
nbins="1536" min="0" max="1536" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_BETADX" type="histogram" name="3901_CLUSTER_STRIP
Beta dx" title="3901 cluster strip Beta dx" yMinLimit="0.01"
yMaxLimit="1.5">
    <xaxis ref="3901_CLUSTER_STRIP/BETADX" label="Beta dx function"
nbins="1500" min="0" max="1.5" />
    <yaxis label="Count" />
</graph>
<graph id="3901_CLUS_ST_BETASX" type="histogram" name="3901_CLUSTER_STRIP
Beta sx" title="3901 cluster strip Beta sx" yMinLimit="0.01"
yMaxLimit="1.5">
    <xaxis ref="3901_CLUSTER_STRIP/BETASX" label="Beta sx function"
nbins="1500" min="0" max="1.5" />
    <yaxis label="Count" />
</graph>
...
</graphs>
```

The 3901_CLUSTER graphs (3901_graph.xml)

3.4 Simple image data model

All the data models we have seen before have fields based on simple types or array of simple types allowing the representation, for example, of the FITS tables. The CIWS-FW supports also images as simple types. A simple input data model to handle 2d images from FITS could be modeled using the following xmls.

```
<?xml version="1.0" encoding="UTF-8"?>
<type id="img">
  <field name="imgdata" type="image2d" />
</type>
```

The img data type (img_type.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<ddl>
  <type id="imgdata" name="imgdata">
    <data>
      <image dimensions="2" pixType="float32" description="a testing image"
/>
    </data>
  </type>
</ddl>
```

The img data format (img_format.xml)

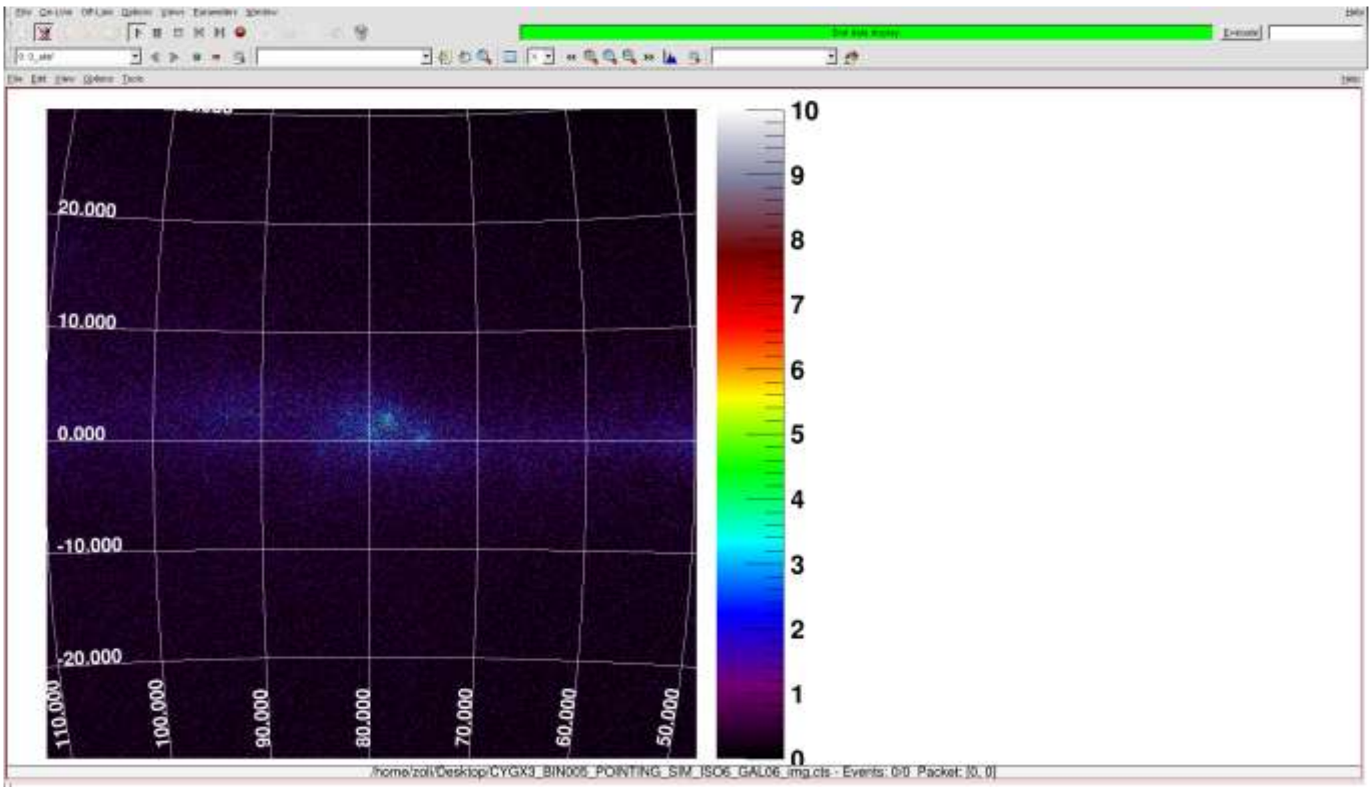
```
<?xml version="1.0" encoding="UTF-8"?>
<mapping type="img" inputFormat="fits">
  <link field="imgdata" ref="imgdata" />
</mapping>
```

The img data mapping (img_mapping.xml)

The 2d image, using the Quick Look component, can be visualized as an histogram 2d or as an astronomical image. In the following graph file we used the latter one, defining “image2d” as type.

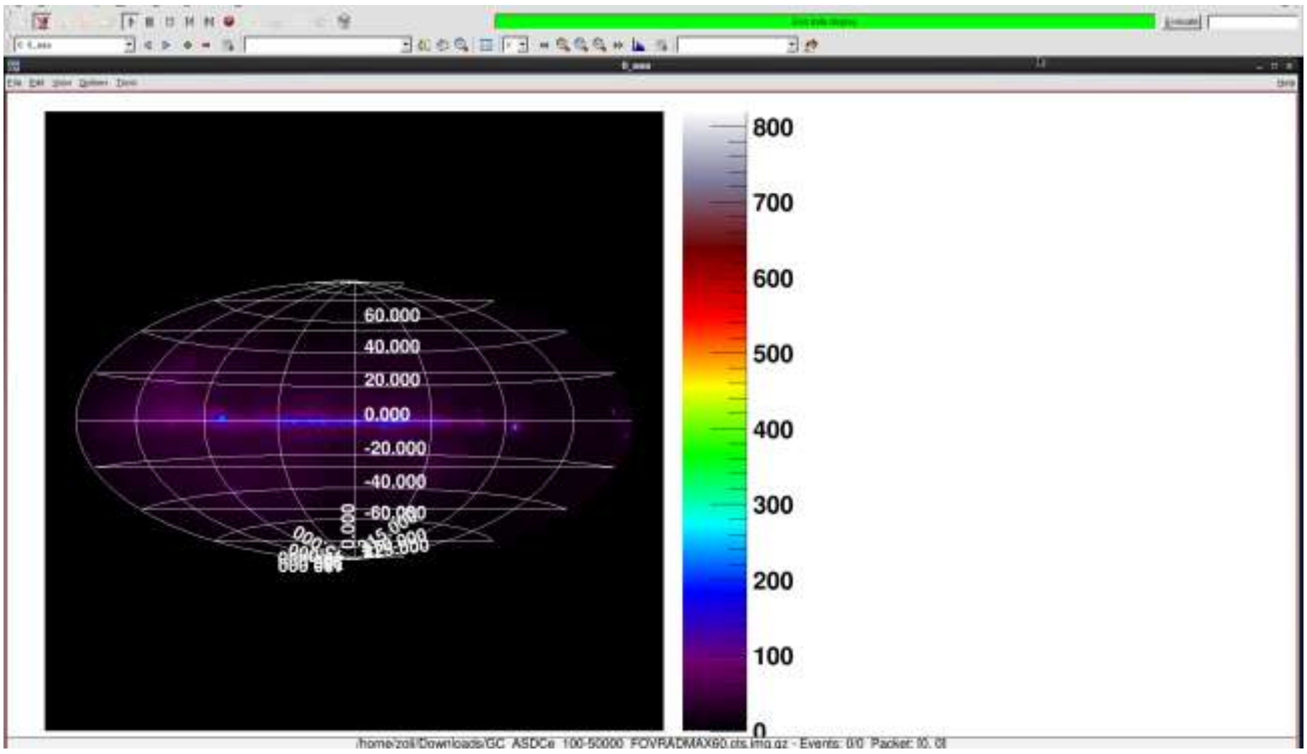
```
<?xml version="1.0" encoding="UTF-8"?>
<graphs>
<graph id="immagine di prova" type="image2d" name="immagine di prova"
title="immagine di prova">
<xaxis ref="img/imgdata" label="asse x" />
<yaxis ref="img/imgdata" label="asse y" />
</graph>
</graphs>
```

The graph for the img (img_graph.xml)



/home/zoli/Desktop/CYGX3_BIN005_POINTING_SIM_ISO6_GAL06_img.cts - Events: 0/0 Packet: [0, 0]

Screenshot of the Cygnus Region visualized as astronomical image



/home/zoli/Downloads/GC_ASDCe_100-50000_FOVRADMAX60.cts.img.gz - Events: 0/0 Packet: [0, 0]

Screenshot of an AGILE Aitoff