Archiving and accessing PVA data at ITER

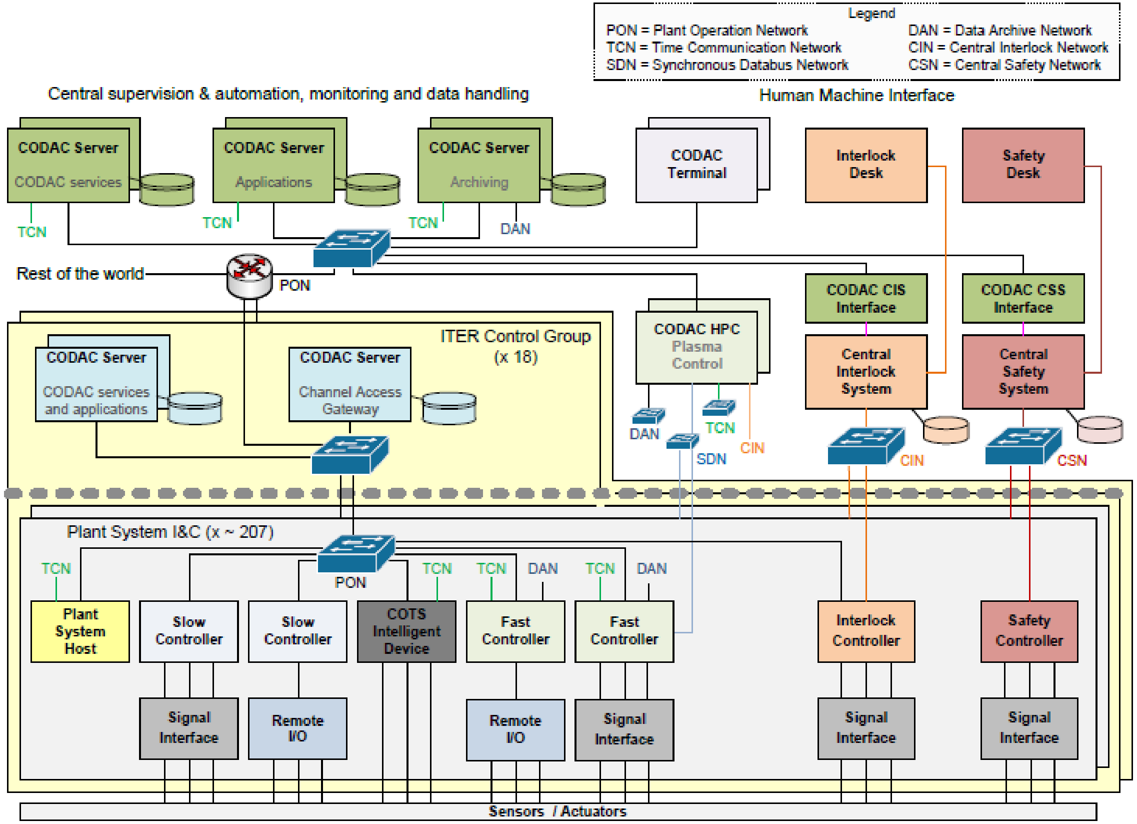
Lana Abadie [1], Bertrand Bauvir [1], Rodrigo Castro [2], Ralph Lange [1],Yury Makushok [3], Andre Neto [4]

[1] ITER Organization
[2] CIEMAT
[3] INDRA
[4] F4E

Disclaimer: The views and opinions expressed herein do not necessarily reflect those of the ITER Organization

Background information

DAN – data archiving network
SDN – real-time network
PON – epics network
TCN : timing network



Challenges/Requirement

1. High data rates
    1. up to 2GB/sec for one channel - DAN
    2. Up to 10KHz, max. 64KB payloads – SDN
    3. Up 10Hz (Epics)
2. Dual data streaming to internal and external network to warranty similar data access time
3. Data access :
    1. Uniform data access regardless of the publishers (DAN,SDN, PON,CIS,CSS)
    2. Performance : some operations like data visualization need to be very fast a few seconds to retrieve and plot hundreds of signals
    3. Concurrency: 100-1000 concurrent access
    4. Need to access the full structure (Data processing)
    5. Need to access the leaf of structure tree (data visualization)
    6. Need to access a sub-tree of structure (Data processing)
    7. Need to know what was the structure definition at given time

Data archiving repositories – 1st version

- PON (EPICS) archiving systems ~ many signals but slow
  - BEAUTY (RDB) – community tool, main client is CS-Studio
  - PVA archiver in-house development, supports PVAccess and Channel Access, use of HDF5 files -> structured data
- SDN (real time) archiving systems
  - All SDN data is captured
  - up to 10KHz, like PVAccess, structure can be complex
- DAN (daq) archiving systems
  - Atomic data type or simple structure
  - Up to 50GB/sec (camera data)
- CIS (interlock) slow but important
  - Data from plant interlock will be archived in BEAUTY
  - Critical data will be replicated in real-time to CODAC
- CSS (safety) –N/OS slow but important
  - Regular snapshot of their repository with data transformation

Is it really code optimized?

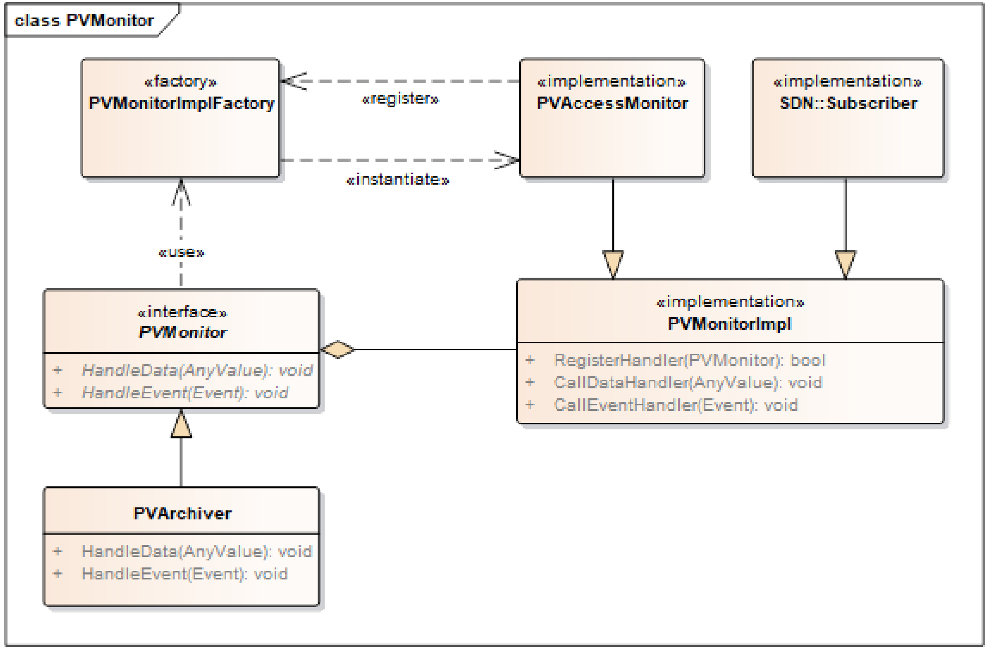
- **PVA @ITER :** extensive use of user-defined structures, mapping complex system states to few structured PV
- Why not merging PVA and SDN archiver and eventually PON archiver
  - Similar front-end code, abstract the transport layer (CA, PVA or SDN)
  - Back-end : support for writing data to files
  - Disconnection/connection events to be logged instead of merging that into the archived data to avoid data structures disruptions

Design and implementation consideration

- Encapsulation
  - External third party software is hidden from user
  - No direct exposure of the HDF5 layout to end-user
- Modularity
  - Different plugins to read and write data (PON, SDN, DAN)
- Code reusability
  - When it is possible, minimize code : e.g.
  - PVA and SDN archivers will share same front-end and back-end. The transport plugin is loaded at run-time
- Code Quality
  - SDN and DAN archivers are SWIL-1
  - We need to reach a code coverage >95%
  - Standard checks (cppcheck)
  - Sonar report

Example of code reusability

- PON archiving and SDN archiving systems – same interface different implementation – collaboration with Fusion for Energy
- Split into front-ends/back-ends


class PVMonitor
«factory»
PVMonitorImplFactory
«register»
«instantiate»
«implementation»
PVAccessMonitor
«implementation»
SDN::Subscriber
«use»
«interface»
PVMonitor
+ HandleData(AnyValue): void
+ HandleEvent(Event): void
«implementation»
PVMonitorImpl
+ RegisterHandler(PVMonitor): bool
+ CallDataHandler(AnyValue): void
+ CallEventHandler(Event): void
PVArchiver
+ HandleData(AnyValue): void
+ HandleEvent(Event): void

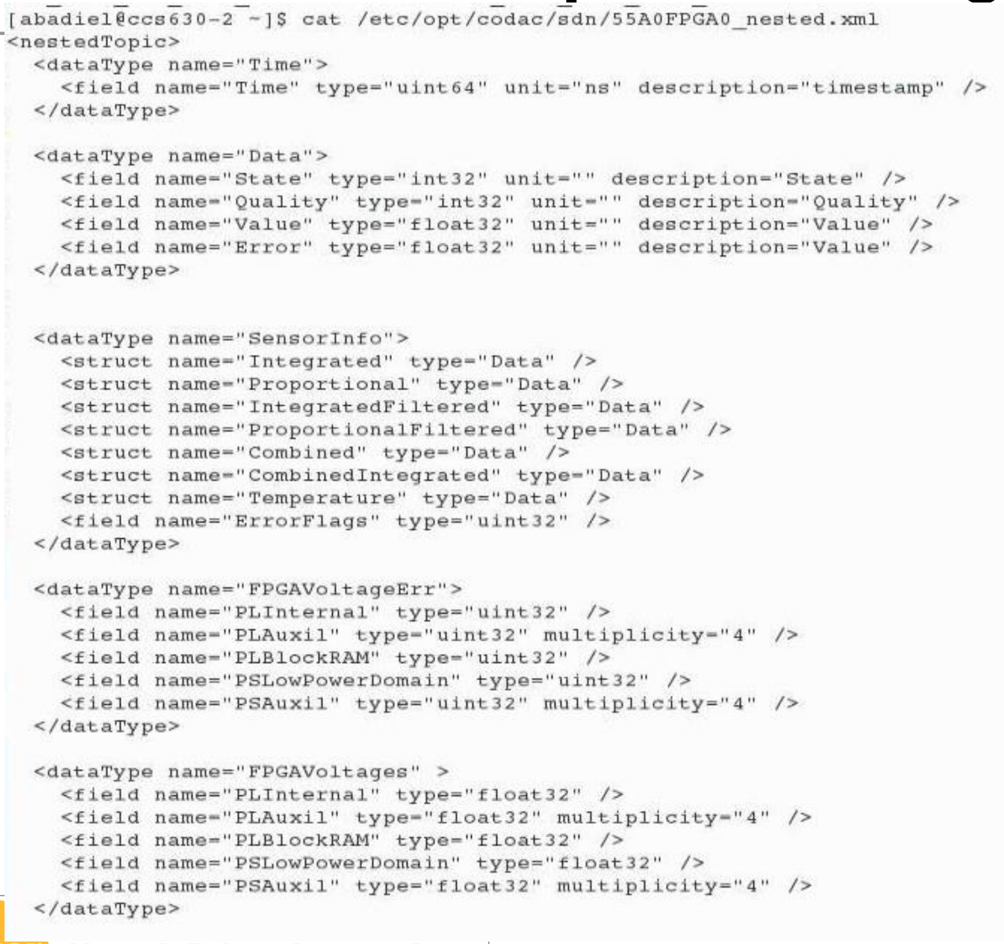
Current Status (1/2)

- SDN archiver and PVA archiver first version have been produced (two different code bases)
- Demonstrated support for storing complex structures such as magnetics structures
- Now code refactored to have common code bases, first prototype achieved a few days ago…
- Both archivers have a support for file rotation with a configurable file size (to support long continuous acquisition)
- HDF5 files are produced (1 file per PVA)

Current Status (2/2)

- Configuration
  - PVA and SDN archivers can be started using a XML configuration file to create the structure and to have metadata like description, units, field which corresponds to the timestamp field
  - If there is no XML configuration, can discover the structure on the fly and create the file : however when you start the tool you need to specify which field is the timestamp. And in that case there is no recording of units/description
- Timestamp : represented in nanoseconds since Epoch Linux Time as uint64 (DAN, SDN and PVA)
- HDF5 files
  - Use of SWMR (1.12), C API
  - All codes is in C++

Example of configuration file



```
[abadiel@ccs630-2 ~]$ cat /etc/opt/codac/sdn/55A0FPGA0_nested.xml
<nestedTopic>
  <dataType name="Time">
    <field name="Time" type="uint64" unit="ns" description="timestamp" />
  </dataType>

  <dataType name="Data">
    <field name="State" type="int32" unit="" description="State" />
    <field name="Quality" type="int32" unit="" description="Quality" />
    <field name="Value" type="float32" unit="" description="Value" />
    <field name="Error" type="float32" unit="" description="Value" />
  </dataType>


  <dataType name="SensorInfo">
    <struct name="Integrated" type="Data" />
    <struct name="Proportional" type="Data" />
    <struct name="IntegratedFiltered" type="Data" />
    <struct name="ProportionalFiltered" type="Data" />
    <struct name="Combined" type="Data" />
    <struct name="CombinedIntegrated" type="Data" />
    <struct name="Temperature" type="Data" />
    <field name="ErrorFlags" type="uint32" />
  </dataType>

  <dataType name="FPGAVoltageErr">
    <field name="PLInternal" type="uint32" />
    <field name="PLAuxil" type="uint32" multiplicity="4" />
    <field name="PLBlockRAM" type="uint32" />
    <field name="PSLowPowerDomain" type="uint32" />
    <field name="PSAuxil" type="uint32" multiplicity="4" />
  </dataType>

  <dataType name="FPGAVoltages" >
    <field name="PLInternal" type="float32" />
    <field name="PLAuxil" type="float32" multiplicity="4" />
    <field name="PLBlockRAM" type="float32" />
    <field name="PSLowPowerDomain" type="float32" />
    <field name="PSAuxil" type="float32" multiplicity="4" />
  </dataType>
```

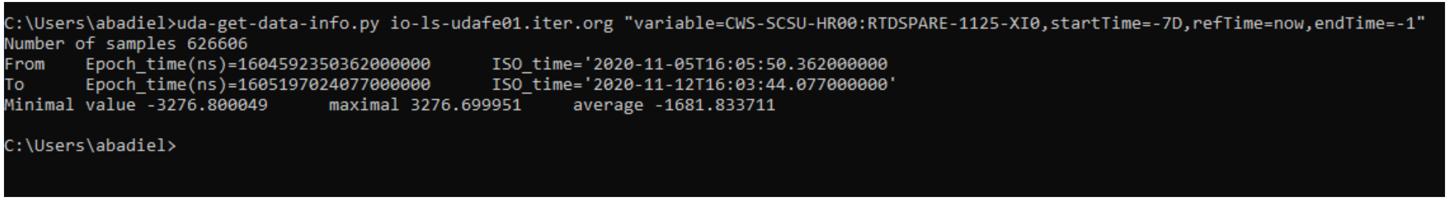
Data access

- All HDF5 files are automatically indexed by an agent
  - Watch for file appearance, structure creation and file closure
  - Use of a Postgresql database to store information about variable and files
  - Extraction of the structure into postgresql to speed up transversal search
- UDA (unified data access) – data access server
  - to retrieve the data structure at a given time
  - To retrieve full data for a given time window
  - To retrieve a given structure leaf for a given time window

A few snapshots of the API utilities

Example of Matlab script using UDA API

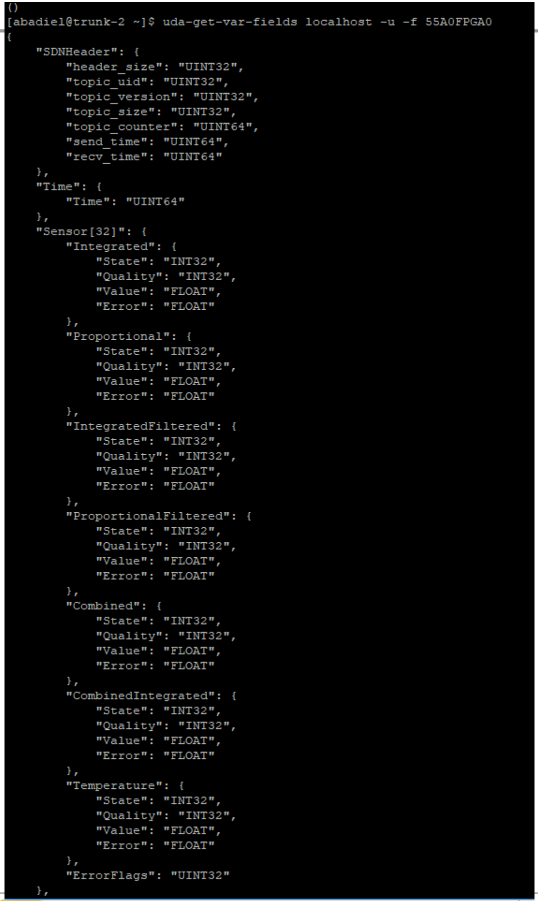
```
UCR = uda_client_reader.UdaClientReaderMatlab('io-ls-udafe01.iter.org', 3090);
Req="variable=CWS-SCSU-HR00:ML0004-LT-XI,startTime=2020-11-01T00:00:01,endTime=-1"
handle = UCR.fetchData(char(req));
If handle<0
Fprintf("request failed %s", UCR.getErrorMsg() )
else
Data = UCR.getDataAsDouble(handle);
TimeStamps = UCR.getTimeStampsAsLong(handle);
Unit=UCR.getUnitsY(handle);
end
```



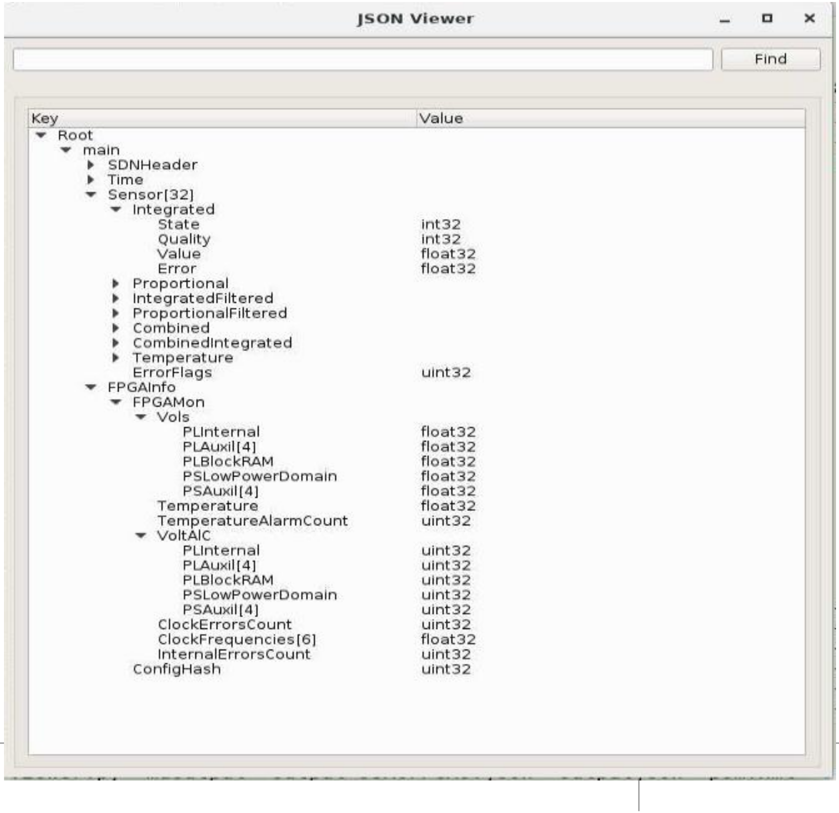
```
C:\Users\abadiel>uda-get-data-info.py io-ls-udafe01.iter.org "variable=CWS-SCSU-HR00:RTDSPARE-1125-XI0,startTime=-7D,refTime=now,endTime=-1"
Number of samples 626606
From    Epoch_time(ns)=1604592350362000000      ISO_time='2020-11-05T16:05:50.362000000
To      Epoch_time(ns)=1605197024077000000      ISO_time='2020-11-12T16:03:44.077000000'
Minimal value -3276.800049      maximal 3276.699951     average -1681.833711

C:\Users\abadiel>
```

A few snapshots of the API utilities

Uda-get-var-fields (without X-term and with X-term



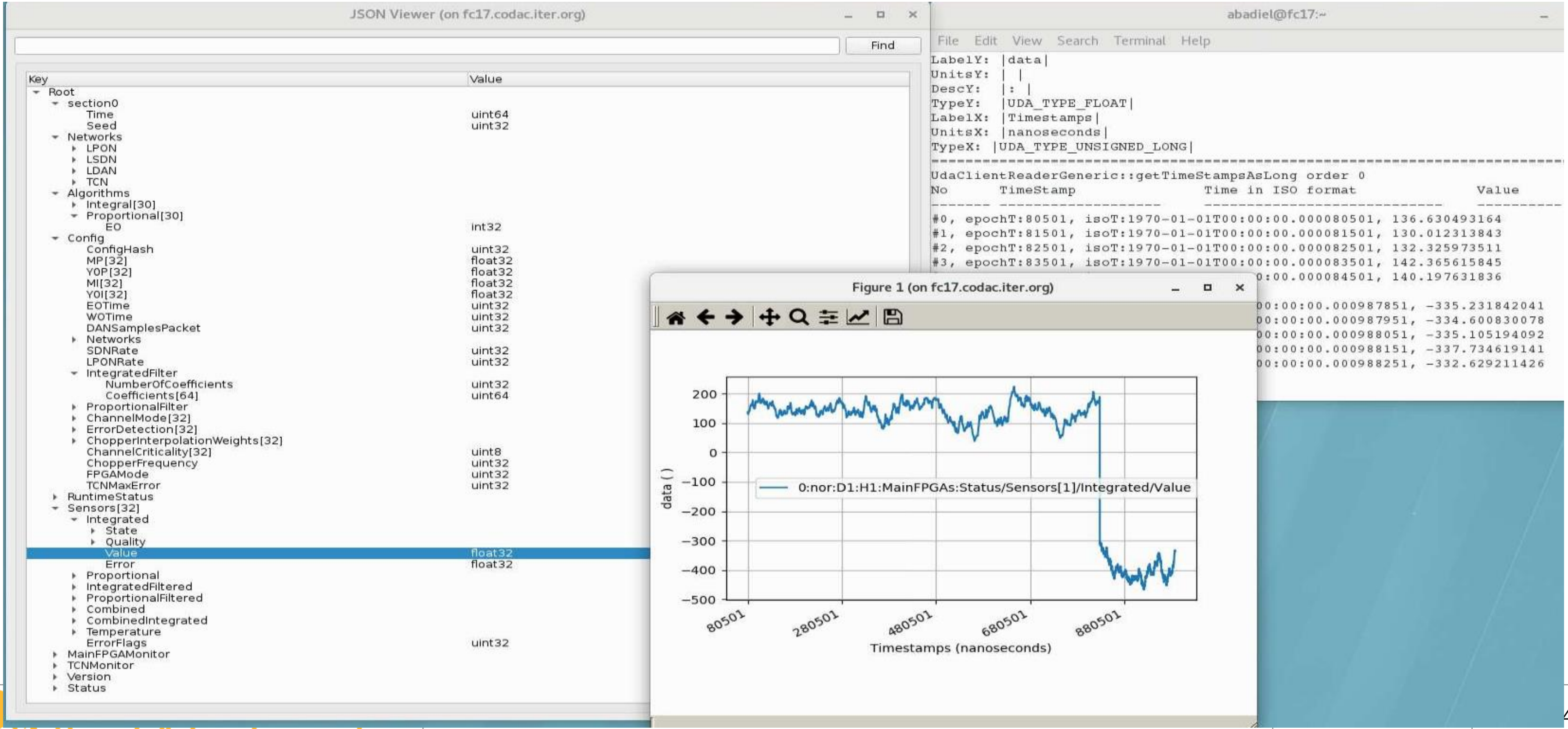
```
()
[abadiel@trunk-2 ~]$ uda-get-var-fields localhost -u -f 55A0FPGA0
{
    "SDNHeader": {
        "header_size": "UINT32",
        "topic_uid": "UINT32",
        "topic_version": "UINT32",
        "topic_size": "UINT32",
        "topic_counter": "UINT64",
        "send_time": "UINT64",
        "recv_time": "UINT64"
    },
    "Time": {
        "Time": "UINT64"
    },
    "Sensor[32]": {
        "Integrated": {
            "State": "INT32",
            "Quality": "INT32",
            "Value": "FLOAT",
            "Error": "FLOAT"
        },
        "Proportional": {
            "State": "INT32",
            "Quality": "INT32",
            "Value": "FLOAT",
            "Error": "FLOAT"
        },
        "IntegratedFiltered": {
            "State": "INT32",
            "Quality": "INT32",
            "Value": "FLOAT",
            "Error": "FLOAT"
        },
        "ProportionalFiltered": {
            "State": "INT32",
            "Quality": "INT32",
            "Value": "FLOAT",
            "Error": "FLOAT"
        },
        "Combined": {
            "State": "INT32",
            "Quality": "INT32",
            "Value": "FLOAT",
            "Error": "FLOAT"
        },
        "CombinedIntegrated": {
            "State": "INT32",
            "Quality": "INT32",
            "Value": "FLOAT",
            "Error": "FLOAT"
        },
        "Temperature": {
            "State": "INT32",
            "Quality": "INT32",
            "Value": "FLOAT",
            "Error": "FLOAT"
        },
        "ErrorFlags": "UINT32"
    },
```



```
Key                                  Value
▾ Root
  ▾ main
    ▸ SDNHeader
    ▸ Time
    ▾ Sensor[32]
      ▾ Integrated
          State                      int32
          Quality                    int32
          Value                      float32
          Error                      float32
      ▸ Proportional
      ▸ IntegratedFiltered
      ▸ ProportionalFiltered
      ▸ Combined
      ▸ CombinedIntegrated
      ▸ Temperature
        ErrorFlags                   uint32
    ▾ FPGAInfo
      ▾ FPGAMon
        ▾ Vols
            PLInternal               float32
            PLAuxil[4]               float32
            PLBlockRAM               float32
            PSLowPowerDomain         float32
            PSAuxil[4]               float32
          Temperature                float32
          TemperatureAlarmCount      uint32
        ▾ VoltAlC
            PLInternal               uint32
            PLAuxil[4]               uint32
            PLBlockRAM               uint32
            PSLowPowerDomain         uint32
            PSAuxil[4]               uint32
          ClockErrorsCount           uint32
          ClockFrequencies[6]        float32
          InternalErrorsCount        uint32
        ConfigHash                   uint32
```

A few snapshots of the API utilities

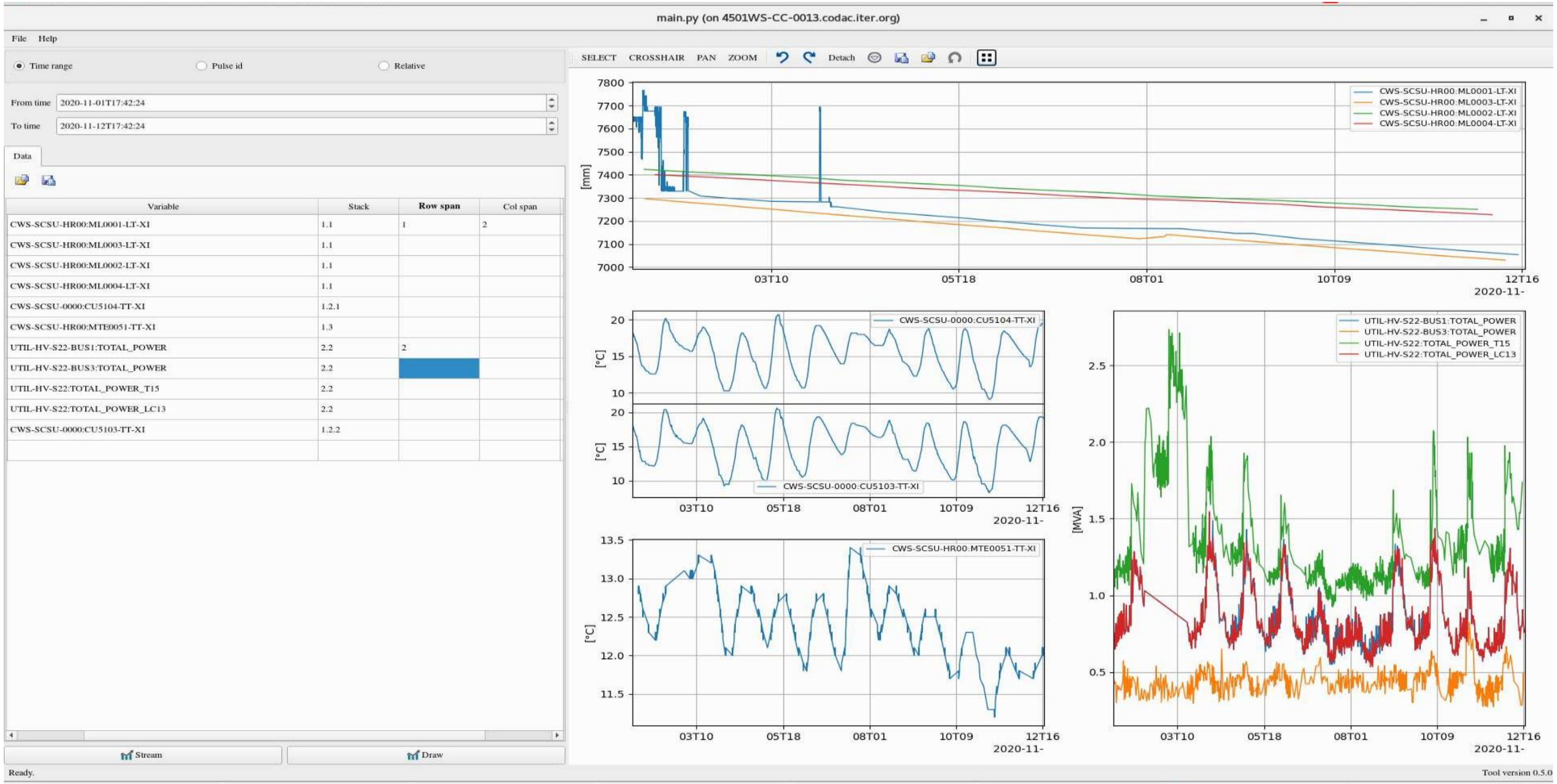
Uda-get-data/plot


JSON Viewer (on fc17.codac.iter.org)
Find
abadiel@fc17:~
File Edit View Search Terminal Help
Figure 1 (on fc17.codac.iter.org)
200
100
0
−100
−200
−300
−400
−500
data ( )
0:nor:D1:H1:MainFPGAs:Status/Sensors[1]/Integrated/Value
80501
280501
480501
680501
880501
Timestamps (nanoseconds)

```
Key                                          Value
Root
  section0
    Time                                     uint64
    Seed                                     uint32
  Networks
    LPON
    LSDN
    LDAN
    TCN
  Algorithms
    Integral[30]
    Proportional[30]
      EO                                     int32
  Config
    ConfigHash                               uint32
    MP[32]                                   float32
    YOP[32]                                  float32
    MI[32]                                   float32
    YOI[32]                                  float32
    EOTime                                   uint32
    WOTime                                   uint32
    DANSamplesPacket                         uint32
    Networks
    SDNRate                                  uint32
    LPONRate                                 uint32
    IntegratedFilter
      NumberOfCoefficients                   uint32
      Coefficients[64]                       uint64
    ProportionalFilter
    ChannelMode[32]
    ErrorDetection[32]
    ChopperInterpolationWeights[32]
    ChannelCriticality[32]                   uint8
    ChopperFrequency                         uint32
    FPGAMode                                 uint32
    TCNMaxError                              uint32
  RuntimeStatus
  Sensors[32]
    Integrated
      State
      Quality
      Value                                  float32
      Error                                  float32
    Proportional
    IntegratedFiltered
    ProportionalFiltered
    Combined
    CombinedIntegrated
    Temperature
    ErrorFlags                               uint32
  MainFPGAMonitor
  TCNMonitor
  Version
  Status
```

```
LabelY:  |data|
UnitsY:  | |
DescY:   |: |
TypeY:   |UDA_TYPE_FLOAT|
LabelX:  |Timestamps|
UnitsX:  |nanoseconds|
TypeX: |UDA_TYPE_UNSIGNED_LONG|
--------------------------------------------------------------------------------
UdaClientReaderGeneric::getTimeStampsAsLong order 0
No       TimeStamp                    Time in ISO format                Value
------- -------------------    ---------------------------     ----------
#0, epochT:80501, isoT:1970-01-01T00:00:00.000080501, 136.630493164
#1, epochT:81501, isoT:1970-01-01T00:00:00.000081501, 130.012313843
#2, epochT:82501, isoT:1970-01-01T00:00:00.000082501, 132.325973511
#3, epochT:83501, isoT:1970-01-01T00:00:00.000083501, 142.365615845
                                    0:00:00.000084501, 140.197631836

                                    00:00:00.000987851, -335.231842041
                                    00:00:00.000987951, -334.600830078
                                    00:00:00.000988051, -335.105194092
                                    00:00:00.000988151, -337.734619141
                                    00:00:00.000988251, -332.629211426
```

A few snapshots of the API utilities



main.py (on 4501WS-CC-0013.codac.iter.org)

● Time range ○ Pulse id ○ Relative

From time 2020-11-01T17:42:24

To time 2020-11-12T17:42:24

Variable	Stack	Row span	Col span
CWS-SCSU-HR00:ML0001-LT-XI	1.1	1	2
CWS-SCSU-HR00:ML0003-LT-XI	1.1		
CWS-SCSU-HR00:ML0002-LT-XI	1.1		
CWS-SCSU-HR00:ML0004-LT-XI	1.1		
CWS-SCSU-0000:CU5104-TT-XI	1.2.1		
CWS-SCSU-HR00:MTE0051-TT-XI	1.3		
UTIL-HV-S22-BUS1:TOTAL_POWER	2.2	2	
UTIL-HV-S22-BUS3:TOTAL_POWER	2.2		
UTIL-HV-S22:TOTAL_POWER_T15	2.2		
UTIL-HV-S22:TOTAL_POWER_LC13	2.2		
CWS-SCSU-0000:CU5103-TT-XI	1.2.2		

Ready.

Tool version 0.5.0

File
Help
Data
Stream
Draw
SELECT
CROSSHAIR
PAN
ZOOM
Detach
[mm]
7800
7700
7600
7500
7400
7300
7200
7100
7000
CWS-SCSU-HR00:ML0001-LT-XI
CWS-SCSU-HR00:ML0003-LT-XI
CWS-SCSU-HR00:ML0002-LT-XI
CWS-SCSU-HR00:ML0004-LT-XI
03T10
05T18
08T01
10T09
12T16
2020-11-
[°C]
20
15
10
CWS-SCSU-0000:CU5104-TT-XI
CWS-SCSU-0000:CU5103-TT-XI
13.5
13.0
12.5
12.0
11.5
CWS-SCSU-HR00:MTE0051-TT-XI
[MVA]
2.5
2.0
1.5
1.0
0.5
UTIL-HV-S22-BUS1:TOTAL_POWER
UTIL-HV-S22-BUS3:TOTAL_POWER
UTIL-HV-S22:TOTAL_POWER_T15
UTIL-HV-S22:TOTAL_POWER_LC13

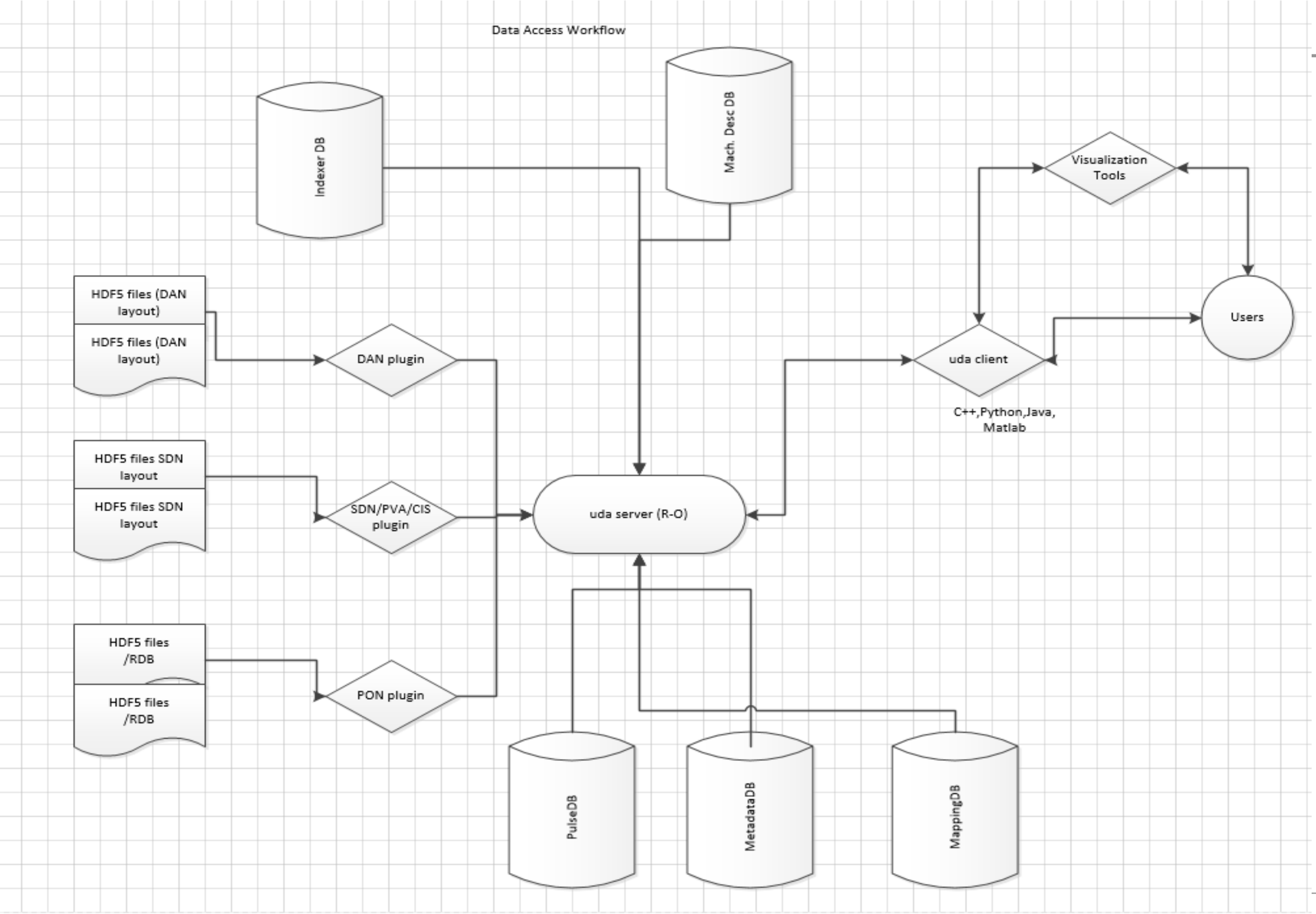
Metrics / Monitoring

- PVA/SDN archiver and DAN archiver produces metrics (number of sample lost, archived samples, number of writers)
  - Use of collectd
  - Influxdb to collect the metrics
  - Grafana to create dashboard
- Use of centreon to monitor the machines (CPU,mem,disk) and to get alerts

Conclusions/Discussions

- Good progress on PVA/SDN archiver code reuse
- Discussion
  - Early adapters of PVAccess, normative type is a nice concept but of very limited use at ITER
  - PVA supports user-defined structures and it is very good!
  - What is the path to integrate into CS-Studio/EPICS ecosystem?

Data Access – architecture


Data Access Workflow
Indexer DB
Mach. Desc DB
Visualization Tools
HDF5 files (DAN layout)
HDF5 files (DAN layout)
DAN plugin
uda client
Users
C++,Python,Java, Matlab
HDF5 files SDN layout
HDF5 files SDN layout
SDN/PVA/CIS plugin
uda server (R-O)
HDF5 files /RDB
HDF5 files /RDB
PON plugin
PulseDB
MetadataDB
MappingDB