Conditions database ART service and C++ interface

Ana Paula Vizcaya

5 - March - 2024

Colorado State University

Art and C++ interface with database

- New (updated) art an c++ interface with conditions database
- Based on: /nuevdb/IFDatabase
  - More functions (upload data, load data from other databases)
  - Can't upload more that 3 columns or so (bug)
  - Can't specify new URL or interface with new conditions database
- New code
  - Different options:
    - Provide URL
    - Provide table name and schema
  - Still uses some of the old code
    - Column
    - Row

ART and C++ code

- The code
- General code to access the conditions tables in the conditions database

```
RunConditions.cxx
RunConditions.h
```

- Code specific to a table, where variables corresponding to table columns are defined

```
RunConditionsProtoDUNE.cxx
RunConditionsProtoDUNE.h
```

- ART service specific to the table

```
RunConditionsServiceProtoDUNE.h
RunConditionsServiceProtoDUNE_service.cc
```

C++ interface - Run Conditions table example

Include header file

```
#include "dunecalib/Calib/RunConditionsProtoDUNE.h"
```

Declare Run Condition table

```
runc::RunConditionsProtoDUNE* runCond = new runc::RunConditionsProtoDUNE();
runCond->SetTableURL("https://dbdata0vm.fnal.gov:9443/dune_runcon_prod/");
runCond->SetTableName("pdunesp.test");
runCond->Update(gRun);
runCond->LoadConditionsT();
```

Just needs 3 inputs:
URL
Table Name
Run/time

Declare row (one per run in this example, use run number value)

```
runc::RunCond_t rc = runCond->GetRunConditions(0);
std::cout << "\tstart time = "  << rc.start_time
          << "\n\tdata type = " << rc.data_type
          << "\n\tRun Number/so w = " << rc.run_number
          << "\n\tupload time =  " << rc.upload_t
          << "\n\tsoftware vers on = " << rc.software_version
          << "\n\tstop_time = "  << rc.stop_time
          << "\n\tbuffer = " << rc.buffer
          << "\n\tac_couple = "  << rc.ac_couple
          << "\n\trun type = "  << rc.run_type << std::endl;
```

Variables ready to use, previously declared at the code specific to the table

C++ interface - Run Conditions table example

Output of c++ script

```
$ getRunConditionsPDUNE -r 23300
```

```
Run Conditions for channel 0:
        start time = 1.70007e+09
        data type = np02_coldbox
        Run Number/sofw = 23300
        upload time = 1.70007e+09
        software version = fd-v4.2.0-c6
        stop_time = 0
        buffer = 0
        ac_couple = 0
        run type = TEST
```

ART service:


Service name
BEGIN_PROLOG
pdune_runconditions :
{
service_provider: "RunConditionsServicePDUNE"
TableURL: "https://dbdata0vm.fnal.gov:9443/dune_runcon_prod/"
TableName: "pdunesp.test"
RunNumber: 23302.0
RunNumber1: 0
DBTag: "v1.1"
Verbosity: 1
}
END_PROLOG
Inputs:
URL
Table Name
Run Number to get data from, if runNumber1 is also specified the in returns data from the interval
Table tag, returns newest if left blank
Verbosity is how much output you get

Examples

- c++ example of how to access the data from the runs condition table
  - /dunecalib/ConInt/getRunConditionsPDUNE.cc

- For the art service
  - /dunecalib/ConIntServices/RunConditionsServicePDUNE_service.cc
  - /dunecalib/ConintServices/runconditions_pdune.fcl

To-Do

- Write documentation on
  - How to create code for new table
    - Once per table
    - Follow template
  - Users
    - How to access data

- Code
  - Null value on int, float variables