INTT dead channel map

❑ Map generator: **calibrations**/Tracking/INTT/macros/**Construct_DeadMap.C**

- Right now: poisson randomly generated dead strip with a fixed mean.
  - 1% dead channels
- https://github.com/sPHENIX-Collaboration/calibrations.git

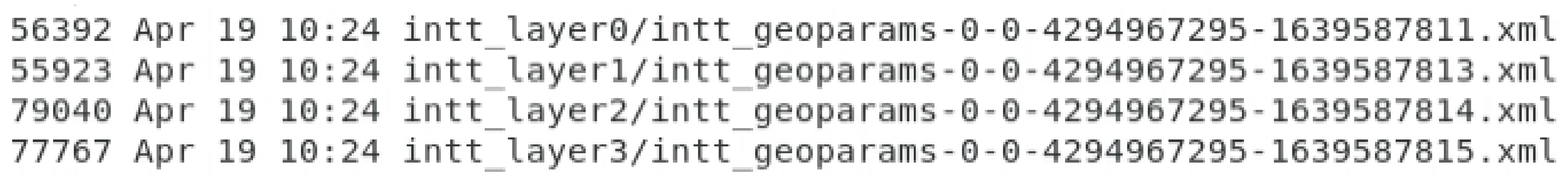
❑ The map is written in XML file with the following format for each layer with values are all set to 1.0.

Form("INTT_%02d_%d_%03d_%03d", ladder_phi, ladder_z, strip_z, strip_phi)

- ladder_phi: sensor index in phi for each layer
- ladder_z: sensor index along Z-axis or each layer
- strip_z, strip_phi: strip index in each individual sensor.

❑ The map is stored at: $CALIBRATIONROOT/Tracking/INTT/DeadMap/

- Automatically loaded when running Fun4All for reconstruction.



```
56392 Apr 19 10:24 intt_layer0/intt_geoparams-0-0-4294967295-1639587811.xml
55923 Apr 19 10:24 intt_layer1/intt_geoparams-0-0-4294967295-1639587813.xml
79040 Apr 19 10:24 intt_layer2/intt_geoparams-0-0-4294967295-1639587814.xml
77767 Apr 19 10:24 intt_layer3/intt_geoparams-0-0-4294967295-1639587815.xml
```

Total size: ~287 KB

❑ Load dead channel map & functions to use it:

- InttDeadMap.cc/.h & InttDeadMapv1.cc/.h & PHG4InttDeadMapLoader.cc/.h
- https://github.com/sPHENIX-Collaboration/coresoftware/tree/master/simulation/g4simulation/g4intt

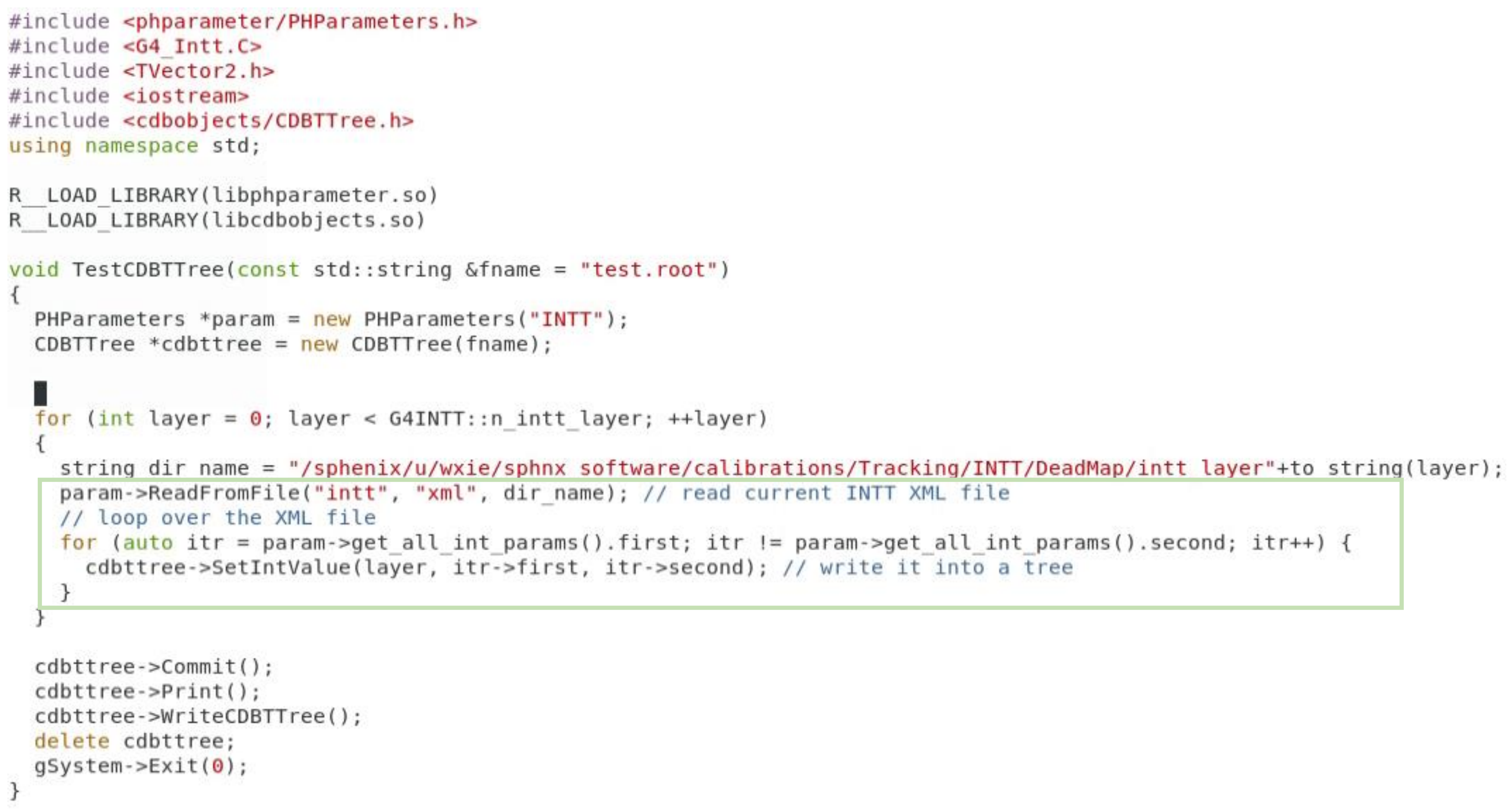
In the last workfest, stumbled on a subsystem presentation using Conditional Database (CDB) to store these maps and then contacted Chris:

that's basically a size question. The cdb just stores files - it doesn't have any requirements about types or content. I would draw the line at a few kB where then switching to binary gives your an order of magnitude space saving. A frightening example is some calorimeter calibration which is 18MB of xml-ascii and about 100kB in root binary.

You jobs currently ends at creating the file and reading it [illegible]ath). The only thing the cdb provides is taking this file and storing it in cvmfs and on request (which comes with a time stamp) returns the full filename (with path). That's still being worked on but if you can write this file and then read it, adding the cdb is easy later.

We have storage based on a ttree or histograms which deals with schema evolution transparently (means you can modify your calibrations while keeping the old file readable), if you spin your own you are responsible to make sure your old calibrations stay readable for the lifetime of sPHENIX (where xml fulfills this, but e.g. saving a handmade TObject does not).

300kB is on the high side. It is possible to play with this using roo[illegible]feeling if we have what is needed. The API is simple - variable name/value and there is the concept of multiple channels:
https://github.com/sPHENIX-Collaboration/macros/tree/master/CDBTest
The caveat is that the lookup is expensive, so one should copy the content of the DB object to your own internal data structures (which you need to do for xml anyway)

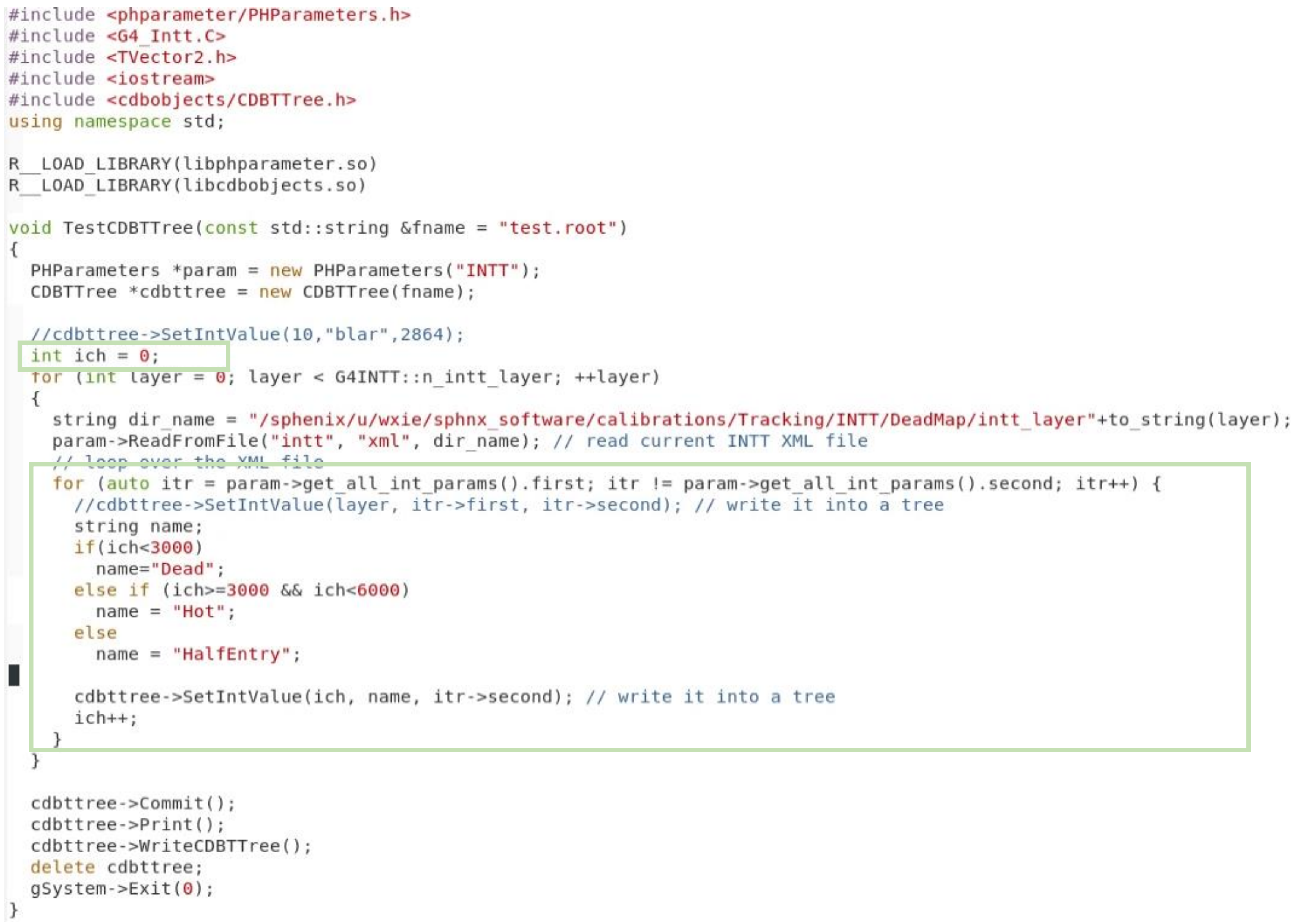


```
#include <phparameter/PHParameters.h>
#include <G4_Intt.C>
#include <TVector2.h>
#include <iostream>
#include <cdbobjects/CDBTTree.h>
using namespace std;

R__LOAD_LIBRARY(libphparameter.so)
R__LOAD_LIBRARY(libcdbobjects.so)

void TestCDBTTree(const std::string &fname = "test.root")
{
  PHParameters *param = new PHParameters("INTT");
  CDBTTree *cdbttree = new CDBTTree(fname);


  for (int layer = 0; layer < G4INTT::n_intt_layer; ++layer)
  {
    string dir_name = "/sphenix/u/wxie/sphnx_software/calibrations/Tracking/INTT/DeadMap/intt_layer"+to_string(layer);
    param->ReadFromFile("intt", "xml", dir_name); // read current INTT XML file
    // loop over the XML file
    for (auto itr = param->get_all_int_params().first; itr != param->get_all_int_params().second; itr++) {
      cdbttree->SetIntValue(layer, itr->first, itr->second); // write it into a tree
    }
  }

  cdbttree->Commit();
  cdbttree->Print();
  cdbttree->WriteCDBTTree();
  delete cdbttree;
  gSystem->Exit(0);
}
```

- ❑ /sphenix/u/wxie/sphnx_software/calibrations/Tracking/INTT/macros/CDBTest/TestCDBTTree.C
- ❑ Output files **in ROOT tree** format (~500 KB), **larger than** in **XML** format (~300 KB).

What is meant by this is that the lookup on your side is slow since it is a map keyed with a string (whose content is saved as a TTree). string compares are just expensive, you don't want to do this thousands of times for every event. On the user end one typically knows how the data looks like and a fixed sized array or so will be much faster.

The problem with the size are your strings - each of which creates a separate branch. You can just look at the root file with a TBrowser. If you reorganize this be e.g. using the layer number in a more efficient way. There seems to be some structure to the strings but no visible pattern. What do they actually mean? They seem to be all over the place, how do you map this onto channels?. In the end you probably want to store the deadmap by channels and not by whatever these strings are. This should reduce the size drastically.



```
#include <phparameter/PHParameters.h>
#include <G4_Intt.C>
#include <TVector2.h>
#include <iostream>
#include <cdbobjects/CDBTTree.h>
using namespace std;

R__LOAD_LIBRARY(libphparameter.so)
R__LOAD_LIBRARY(libcdbobjects.so)

void TestCDBTTree(const std::string &fname = "test.root")
{
  PHParameters *param = new PHParameters("INTT");
  CDBTTree *cdbttree = new CDBTTree(fname);

  //cdbttree->SetIntValue(10,"blar",2864);
  int ich = 0;
  for (int layer = 0; layer < G4INTT::n_intt_layer; ++layer)
  {
    string dir_name = "/sphenix/u/wxie/sphnx_software/calibrations/Tracking/INTT/DeadMap/intt_layer"+to_string(layer);
    param->ReadFromFile("intt", "xml", dir_name); // read current INTT XML file
    // loop over the XML file
    for (auto itr = param->get_all_int_params().first; itr != param->get_all_int_params().second; itr++) {
      //cdbttree->SetIntValue(layer, itr->first, itr->second); // write it into a tree
      string name;
      if(ich<3000)
        name="Dead";
      else if (ich>=3000 && ich<6000)
        name = "Hot";
      else
        name = "HalfEntry";

      cdbttree->SetIntValue(ich, name, itr->second); // write it into a tree
      ich++;
    }
  }

  cdbttree->Commit();
  cdbttree->Print();
  cdbttree->WriteCDBTTree();
  delete cdbttree;
  gSystem->Exit(0);
}
```

- Output files **in ROOT tree** format (~20 KB), **much less than XML** format (~300 KB).

Conclusion

- ❑ We need to assign a unique "**channel index**" for each channel and store it in the CDB.
- ❑ Update the following code:
  - InttDeadMap.cc/.h & InttDeadMapv1.cc/.h & PHG4InttDeadMapLoader.cc/.h
  - https://github.com/sPHENIX-Collaboration/coresoftware/tree/master/simulation/g4simulation/g4intt

  via decoding the "channel index" into the following convention which is being used for offline reco
  - Layer: 0 - 3
  - ladder_phi: sensor index in phi for each layer
  - ladder_z: sensor index along Z-axis or each layer
  - strip_z, strip_phi: strip index in each individual sensor.

- ❑ Cheng-wei and Joseph provide "**channel index"** from hardware convention, i.e. their calibration output of bad channel, as well as the decoder from "channel index" to "layer/phi/z/strip".