Tutorial on the LHCb Conditions Database Framework

Marco Clemencic (CERN-LHCb)
&
Nicolas Gilardi (University of Edinburgh)

Tutorial Overview

- Introduction
- Use Case 1: Algorithm using a condition
- Use Case 2: Creation of a new XML condition
- Use Case 3: Detector Element using a condition
- Use Case 4: Example of the Alignment
- Conclusion

Introduction

Introduction 1/2

- Prerequisites:
  - How to compile and run an LHCb/Gaudi application.
  - How to write XML detector description files.
- Getting the files and environment:
  - Get the Tutorial files: `getpack Tutorial/CondDB v1r0`
  - Get the CondDB browser: `getpack Tools/CondDBUI v0r3`
  - This tutorial should work with any LHCb version based on LCGCMT 42 (official version: LHCb v20r3).

Introduction 2/2

- What you will learn:
  - How to use a condition from an algorithm
  - How to use a condition from a detector element
  - How to create an XML file for a condition
- Additional information:
  - How to browse the CondDB
  - How to edit the CondDB via the CondDBUI
  - How to transfer XML files into the CondDB
  - How to “manually” modify a condition in a function.
- Things we are not going to talk about...
  - Tagging (not yet stable)
  - Single version and multi version folders (not needed for the moment)

Use Case 1

Algorithm Using a Condition

`src/pressureExample.cpp`
`options/pressure.opts`

Algorithm using a condition 1/2

We want to access the LHCb pit pressure from the CondDB, and see it change over events.

- The LHCb structure file in XmlDDDB tells us that the path to the condition `IP8Pressure` in the transient store is:

```
/dd/Conditions/Environment/LHCb/IP8Pressure
```

- We need a function to cache the pressure value for later display:

```
StatusCode PressureExample::i_updateCache ( ) {
  m_LHCb_pressure = m_LHCb_cond->param<double>("IP8Pressure");
  return StatusCode::SUCCESS;
}
```

- At initialisation, we need to tell the Update Manager Service that our `i_updateCache` function requires an up-to-date `IP8Pressure` value:

```
registerCondition("/dd/Conditions/Environment/LHCb/IP8Pressure",
                  m_LHCb_cond,
                  &PressureExample::i_updateCache);
```

Algorithm using a condition 2/2

Now we need to setup our options and environment to access our SQLite CondDB in `${CONDDBROOT}/db/CondDB.sqlite`

- Our `pressure.opts` file must include the `cool.opts` file:

```
// Dynamically load component library for DetCond services and converters
ApplicationMgr.DLLs += { "DetCond" };
// Detector Persistency service setup:
DetectorPersistencySvc.CnvServices += { "CondDBCnvSvc" };
// Connection string to access to the CondDB
CondDBAccessSvc.ConnectionString = "$CONDDBCONNECTIONSTRING";
```

- In our `requirements` file, we have set `$CONDDBCONNECTIONSTRING` to:

```
"sqlite://none;user=none;password=none;schema=${CONDDBROOT}/db/CondDB.sqlite;dbname=DDDB"
```

- We need to set the environment variable `$CONDITIONS_PATH` to:

```
conddb:/Conditions/conditions.xml
```

- Now we can run our algorithm and follow the evolution of the pressure.

Use Case 2

Creation of a New XML Condition

`src/dummyConditionExample.cpp`
`options/dummyCondition.opts`

Creation of a new XML condition 1/5

We want to create a new XML condition from scratch and use it in an algorithm.

- We want to put our condition under the following transient store path:

```
/dd/Conditions/Environment/DummyDE/DummyCondition
```

- It will contain one integer parameter called `DummyValue`:

```
<condition name="DummyCondition">
  <param name = "DummyValue" type = "int"> 41 </param>
</condition>
```

- It will be referenced by the `XmlConditions/DDDB/Local/DummyDE/environment.xml` file:

```
<catalog name="DummyDE">
    <conditionref href = "dummyCondition.xml#DummyCondition"/>
</catalog>
```

- We need to update the file XmlConditions/DDDB/conditions.xml:

```
<catalog name="Environment">
  <catalogref href = "Local/LHCb/environment.xml#LHCb"/>
  <catalogref href = "Local/DummyDE/environment.xml#DummyDE"/>
</catalog>
```

Creation of a new XML condition 2/5

Now, we need a simple modification of our algorithm and environment to be able to retrieve the new condition.

- The intialisation function of the algorithm now contains:

```
registerCondition("/dd/Conditions/Environment/DummyDE/DummyCondition",
                  m_DummyCond,
                  &DummyConditionExample::i_updateCache);
```

- To use the XmlConditions files, we need to set `$CONDITIONS_PATH` to:

```
${CONDDBROOT}/XmlConditions/DDDB/conditions.xml
```

- Now if we run the algorithm, we can see the value of the dummy condition.
- Of course, the value is constant over the events: we are getting it from an XML file !

Creation of a new XML condition 3/5

If we want to use time varying value for this condition, we first need to put it in the CondDB

- The CondDB data are XML strings. Thus there is almost nothing to change in the condition file.
- A Python script is provided. It allows to move an XML file into the CondDB, under a given node. It does automatically the necessary modifications to the XML string before writing them in the database.
- Here is the command putting `DummyConditions` into an existing `CondDB.sqlite`:

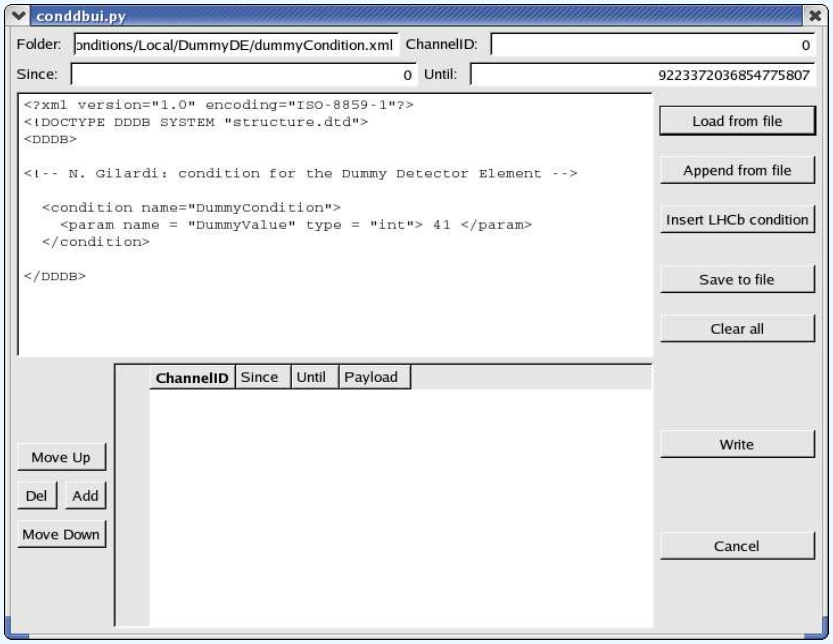
```
python ${CONDDBROOT}/python/copy_files_to_db.py
     --connect-string “sqlite://none;schema=${CONDDBROOT}/db/CondDB.sqlite;dbname=DDDB”
     --source ${CONDDBROOT}/XmlConditions/DDDB/Local/DummyDE
     --destination /Conditions/Local/DummyDE
     --keep-db
```

- You can verify that everything is in place by opening the CondDB with the CondDBUI.

Creation of a new XML condition 4/5

We can now use the CondDBUI to modify the IOV and values of the DummyCondition.

- In the CondDB tree, select the channel 0 of /Conditions/Local/DummyDE
- In the Data display on the left, select the payload of the condition object
- Now, from the Edit menu, select “Add Condition”. This will open a dialog which looks like this:



Creation of a new XML condition 5/5

- Set the “Since” and “Until” values to 10 and 20, and in the text editor, change the DummyValue to any integer value you want. When done, press the “Add” button on the bottom left pad.
- Repeat the operation any time you want (just try to create contiguous intervals of validity for the moment).
- When you are satisfied with the list of data you have, press the button “Write” on the bottom right. This will automatically update the CondDB. You can close the CondDBUI.
- Now, if we set `$CONDITIONS_PATH` to `conddb:/Conditions/conditions.xml` and run again our algorithm, we should see the DummyValue change over time.

Use Case 3

Detector Element Using a Condition

`src/dummyDEExample.cpp`
`options/dummyDE.opts`

Detector Element using a condition 1/2

We want a detector element to use a condition instead of a parameter.

- The transient store path to the detector element is `/dd/Structure/LHCb/DummyDE`
- Its definition is in `${CONDDBROOT}/XmlDDDB/DDDB/DummyDE/structure.xml` :

```
<detelem classID="6669999" name="DummyDE">
  <param name="comment" type="string"> Dummy detector element for the Tutorial </param>
  <conditioninfo name="DummyCondition"
                 condition="/dd/Conditions/Environment/DummyDE/DummyCondition"/>
</detelem>
```

- DummyDE will simply cache the DummyValue into private variable. A private function will be in charge of doing this:

```
StatusCode DummyDetectorElement::i_updateDummyValue(){
  m_dummyValue = condition("DummyCondition")->param<int>("DummyValue");
  return StatusCode::SUCCESS;
}
```

Detector Element using a condition 2/2

- DummyDE's `initialize` contains the registration to the update manager:

```
updMgrSvc()->registerCondition(this, condition("DummyCondition").path(),
                               &DummyDetectorElement::i_updateDummyValue);
sc = updMgrSvc()->update(this);
```

- The algorithm using DummyDE will ask it to return `m_dummyValue`:

```
StatusCode DummyDEExample::execute() {
  ++m_evtCount;
  info() << "Dummy Condition Value: " << m_dummy->getDummyValue() << endmsg;
  return StatusCode::SUCCESS;
}
```

- The algorithm doesn't need to register anything to the update manager:

```
StatusCode DummyDEExample::initialize() {
  ...
  m_dummy = getDet<DummyDetectorElement>( "/dd/Structure/LHCb/DummyDE" );
  ...
}
```

- Now we can set `$CONDITIONS_PATH` to use either the XML files or the CondDB, and we will get similar output as for the previous use case.

Use Case 4

Example of the Alignment

`src/alignmentExample.cpp`
`options/alignment.opts`

Example of the Alignment 1/2

The alignment condition is available for all detector elements and automatically managed by the geometry framework. Here, we simply show how a modification of the alignment of the Velo System is automatically propagated to its sub-elements.

- The algorithm is observing 3 detector elements: `LHCb`, `Velo`, and `VeloLeft`:

```
m_lhcb     =getDet<DetectorElement>("/dd/Structure/LHCb");
m_velo     =getDet<DetectorElement>("/dd/Structure/LHCb/BeforeMagnetRegion/Velo");
m_velo_left=getDet<DetectorElement>("/dd/Structure/LHCb/BeforeMagnetRegion/Velo/VeloLeft");
```

- It gets also the alignment condition of the `Velo` detector element:

```
m_velo_align=getDet<AlignmentCondition>("/dd/Conditions/Alignment/Velo/VeloSystem");
```

Now, thanks to the geometry framework, if we modify the `m_velo_align` condition, it will be propagated to child detector elements of `Velo` (`VeloLeft`), but the alignment of the parent detector element (`LHCb`) will remain unmodified.

Example of the Alignment 2/2

- We modify the alignment of the Velo detector element:

```
std::vector<double> nothing(3);
std::vector<double> translation(3);
translation[0] = m_evtCount * 1.0;
translation[1] = m_evtCount + 3.0;
translation[2] = m_evtCount * -2.0;
m_velo_align->setTransformation(translation, nothing, nothing);
```

- Now, as this modification is done "manually" (i.e. not via the automatic validity checking procedure), we need to tell the update manager about it:

```
updMgrSvc()->invalidate(m_velo_align);
```

- The update manager is now aware of the change and will propagate the information before the next event.
- If we now set the `$CONDITIONS_PATH` to use XML files, we can see how the misalignment of the `Velo` is propagated.

Conclusion

Conclusion

- We expect these use cases to cover 90% of your needs.
- The other source of information is the Detector Conditions web page:
  - http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/DetCond
- For more “advanced” needs, please ask us directly:
  - marco.clemencic@cern.ch (general CondDB framework)
  - ngilardi@ph.ed.ac.uk (tutorial and CondDBUI)
  - juan.palacios@cern.ch (geometry framework)

Finally, please keep in mind that COOL will experience many changes in the coming months. It will have minor effects (if any) on the LHCb framework, but backward compatibility of the databases will be an issue !