

# LHCb Conditions Database Graphical User Interface

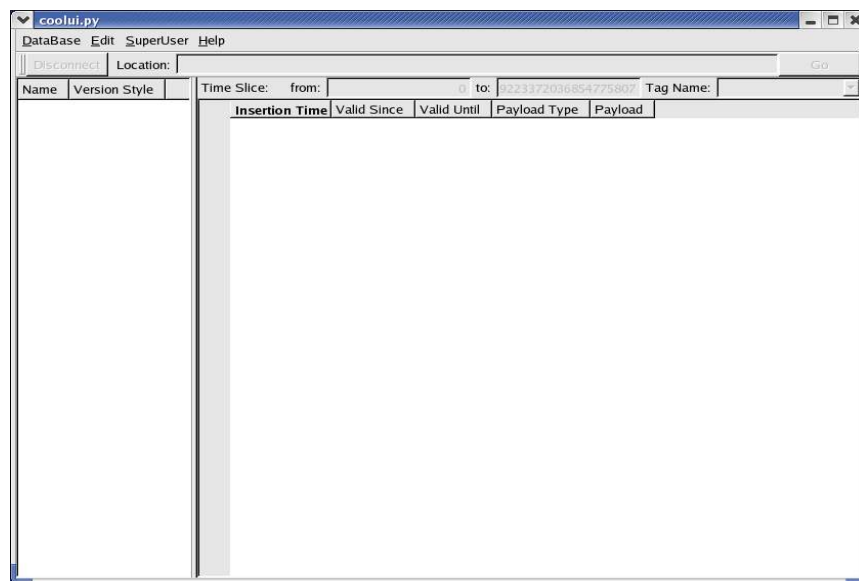
*v0r3*

## Introduction

This document is a description of the current features of the “coolui” program which allows to browse and edit a conditions database. It is written in pure Python and based on PyCool, the python binding for COOL, and PyQt, the python binding for the graphical toolkit Qt.

The features of this browser are often “LHCb oriented”. However, being coded in Python, with absolutely no dependencies on Gaudi/LHCb code, it is easy to adapt to other needs.

## The Main Window



The main window is separated into 4 main areas: the menu bar (top), the location bar (under the menu), the condDB tree (left) and the condition object table (right).

The condition objects table itself is separated into 3 areas: the time slice and Tag selection, the Condition Objects list, and the payload representation (which is invisible by default).

## The menu bar

There are 4 menus available, but some features are not yet active, and some are restricted on purpose.

## Database menu

This menu concern all actions taken at the level of the full database.

- Open: to open an existing database or create a new one
- Slice: copy a slice of the current CondDB to another one (disabled)
- Close: close the connection to the active CondDB
- Quit: quit the program.

## Edit menu

In this menu, action are taken to edit the contents of the database.

- New Folder: creates a new node (folder or folderset) in the CondDB
- Add Condition: add a new condition object to folder
- Tag: give a tag to the current HEAD version

## SuperUser menu

This menu is not active by default because it is a very sensitive one: data can be destroyed using it and errors are not recoverable.

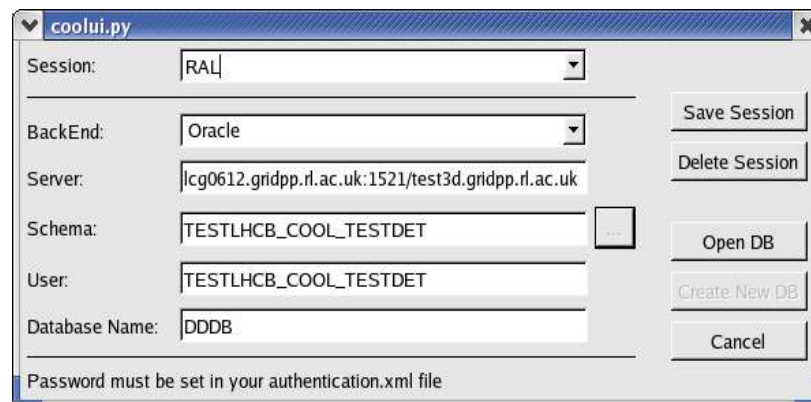
- Delete Folder: delete a node (folder or folderset) from the CondDB
- Delete Tag: delete an existing tag
- Delete Database: delete a database (not implemented)

## Help menu

This menu does not contain much for the moment ;-)

- About: info about the current version of the browser.

## Opening a Database



The screenshot shows a dialog box titled "coolui.py" with the following fields and buttons:

- Session: RAL
- BackEnd: Oracle
- Server: lcg0612.gridpp.rl.ac.uk:1521/test3d.gridpp.rl.ac.uk
- Schema: TESTLHCB\_COOL\_TESTDET
- User: TESTLHCB\_COOL\_TESTDET
- Database Name: DDDB

Buttons on the right side: Save Session, Delete Session, Open DB, Create New DB, Cancel.

Bottom text: Password must be set in your authentication.xml file

The first action to take after starting the browser is to open an existing database. This action is accessible from the menu Database/Open.

The dialog window will ask for some connection details, starting by the backend of the database (available are Oracle, MySQL and SQLite).

Depending on the backend chosen, some connection parameters may be required or not. In the case of SQLite, no authentication is required. You just need to specify the schema (which is a filename; the “...” button allows you to browse the file system) and the database name.

In the case of connection to DB servers (Oracle or MySQL), you will need a server name, a user name and a password. For security reasons, the password can't be given in the dialog window. You have to define a POOL\_AUTH\_PATH environment variable pointing to the “authentication.xml” file. A basic of authentication file will look like this:

```
<?xml version="1.0" ?>
<connectionlist>
  <connection name="mysql://myserver.cern.ch/MYSCHEMA">
    <parameter name="user" value="myname" />
    <parameter name="password" value="myPaSw0rd" />
  </connection>
</connectionlist>
```

Once the required parameters are set, you can press “Open DB” to load the contents in the browser. Building the CondDB tree can take some time, depending on the size of the DB and the network overhead.

In order to gain some time, it is possible to save the current content of the fields under a specific name, by editing the “Session” field and pressing “Save Session”. This will save the current set of parameters in a file called “sessions.dbm”. They can be reloaded simply by selecting the session name from the “Session” list.

To delete the selected session from the persistent file, simply press “Delete Session”. This does not affect the CondDB referenced by the session.

When the correct parameters are given, pressing “Open DB” will build the CondDB tree and browsing can start.

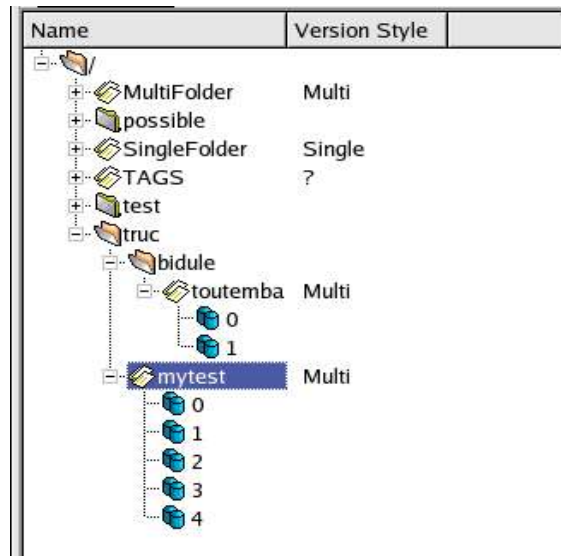
It is also possible to create a new CondDB from scratch (i.e. containing only the root folderset “/”). This is however subject to two limitations:

- you can do this only if the backend is SQLite;
- the program is not allowed to override existing databases. If you want to use an SQLite file name already in use, you will have first to delete it manually.

If these constraints are fulfilled, pressing “Create New DB” will create a new SQLite file and open an empty CondDB.

## Browsing the Database

### The CondDB Tree



Name	Version Style
MultiFolder	Multi
possible	
SingleFolder	Single
TAGS	?
test	
truc	
bidule	
toutemba	Multi
0	
1	
mytest	Multi
0	
1	
2	
3	
4	

The data inside the CondDB are stored in a hierarchical way. Foldersets contains other foldersets and folders. Folders contain channels, and channels are the containers of the Condition Objects. Each element of the CondDB is accessible by a name which is similar to a file system name (i.e. of the form “/my/file/name”).

The browser is thus providing a tree representation of the contents of the CondDB with three type of elements: folderset, folders and channels. Conditions objects are not represented in the tree. Navigation in the tree is standard, i.e. nodes can be opened/closed by double clicking then or pressing their “+” or “-” icons.

Folders can be of two types: single version or multi version. This information is provided in the tree by the column “Version Style”. However, for performance reason, it is not possible to know this version style the first time the tree is constructed. To know the version style of a folder, you just need to select it once, and the “?” will be replaced by the information you need.

### The Location Bar

It is also possible to browse the tree using the location bar (which is just under the menu). By default, it shows the path to the current element selected in the tree. But it can work the other way around: typing a path and pressing “Go” or the return key will automatically select the corresponding element in the tree.

**WARNING:** it is not possible to access to a channel via the location bar if this channel hasn't been loaded before. This should be fixed in future version.

## The Condition Objects Table

When a channel is selected in the CondDB Tree, the Condition Objects it contains are displayed on the right, in the Condition Objects Table.

	Insertion Time	Valid Since	Valid Until	Payload Type	Payload	
1	2006.01.19; 17:55:41	0	10	string	0 -> 10	
2	2006.01.19; 17:55:41	10	100	string	10 -> 100	
3	2006.01.19; 17:55:41	100	1000	string	100 -> 1000	
4	2006.01.19; 17:55:41	1000	10000	string	1000 -> 10000	
5	2006.01.19; 17:55:41	10000	9223372036854775807	string	Click to Display	

```
<rien>
  nothing
</rien>
```

The table displays the list of conditions objects which are valid for the selected time slice (defined by the fields “from:” and “to:”), and, in case of a multiversion folder, for a given tag (chosen in the list “Tag Name:”).

The information retrieved consists of the object insertion time, the lower bound (“since”) and upper bound (“until”) of the Interval Of Validity (IOV), the payload type, and the payload itself. If this payload can be represented in string of one line and less than 20 characters, then it will be shown in the “payload” column. Otherwise, the string “Click to Display” will appear instead. Clicking on this cell will reveal the payload display, at the bottom of the table, with a text representation of the payload.

The condition objects table is read only, except for the time slice and the tag selection. Actually, it is not possible to modify the contents of a condition object when it is stored in the CondDB. This is the reason why adding a new object has to be done very carefully.

## Changing the Time Slice

Changing the time slice is trivial: you just need to give whatever integer value you want in the “from:” and “to:” fields, then press return and see only those condition objects valid for this period of time. Please note that unlike the condition object's IOV, where the “until” time is excluded from the validity, the “to:” time is included in the

time slice. For example, if you have three objects A, B and C, such that their IOVs are respectively [0,10[, [10, 100[ and [100, 1000[, asking for a time slice from 0 to 10 will show A and B as valid objects.

There are some “experimental” protection concerning what users can put as parameters of the time slice. For instance, it is not possible to put non numerical characters (including “-” and “.”: validity keys are positive integers). It is also impossible to give an empty value and this will be automatically replaced by a “0”. Finally, it is not possible to give a value greater than the ValidityKeyMax (which is 9223372036854775807). Trying to do so will replace the value by ValidityKeyMax (this is quite useful if you need to enter this value and don't really want to remember the 19 digits: simply try to put 99999999999999999999 ;-)).

## Changing the Tag

COOL's tagging policy is not stable yet. This implies that the current system is very likely to be obsolete in the future versions of the browser. However, for the time being it works.

Each time you select a folder or a channel in the CondDB tree, the list of available tags is loaded to the condition objects table. By selecting a specific tag in the list, you will only display the objects which appear in this version of the CondDB. This adds up with the time slice. You can thus select the objects under tag “PROD” which are valid between  $t = 100$  and  $t = 50000$ .

Sometimes, the tag names will include the path to the folder where the selected channel is stored. This is an LHCb trick to solve the issue of COOL missing global tagging (i.e. tags at the level of the CondDB, as used by LHCb. Current COOL tags are defined at the level of the folders).

## Changing time slice and tags at the level of folders

So far, we've seen that it was possible to change tags and time slice when a channel was selected in the CondDB tree. However, when a channel contains a huge number of condition objects, it can take a lot of time to load them all. And as the default time slice and tag are not necessary what you are interested in, this is a *waste* of time.

To solve this problem, it is actually possible to set the time slice and chose the tag when a folder is selected. Selecting a folder does not load the condition objects, so you can set the selection you want and then select the channel you are interested in. The only condition objects that will be loaded will be the valid ones.

## About the HEAD

As a reminder, the “HEAD” tag is absolutely irrelevant in LHCb. In practice, it should never be used in an application as it may contain condition objects that have  $t=$ nothing to do with one another.

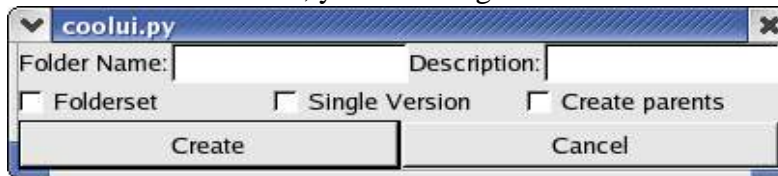
For the moment, it is the default tag of the browser, but this will change in the future (as soon as a “production” tag will be define).

## Editing the Database

Three actions can be taken to edit the CondDB: create a new folder or folderset, add new condition objects and tag the current HEAD.

### Creating a New Folder

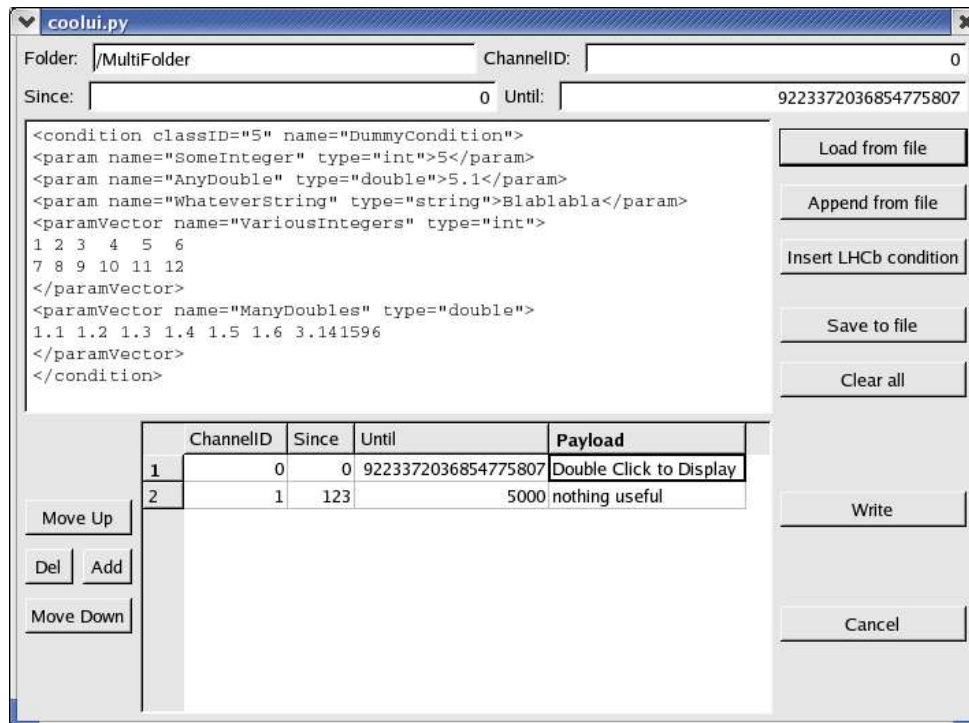
To create a new folder or folderset, you have to go to the menu Edit/New Folder.



The folder name is the full path of the new folder. If the parents don't exist, you can check the “Create Parents” option to do it automatically. By default, a multi version folder is created. If you want to create a single version folder or a folderset, you just have to check the relevant options. When you are done, simply press “Create” and if everything goes well, the new folder/folderset will appear in the CondDB tree.

### Adding Condition Objects

This is accessible via the menu Edit/Add Condition. You need to select a folder to be able to access the editor.



This dialog window is much more complex than the other ones. From top to bottom, we have:

- the name of the folder which will host the objects
- the channel ID of the object to add
- the IOV of the object to add
- a text editor for the payload (this editor only accepts string payloads)
- the list of condition objects that are ready to be written to the CondDB.

### **Folder Name and Channel ID**

The folder parameter is not editable. It corresponds to the folder selected in the CondDB tree when the editor was opened.

The channel ID is simply a positive integer defining (or referring to) a channel. A channel can only accept condition objects sharing the same payload type. For LHCb, this is irrelevant as all our payloads are strings. However, we can use it as another level of sorting for our data.

### **The Interval of Validity**

The IOV limits (since and until) are restricted exactly in the same way as previously explained for the time slice definition (i.e. positive integers smaller than ValidityKeyMax). Keep in mind that the “until” time is excluded of the IOV.

### **The payload Editor**

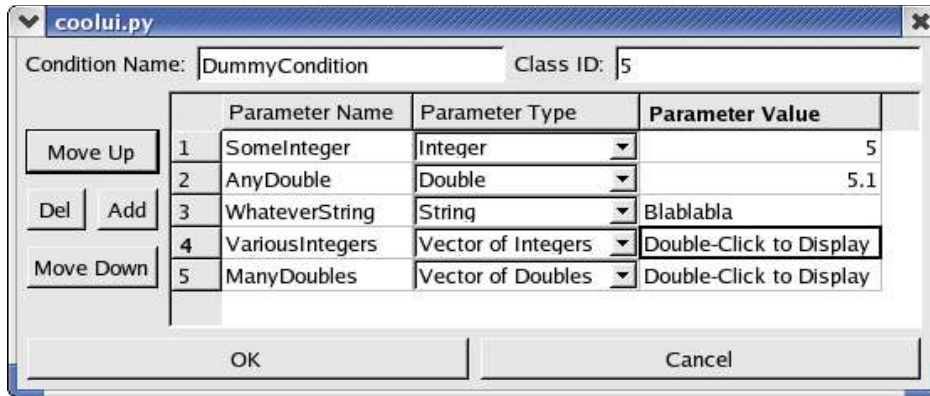
The payload editor is basically a text editor where you can put anything you want. You can also load a new text from a file (via the button “Load from file”). This will discard previously typed text. To append some saved text to the already typed one, click on “Append from file”. To save text to a file for further use, click “Save to file”. And to clear the whole typed text, simply click “Clear All”.

### **Inserting LHCb conditions**

The special button “Insert LHCb condition” opens another dialog which will generate an XML definition of your payload, conform to LHCb rules.

In this editor, you simply chose a name for the condition, a class ID (usually, 5), and then you add parameters to it. When you click the “Add” button, a new parameter entry is created. You must give it a name, a type (one among five), and a value. If the type is a vector of integers or doubles, a very simple text editor will open when you double click on the “parameter value” cell. This allows to do a bit of text formatting.





You can change the order in which the parameters will appear in the XML using the “Move Up” and “Move Down” buttons. You can even remove a parameter by selecting one of its cells and pressing “Del”. When you are done, press “OK” and the XML is generated and appended to the text in the main condition object editor window.

### The Condition objects list

The last step is to add the object to the list that will be written to the CondDB. This is done by pressing the “Add” button on the bottom left pad. This will add a new entry in the list based on what is currently set in the editor. The parameters are summarised in the various columns of the list.

If you want to add another object, simply modify the parameters and click “Add” again. And if you are not satisfied with an object, you can remove it by selecting one of its cells and press “Del”. This way, you can also “modify” an entry: double click on its payload to display it in the text editor, modify what you want, add the new object, and delete the old one (this odd procedure will be simplified in future versions).

The last thing you have to take into account, is the order in which the objects will be inserted in the CondDB. Imagine you add two objects A and B in channel 0, and their IOVs are overlapping: for instance A is valid for [0, 100[ and B for [50, 200[. If A is written before B, the result will show A valid for [0, 50[ and B for [50, 200[. If B is inserted before A, then the result will show A valid for [0, 100[ and B for [100, 200[.

For this reason, it is possible to change the order in which objects appear in the list, using buttons “Move Up” and “Move Down”. The objects on top of the list are inserted first.

### Last warning

When an object is added to the CondDB, there is no way to remove it. The only solution to make it invisible is to insert another object on top of it (in term of IOV). For this reason, you have to be very careful when adding a new object in the CondDB, especially if it has a big payload.

## Tagging

This is done via the menu Edit/Tag.



Creating a new tag will associate a name to the data currently visible in the HEAD version of the CondDB. Most of the time, a new tag is applied immediately after the insertion of some new condition objects. Once a tag is applied, the data it refers to can't be changed (this is not the case for the HEAD, for example).

In the COOL API, tags can only be associated to folders and must be unique in the whole database. This is impractical for LHCb use cases where we often want to tag a state of the full database. To solve this problem while waiting for this feature, the “LHCb Global Tag” option is creating a “/TAGS” folder in the CondDB which will store the names of the global tags (for example “PROD”), and then tags every folder of the CondDB with a tag named <folderPath>-<tagName> (for example: /myFolder-PROD). This option is the one available by default in the browser when you want to tag the current HEAD. The main problem is that for large CondDBs, it can be very slow.

The other possibility for tagging is the COOL way. If you uncheck the “LHCb Global Tag” option, you can specify the path to the folder you want to tag (by default, it is the one selected in the CondDB tree, but you can modify it).