LHCb Configuration Database Visualizer

LHCB Technical Note

Issue: v3r2

Created: 20th August 2007
Last modified: 31 August 2007
Prepared By: Thomas Johansen, Lana Abadie, Eric van Herwijnen, Robbie Shade, Walid Belballi

Abstract

The aim of this note is to give details about porting the CDBVis program to Qt. It gives instruction to install Qt and explain the main modification made to CDBVis.
The LHCB note of the version 2.7 of CDBVis gives more details about the application before porting it.

Table of contents

LHCB Technical Note
Issue: v3r2
Abstract
Table of contents

Installation

1. Program requirements

The program was developed and tested using the following versions of external software/libraries:

Requirements

The program should in theory run smoothly with:

- Python 2.4
- Qt 3
- ConfDB Library version >3.5
- (AFS Client so that the Boost library can be found on the CERN network.)

Python 2.4 is used due to compatibility issues with the ConfDB Library Python wrapper.

2. Installing on Windows (NICE)

a. Installing Python

You can find the recommended stable versions of Python here: http://www.python.org/download/.

To check the version of Python you are running, run the following command on the command line:

```
python -V
```

If you get command not found, Python is not installed on your system. For Windows you will need Python version 2.4, due to compatibility issues with ConfDBLibrary.

It is possible to run two or several versions of Python in parallel, but not recommended unless you know what you're doing. You then download the Python executable from the website, and execute the *.msi or *.exe file to install it. Choose a directory to install it in, and remember this directory. You will have to add the path to the directory to the Path environment variable. This is done by right-click My Computer on your desktop, choose properties, choose the advanced tab and click on the button that says “Environment Variables”. In the system variable window, you look for the Path variable, choose to edit, and add the path to the directory where you installed Python at the back of the string. If the string that was already set for the path did not end with a ‘;’ (semicolon), you will have to add that to the string before you add (append) your new path. Click Ok all the way back, to close the windows. You will have to close all open command line windows to make the change take effect (but in some cases you will have to log off and on again).

To check if Python was successfully installed, open a new command line window, and run python –V again to see that your python executable echoes the new version number. If the version number is incorrect it is most likely because you have another version of Python installed on your system that is executed instead. To deal with this; change the name of one of the Python executables (found in the Python root directory), and execute the command:

<name_of_my_python_executable> -V

To see that it is the correct version. Now you will have to use this name instead of Python when you are running python scripts.

b. Installing Qt 3

Due to compatibility issues with ConfDBLibrary we cannot use Python 2.5. While Qt 4 can only be used with python 2.5 we are bounded to use Qt v3.
Qt is available in a free edition for WINDOWS only since the version 4. So we will use a native win32 port of the Qt/x11 (under GPL) sources which use native win32 api and does not require cygwin.
The installation instructions for Qt are available on this page:
http://qtwin.sourceforge.net/qt3-win32/index.php
In this page you will have to choose how you would compile the Qt sources (using MinGW or MS Visual Studio …) then you will find the corresponding instructions.
We have installed Qt using MinGW and we have not encountered any problem compiling the sources.
You are also recommended to download the documentation and Qt demos from the same webpage.

c. Installing PyQt

Qt is a C++ library and we cannot use it directly with python. You will need to Install PyQt which is a set of Python bindings for Trolltech's Qt application.
Before you can build PyQt from source you must have already built and installed SIP which is a tool that makes it easy to create Python bindings for C++.
Instructions to install SIP then PyQt are available on this page:
http://kscraft.sourceforge.net/pyqt-windows-install.xhtml
You can find documentation about SIP and PyQt on:
http://www.riverbankcomputing.co.uk

d. Installing CdbVis

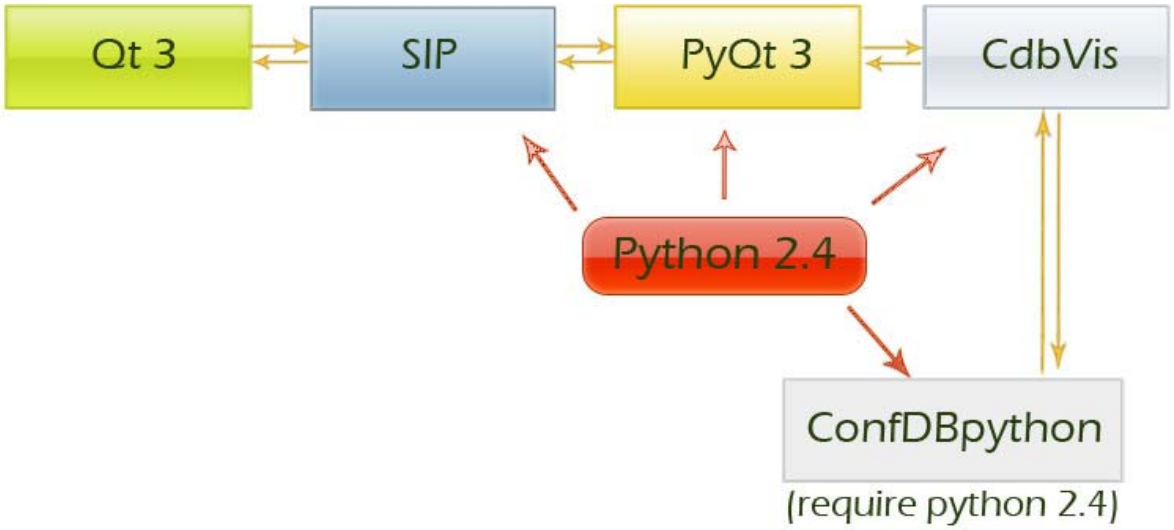
This will be downloadable from the web in a zip file to be extracted to an arbitrary directory on the user's hard drive.

e. Installing Oracle Instant Client 10g

This library is needed for the Oracle database connection used by ConfDB library.
Installation instructions for Windows can be found here: http://lhcb-online.web.cern.ch/lhcb-online/configurationdb/APIusage.htm
2.5 Installing ConfDB library
This is the library used by CdbVis to communicate with the ConfDB. Download the Python Interface library for Windows from: http://lhcb-online.web.cern.ch/lhcb-online/configurationdb/APIusage.htm. The two *.dll and the two *.lib files should be extracted to the installation directory of CdbVis.
2.6 Installing Python modules
You will need to install the cx_Oracle module that allows access to Oracle databases. The confDB library already access to the database but we need the cx_Oracle module to do some actions on the database which are not yet included in the API (dealing with the URL table).

Installation instructions are available on this page:
http://www.python.net/crew/atuining/cx_Oracle/
For WINDOWS you just have to download and execute the latest release of the installer for Python 2.4.

The scheme below shows how the component required for CDBVis are connected to each other.



Testing Installation

If CdbVis is “not working” after you have installed all of its dependencies, here are some tests you should do to see what that fails. To make it easier to debug possible errors, we take one step at a time.

a. Test Python

Run the following command:

```
Python
```

You'll enter Python Shell, type:

```
print “hello world”
```

and press enter. If it echoes “hello world” back to you, then it works. Press Ctrl-Z to exit the shell.

b. Test Qt:

Run Python then enter:

```
Import qt
```

If this command succeeds without any error you have installed successfully Qt. otherwise try to install qt again.

c. Test CdbVis

Go to the directory where you installed CdbVis and run the command:

```
python main.py
```
If it runs, you have done things correct so far, the ConfDB library files have been found as well. If it fails, it is probably because the ConfDB library files were not found, see through the installation of the ConfDB library as described above to see if you missed something.
If *.py files is associated with Python you can actually just double-click on main.py and it will start. (If the *.py files have a icon of a python snake).

d. Test ConfDB Connection

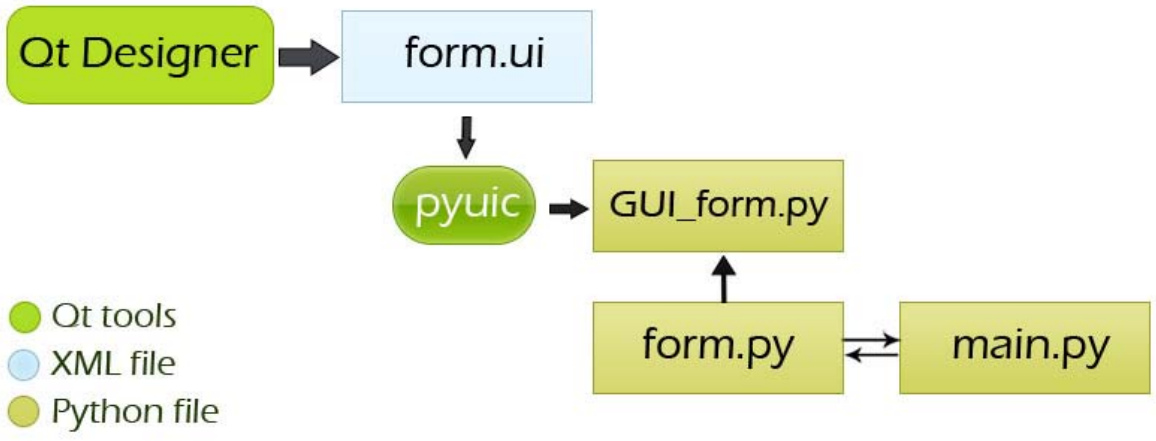
Once you have started CdbVis, click on the button which shows a lightning, or choose from the menu: File > Connect, to connect to the ConfDB. If it fails, the error will probably be reported either through CdbVis, or in the command line shell window. Errors here should be directed to Lana Abadie, who can most likely tell you why the database connection failed, and help you out.

The version using Qt and python 2.4 of CDBVis was developed on windows. We have not tested the application on Linux because we failed compiling the ConfDBpython on linux for python 2.4 due to some linking problems.

Description of the main modifications made

Separating GUI part of the code from the non-GUI part

The process of development consists in the following:



First we use the Qt Desginer to make the visual interface we need, we save then our interface as an XML file with the extention ".ui". Then, using the User Inteface Compiler for python (pyuic) we generate the phyton class of our interface.
I used to prefix the name of the files containing the classes generated by pyuic by "GUI". Then we make our class inherit from the GUI one before we can use it in our main application.
This separation between the generated part of the code and the non-generated one can be useful in the way that we can use the Designer to modify the visual part of our interface. Then we will just have to generate a new GUI_form.py file which will erase the first one.

This process can be applied to all the files having the prefix “GUI” except for GUImainWindow.py and GUImainPanel.py which have been modified after they have been generated by the pyuic.
I ported the version 2.7 of CDBVis which was using wxPyhton library. The sources of this version can be found the web page of the configuration database visualizer. This version was suffering of many problems; many features were not working properly which cause some modifications in the version 3 of the program. The work is far from to be finished and many features have to be added.

URL Table

The VELO group needed to associate a web link to every device to connect the configuration database to their test database located in Liverpool.
The best solution was to add a URL column in the table of functional devices, but modifying the table schema needs to modify the API (ConfDB Lib). So I added a new table containing two columns (name of the device and its URL). I also modified the save function of the Device class to include the changes of the web link in the database.
Those modifications require connecting to the database using a different way than the API of the Conf DB. That’s why we used the cx_Oracle module to connect to the Database.

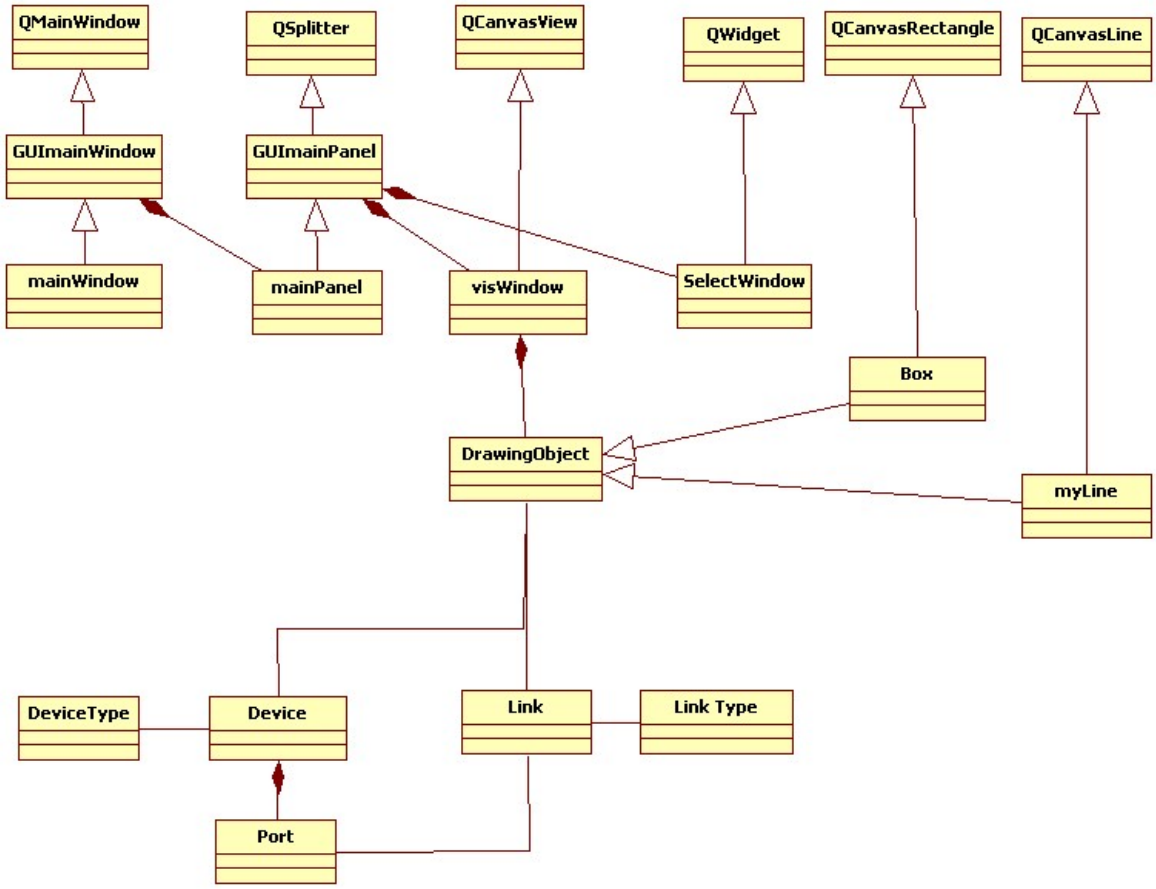
CDBVis Modes

In version 2.7 of CDBVis using wxPython, user can switch from the navigation Mode to the Creation one using buttons on the tools bar.
In the version 2.3 this is done automatically. Creation mode is considered differently as in the version 2.7; the features of creating/modifying devices are available directly from the navigation mode and since the user choose to modify or create a device, the application switch automatically to the creation mode. This is a way to alert the user that modifications have been done and he should commit them to the database using the commit button or a python script.
One the modifications are committed the program switch to navigation mode again.

Spare devices

In the version 2.7 of CDBVis, spare devices of all the systems were grouped in the same part of the selection tree. It would be better to separate the Spare devices of every system. This still have to be done in the version 3 of CDBVis.

UML diagram



The class model above show the relations between the different classes of the program.