Abstract
Due to the massive amounts of components in a huge detector like CMS storing information such as status of this different parts in databases as dps and dpes leads to quite complicated structures. Because of this complexity extracting data from configurations of data point elements contained in a data point type with the most general construction in not as easy as it seems and there is no appropriate tool in WinCCOA for this purpose. The fist step for this extraction is having the list of all dpes in a an accessible format to search for the existence of desired configurations and starting the extraction afterwards. In this project we wrote down a script to make the mentioned list.

Ali Karimi

Supervisor: Wassef Karimeh

August, 2022
1 Introduction

1.1 CMS

The CMS detector is a general purpose detector designed to access a broad range of physics topics over the full range of luminosities provided at the LHC. It is the second biggest one, and gets its name from the detector, which is built around a huge solenoid magnet. Another key feature for the name was its size, as compared to the ATLAS experiment, which is the biggest experiment, it is very compact, with it only being 21 m long, 15 m high and 15 m wide.

1.2 DCS

The Detector Control System (DCS) is responsible of ensuring the safe and optimal operation of the experiment so that high quality physics data can be recorded by the data acquisition system. The DCS input data rate is in the range of $10^4\text{MB/s}$. This data is not only stored to a database but it also needs to be processed at this rate in order to take automatic decisions and sequence commands that are sent to different parts of the system.

1.3 WinCCOA

WinCC OA is a supervisory control and data acquisition (SCADA) software. It is used to monitor and control very large distributed processes. It has been in use at CERN since the year 2000, after a 3-year discussion period.

WinCC OA is data point based, provides a scalable graphic user interface and can support redundant and distributed systems. This latter means, that if the program is run in a certain system, it is possible to retrieve and monitor the data from all its connected subsystems.

1.3.1 CTRL

A Control program (“script”) is used to control WinCC OA, and can be programmed by the actual user of the control system. In this way the user is employing a script to specify the response of WinCC OA to an event. The syntax of the internal controller language “Control” has a similar structure to that of the C programming language.

The Control program is capable of multi-program operation (multi-threading). Blocks can be defined in the CTRL script that are processed when a specific event occurs (for example, message that a data point attribute has changed). Since WinCC OA can process more than one data point at once, the Control program must also be capable of letting
different processes run in parallel. The Control program must perform the following functions:

- Functions for technical processes
- Enable dynamic handling of graphics objects

The Control program processes data points and controls the visualization of process states. The Control syntax is based on the procedural programming language C. Control, however, is implemented as an interpreter, so that the user does not have to perform compiling or linking. This means that changes can be tested immediately. Control provides optimum support for the flexible data point concept of WinCC OA.

1.3.2 JCOP Framework

The LHC experiments' control systems are being developed by hundreds of different developers, distributed widely around the world. This leads to autonomous teams developing solutions which at some point have to be drawn together to form the final experiment control system. The complexity of integrating all these different developments could be very time consuming if measures are not taken to guide the work of the development teams.

JCOP was established as a collaboration between CERN and the LHC experiments to identify common needs and provide common tools and solutions that the four experiments can use as a basis for the development of their final control systems. JCOP has identified a number of standard technologies that shall be used, including several industry standards and products. The chosen technologies include communication protocols, PLCs, field buses and a SCADA system.

The main motivation for the development of the FW is to reduce the effort needed by developers of the experiment control systems. As discussed earlier, by providing guidelines to developers – including standard look and feel, naming conventions, etc. – the task of integrating the many different developments will be simplified. Also, the FW provides common functionality that is required in a high energy physics (HEP) environment – in particular the LHC experiments. By providing this in the FW, experiment control system developers can immediately use a tested piece of software and do not have to develop the whole solution themselves. Any development in the FW is available to all experiments, which means that common features can be developed once for the FW and reused many times within each of the experiments. This helps to avoid that experiments spend time to develop custom software that could instead have been developed once and used by all the experiments.
2 Work Conducted

2.1 Project Overview

In WinccoA we use databases to store information. For this purpose we need to make dp(datapoints) and dpe(data point elements) with appropriate structure.

The main goal of this project was getting the list of all dpes for a defined project (which was the CMS control system) and the challenge of this work was reconstructing this rather complicated structure to make an address to specify a dpe. Actually making an address for a general purpose was a challenge otherwise we could use the WinccoA’s GEDI(graphical editor)’s tools for querying a dpe but for example we couldn’t use that tool for query _address configurations_. What we needed was a script to build this address for dpes contained in a dpt(data point type) which has the most general construction. It may contain reference data point elements\(^1\) or other kinds of structures that a dpe may has(which is indicated in figure 2).

![PARA: a graphical interface for editing data point types and data points. This interface constitutes a tool. With the tool you can access the internal database and make modifications simultaneously.](image)

2.2 Description of Solution

In order to reconstruct the construction of dpts and make the mentioned directory first we need to have the list of all dpts which easily can be achieved by using a built in function called \texttt{dptypes()}\footnote{This type of dpe contains the structure of a dpt which has the same name as the reference dpe. A reference dpe is built to avoid redundancy of building a same structure again and again.} after that we start this reconstruction for every dpt structure included in this list.
the whole idea is defining the function that gets a dpt as an argument and returns all of
the dpes in it by addressing it in a way that can be used by another process for acquiring
data. We name this function `dpgetquery()` and the main idea is using the information
that we get from the built-in function `dpTypeGet()` which has a special format of
returning information that would be very helpful for us, actually it renews the values
of two predefined values called `names` and `types` where `names` contains the names of
all dpes and `types` contains integer numbers which indicate the types of dpes (based on
the figure 2). According to this numbers we can retrieve the type of an element of the
name variable (which is a `dyn_string`) now thanks to the special form of the output
of `dpTypeGet()` (figure3), based on the length of the elements and sub elements of each
element we can fully retrieve the tree of a dpt (as you can better see from the script)

### 2.2.1 Type References

The solution described above works for all dpes except type references because
`dpTypeGet()` does not treat them in the same way as other types and does not return
inner structures for this types. However, we know that there is at least a dpt with
exactly the same name as this dpe (the one that refers to this dpe). Thus, we call the function `dpgetquery()` inside itself for the corresponding dpt (which makes this function a recursive function) you can see it in lines 32-37 of final script.

Figure 3: Output of `dpTypeGet()` for TkPowerGroup dpt (which is indicated in figure 1) in log

Note that because all dps in the same dpt have the same structure we just need to reconstruct the structure for the dpt.

### 2.3 Final CTRL Script

```c
main()
{
  void getparentname(string &parents);
  {
    string n="parents[1]";
    for(int i=2;i <= dynlen(parents);i++){
      n=".parents[i]";
    }
    return n;
  }
  dyn_string flist;
  dyn_dyn_string names;
  dyn_dyn_int types;
  void getdpequery(string dptname,dyn_string &flist);
  {
    dyn_string parents;
    dpTypeGet(dptname,names,types);
    for(int i=2; i <= dynlen(names); i++){
      if (types[i][dynlen(type[i])]! =1&&types[i][dynlen(type[i])]! =41}{
        if(dynlen(name[i])> dynlen(parents)+i){
```
Note that in lines 3-8 we have defined function `getparentname()` to make the appropriate address after getting to a dpe.

After defining `dpgetquery()` we need to execute it for all dpts to get the list of all dpes (which is done in lines 39-44).
3 Conclusion

A huge detector like CMS is composed of thousands of different pieces where each part should work in a proper way to produce reliable data. Therefore, we need to take care of each part by monitoring the status of sensors as well as the other parts and produce proper alarms in the case of errors.

Due to the massive amounts of components storing the information such as status of this different parts in databases as dps and dpes leads to quite complicated structures. Because of this complexity extracting data from configurations of dpes contained in a dpt with the most general construction in not as easy as it seems and there is no appropriate tool in WinCCOA for this purpose specially for \textit{address} and \textit{archive} configurations. We wrote a CTRL script which can be used to make the list of all dpes for examining all kinds of configurations.

For me it was my first experience of dealing with control systems and it was completely out of my comfort zone but overall, it was a great experience full of learning new staff specially about DCS and I realized that how much storing data about status of sensors and other parts of the detector is complicated and important in a physics experiment and how much making collaboration with experts from different fields such as engineering and working in a group is necessary in a real physics experiment to make the process more precise and reliable.

My work was part of the project of renewing the CMS’s DCS interface and I hope that it would be helpful for future users and helps safety team in this process of renewing.

4 References


