Dcs Data Viewer, an Application that Accesses ATLAS DCS Historical Data

C Tsarouchas, S Schlenker, G Dimitrov and G. Jahn
CERN, Geneva, Switzerland
E-mail: tsarouch@cern.ch

Abstract. The ATLAS experiment at CERN is one of the four Large Hadron Collider experiments. The Detector Control System (DCS) of ATLAS is responsible for the supervision of the detector equipment, the reading of operational parameters, the propagation of the alarms and the archiving of important operational data in a relational database (DB). DCS Data Viewer (DDV) is an application that provides access to the ATLAS DCS historical data through a web interface. Its design is structured using a client-server architecture. The pythonic server connects to the DB and fetches the data by using optimized SQL requests. It communicates with the outside world, by accepting HTTP requests and it can be used stand alone. The client is an AJAX (Asynchronous JavaScript and XML) interactive web application developed under the Google Web Toolkit (GWT) framework. Its web interface is user friendly, platform and browser independent. The selection of metadata is done via a column-tree view or with a powerful search engine. The final visualization of the data is done using java applets or java script applications as plugins. The default output is a value-over-time chart, but other types of outputs like tables, ascii or ROOT files are supported too. Excessive access or malicious use of the database is prevented by a dedicated protection mechanism, allowing the exposure of the tool to hundreds of inexperienced users. The current configuration of the client and of the outputs can be saved in an XML file. Protection against web security attacks is foreseen and authentication constrains have been taken into account, allowing the exposure of the tool to hundreds of users world wide. Due to its flexible interface and its generic and modular approach, DDV could be easily used for other experiment control systems.

1. Introduction
The ATLAS[1] Detector Control System (DCS) [2] is responsible for the coherent and safe operation of the detector. The DCS supervises the detector’s hardware and monitors all operational parameters, it provides online status information and signals abnormal behavior. It executes operator commands, it stores important operational information into a database and propagate alarms in case of problems too. The ATLAS DCS Data Viewer (DDV) tool provides offline access to DCS data concerning values of archived parameters and to DCS archived alerts.

2. Aim and specifications
The DDV project aims for implementing a data viewing application for DCS data. It primarily targets users from the whole ATLAS collaboration and is accessible through a user friendly web interface. The development of the application is based on a list of initial specifications: platform and browser independent project, reasonable application start up time (less than 10seconds), small response time to requests (order of second), user friendly navigation through metadata,
multiple output formats (chart, table, ascii, ROOT), current configuration in XML format, database protection mechanisms.

2.1. Project architecture

The DDV service is based on a highly modular architecture of a server-client model which offers several advantages like reduced network load (only the results of the processes are transported), flexibility (new outputs and plugins are easily integrated), efficiency (work load is shared among the server and client software building blocks). Finally, a variety of different languages and technologies are used which keeps the implementation time as minimum as possible and the future maintenance facilitated.

The schematic in figure 1 illustrates the DDV architecture and shows the different software building blocks that it is composed of. The DDV server accepts requests concerning DCS information, constructs and performs database requests to the DB and finally returns the results in a form of a structured output. The DDV Server is split in two main modules, one that deals with the metadata (identifiers that describe some hardware within the DB, e.g. a temperature sensor) and one that deals with the data. The client is a browser independent web application which is composed of a selection module (navigation through metadata), a core module (constructs the server requests and handles the responses), and an output module (different ways of data visualization). A mechanism that saves the current configuration in XML files for a later use is also in place.

Figure 1. The DDV server-client architecture.

3. Server technical aspects

The DDV server aims to perform various low level and diverse functionalities which can be optimally developed in a structured way, by using a few sets of classes and methods. Moreover, often the server software block needs to interact with the machine it is running on and to have well defined interfaces to the DB and to the HTTP requests too. Taking the above into consideration, the DDV server is decided to be written in python.

3.1. Metadata organization

The metadata information is copied from the offline database to an SQLite database cache [3] within the DDV server which serves two different purposes. Firstly the SQLite tables are configured to be stored in memory offering in this way the quickest possible response time of metadata queries. Secondly, all queries concerning metadata are performed exclusively inside the DDV server keeping the database resources as available as possible.

3.2. Data retrieval

The pythonic server acquires data from the DB using the cx_Oracle extension module [4]. The retrieved data contain the DB metadata information concerning the requested item and its values with their time stamps within the requested period.
Intending to protect DB good operation, DDV validates and certifies each request with a minimum-response-time cost and ultimately propagates it to the DB. This database protection mechanism is organized in three levels. Firstly DDV applies some ‘hard’ cuts. Requests with time periods of more than 2 years or including more than 200 items are considered to be potentially dangerous for the DB and are declined. Secondly, DDV performs a light test-request with a time period significantly shorter than the one requested. The number of returned rows is translated to a data rate (Hz) and by extrapolating to the full query time, a decision is made whether this request is acceptable or not. Thirdly in order to protect against unforeseen reasons, DDV cancels the request if no DB response arrives within 20 seconds. If any of the above conditions is reached, the request is revoked and a corresponding explanatory message is constructed and sent to the DDV client so to inform the user.

3.3. Server stand alone use

DDV can be used in batch mode which offers the option of accessing DB information directly by calling the DDV sever (e.g. through a terminal) without the need of a browser or any other graphical environment. The server accepts HTTP requests which are decomposed to a meaningful DB request information with the use of the pythonic web framework CherryPy [5].

4. Client technical aspects

The implementation of the DDV client is based on the Google Web Tool kit (GWT) framework for web applications [6]. The development is done in Java while the result after the GWT compilation is a browser independent AJAX-based web application. The client interface is organized in two main modules. The top module (figure 2), developed with the GWT framework, host the selection and configuration options. The bottom module is based on Java Applet and JavaScript technology and hosts the output plugins.

4.1. Selection

Carefully chosen widgets allow a quick selection in an intuitive and user friendly way. The time and date selection is done via graphical interfaces and the navigation through metadata is performed via column-tree widgets. However, the big number of archived parameters can create difficulties to non-expert users. For such cases DDV provides a Search Engine for DCS metadata. In the server side, the search string is translated into a case insensitive database request pattern which is sent as part of the whole database request. Furthermore, in some cases, users are interested in a subset of data that satisfy some criteria (case of spike). DDV takes into consideration such needs and provides users with a Relational Query mechanism where users can specify an accepted range of values for the selected item(s). In this way the database resources are used as little as possible since only a small fraction of rows that pass the requested criteria is returned back.

4.2. Configuration

DDV offers the possibility of saving the current configuration to a file which can be used in future time and quickly visualize the wanted information. The configuration file is chosen to be in XML standard which is simple, easily readable and flexible enough to accommodate future diverse needs. The contents of the configuration file can be grouped in three parts, general information of the DDV operation mode (numerical data or alert information queries), specific information of the selected items (e.g. starting/ending time, relational query information) and output configurations (e.g. color of lines, Y-axis range, legend names). Furthermore, DDV server hosts a repository of configuration files where each user, via the client interface, has access and rights to store configuration files. In this way the user can call DDV by pointing in the same time
a configuration file that is pre-saved in the server. Since all the requested information is kept in the configuration file, a single call accompanied with the file location as a URL parameter, is enough to pop up a new browser window with the DDV page and the output already populated.

5. DDV outputs
Aiming to satisfy the diverse needs of users, DDV provides currently five different outputs. Moreover a thin interface between the main client module and the outputs is well defined, permitting easy integration of new outputs.

The default output of DDV is a Java Applet chart (Figure 3), implemented using Java Applet technology based on the JfreeChart [7] libraries. Such output offers cross platform compatibility and user verification (approval before download). The default display mode is value over time trend but the user has the option of projecting the data to one or two dimensional histograms (when applicable).

A second chart output of DDV is a JavaScript chart based on JavaScript technology and it is implemented using the versatile jQuery library. This output enjoys the advantages of JavaScript like speed (code functions run immediately in the browser), simplicity (simple to learn and implement), versatility (easily integrated in other environments), universality (supported by most web browsers including smart phone ones).

Apart from the chart outputs a JavaScript table offers powerful sorting and filtering.

Figure 2. The main browser window interface of DDV.
Due to its powerful features, the JavaScript table is used as the default output of the DDV alarm mode, an example of which is shown in figure 4.

![Figure 4](image-url)

**Figure 4.** The output of the DDV alarm mode is a JavaScript table with powerful filtering and sorting capabilities.

**References**

[1] The ATLAS Collaboration 2008 JINST 3, S08003
[2] S. Schlenker et al. ICALEPCS 2011 MOBAUST02
[3] SQLite documentation, http://www.sqlite.org/docs.html
[4] cxOracle Documentation, http://cx-oracle.sourceforge.net/html/index.html
[5] CherryPy Documentation, http://docs.cherrypy.org/stable/
[6] GWT API Reference, http://code.google.com/webtoolkit/doc/latest/RefGWTClassAPI.html
[7] JfreeChart documentation, http://ktipsntricks.com/data/ebooks/java/jfreechart-0.9.1-US-v1.pdf
[8] ROOT Documentation, http://root.cern.ch/root/doc/RootDoc.html