ATLAS DETECTOR CONTROL SYSTEM DATA VIEWER

Charilaos Tsarouchas, S. Schlenker, S. Roe, CERN, Geneva, Switzerland
U. Bitenc, M.L. Fehling-Kaschek, S. Winkelmann,
Albert-Ludwig Universität Freiburg, Freiburg, Germany
S. D'Auria, University of Glasgow, United Kingdom
D. Hoffmann, O. Pisano, CPPM, Marseille, France

Abstract

The ATLAS experiment at CERN is one of the four Large Hadron Collider experiments. DCS Data Viewer (DDV) is a web interface application that provides access to historical data of ATLAS Detector Control System [1] (DCS) parameters written to the database (DB). It has a modular and flexible design and is structured using a client-server architecture. The server can be operated stand alone with a command-line interface to the data while the client offers a user friendly, browser independent interface. The selection of the metadata of DCS parameters is done via a column-tree view or with a powerful search engine. The final visualisation of the data is done using various plugins such as "value over time" charts, data tables, raw ASCII or structured export to ROOT. Excessive access or malicious use of the database is prevented by dedicated protection mechanisms, allowing the exposure of the tool to hundreds of inexperienced users. The metadata selection and data output features can be used separately by XML configuration files. Security constraints have been taken into account in the implementation allowing the access of DDV by collaborators worldwide.

SPECIFICATIONS AND DESCRIPTION

The DDV project aims for implementing a data viewing application for DCS data. The application primarily targets users from the whole ATLAS collaboration allowing to access and display data from database archives. The list of the specifications was the starting point of the development and it went through several revisions driven by development expertise and end-user requests:

- Platform and browser independent project,
- Reasonable application startup time (less than 10sec),
- Small response time to requests (order of second),
- All possible navigation mode options (element_name, alias, description),
- Multiple output formats (chart, table, ascii, ROOT),
- Current configuration in XML format option,
- Database protection mechanisms.

The DDV implementation is based on client-server architecture (see Figure 1). The server part is composed of two main parts, one that deals with data and another one that deals with metadata. The client-server communication is done via the http protocol. The client is composed of two main parts, the request-creation part and the output. The highly modular application offers the possibility of implementation of many interesting features which will be described in the next sections.

DDV SERVER

The DDV server is written in python. It accepts requests, it communicates with the DB, it manipulates data in case it is needed and finally returns back the results.

Metadata Organization The ATLAS DCS metadata is information that refers to an archived parameter (*element_name*) that identifies that specific item in the Oracle tables. Besides, each *element_name* can have a *description* and an *alias* which are more user friendly definitions. The metadata information is copied from the database to an SQLite [2] database cache within the DDV server. The organization of metadata in SQLite file, 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 DDV server keeping the database resources as available as possible.

Data Retrieval In the case of data requests DDV acquires data directly from DB. The server communicates with the DB using the cxOracle [3] extension module that allows access to Oracle databases. The request engine is designed in a way that minimize the DB response time and considers possible changes in the mapping between element_name and alias or comment.

Server Stand Alone Use DDV tool can be used in *batch mode* which offers the option of accessing DB information directly by calling DDV sever (e.g. through a terminal) without the need of a browser or any other graphical environment. The server accepts HTTP requests by using a pythonic framework called cherrypy [4]. As soon as a new URL reach the server, it is decomposed and its parts are used to create a meaningful DB request information.
Requests, GET method:
[server]:[port]/metadata/[queryType]/
[selectedSchema]/[system]/[pattern]
e.g. `http:/pcaticstest07:8089/metadata/element_name/atlas_pvssmdt/ATLMDTPS2/*temp*`
Data requests, POST method:
[serverAddress]:[port]/multidata/getDataSafely
e.g. `http://pcaticstest07:8089/multidata/getDataSafely,queryInfo=atlas_pvssdcs,comment_,CICRackControlLArgY0721A2Humidity,10-10-201012:0,11-10-20100:0`

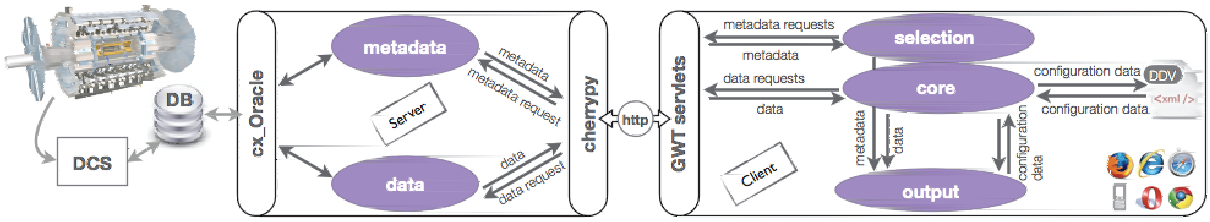

Figure 1: The DDV server-client architecture.

Relational Queries For the cases that users are interested in a subset of data that satisfy some criteria (typical example is the case of a spike) DDV provides a *Relational Queries* mechanism where an accepted range of values can be specified for the selected items. Each item can be configured separately and have its own accepted value region.

Data Base Protection Mechanism Intending to serve a high number of users , DDV aims to be more than an interface to database data. DDV validates and certifies each request with a minimum-response-time cost and finally propagates it to the DB. This protection mechanism is organised in three levels and can schematically be seen in Figure 4. Firstly DDV applies *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. A second protection mechanism 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. Finally, a third kind of protection mechanism is in place to cover the cases that some unforeseen reason can cause the request to hang. In case DDV does not get any answer from the DB during 20 sec, the request is cancelled before completion.

DDV CLIENT

The client part of DDV is largely based on the Google Web Tool kit (GWT) framework for web applications [5] . The development is done in Java while the result after GWT compilation is a browser independent AJAX-based web application. Keeping the development in a type-based language like Java the testability increases and the debugging of code becomes faster with the help of a Java debugger (e.g. eclipse). The client interface is organized in two main parts. The selection and configuration module (Figure 2) is developed with the use of the GWT framework . With the use of menu items, buttons, tables and other widgets the user enters the request selection criteria and a server request is constructed. The bottom part is independent of the GWT framework and hosts the output plugins.

Selection

Column based browsing DDV provides an easy navigation mode through metadata using column-tree widgets. Since the metadata information is kept in the server in cache, the server response is prompt and in each request the result is returned back to the client in a fraction of a second.

Search Engine The *column lists* is an easy navigation mode. However, the vast number of archived parameters can create difficulties to non-expert users. For such cases DDV provides a *Search Engine* for DCS metadata. The user has to provide one or more strings that will be used as a pattern, separated with the wildcard character '*' . In the DDV server side the search string is translated into a case insensitive SQL pattern which is sent as part of the database request. Apart from the user friendly search engine an *Advanced Search Engine* is in place, that supports regular expressions.

Configuration

Despite the intuitive selection interface of the tool and the optimization of the response time in every action, the selection of the wanted items and the configuration of the final output may take some time. DDV offers the possibility of saving the current configuration to a file. In this way, users can save the current configuration and in a future time upload it again and quickly visualize the wanted information. The configuration file includes general information of the request (e.g. starting/ending time and date), the selected metadata information, the selected output, configuration of the selected output and relational queries information. The configuration file is chosen to be in XML standard which is simple, easy readable and on the same time flexible enough to accommodate future diverse needs.

DDV Outputs

DDV aims to offer to users several output options which satisfy different needs. These outputs are connected with the DDV client with a thin, well defined, interface. There are two main categories of outputs, visual outputs that appear on the browser and output-files to be downloaded to the users disk.

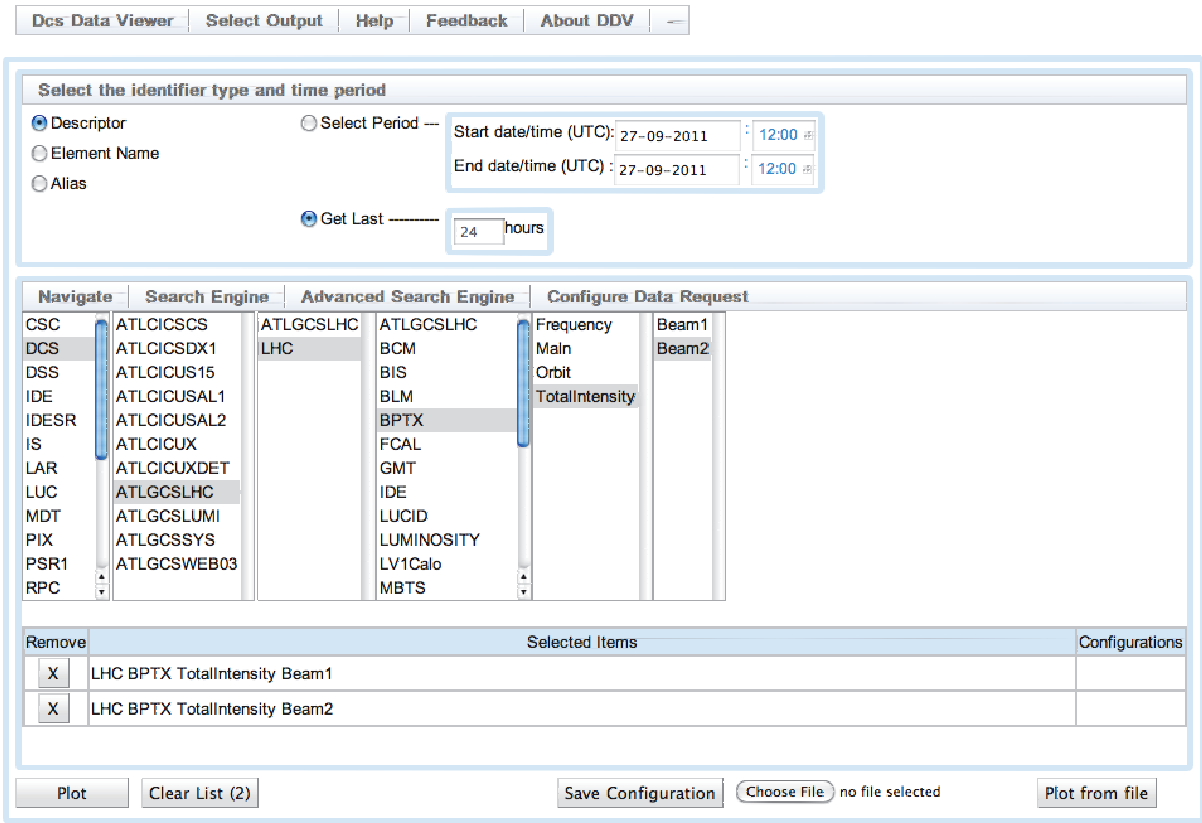

Figure 2: The main browser window interface of DDV. The top part deals with the selection of the navigation mode and the start/end date and time. The middle part offers an easy navigation among the metadata with a column-tree view. The bottom part holds the action buttons like *plot* for a graphical display of data and *save configuration* for saving the current configuration in XML format.

Chart Applet The default output of DDV is a chart (Figure 3), implemented in form of a Java Applet based on the JfreeChart [6] libraries. Depending on the format of the selected data the user can choose to display these as time series, one or two dimensional histograms. The output configuration can be defined either in the main DDV browser interface or interactively in the Applet in a flexible way.

Chart Java Script The Java Script Chart is another output for graphical representation of the data. Concerning the plot options it is more restrictive comparing with the default output but it is more light for the browser. Furthermore, since it is written in Java Script , it can be visualized in environments that do not support java applets (e.g. case of some smart phones).

Table Java Script Data table is a Java Script output that displays the database response in a table. The information held in the table is the archived parameter, the time-stamp and the value. Moreover, the output offers the options of searching and filtering.

ROOT DDV offers the possibility to download the selected data in ROOT [7] format. These files consist of one TTree for every selected data series containing times and values of the contained data points. In this way the user can process the data flexibly in ROOT macros. Besides these trees, there is also a set of predefined TGraphs for the selected data included in the output file.

ASCII Output ASCII file is a simple and highly universal output format. It keeps the information of the archived parameter, the time-stamp and the value.

External Requests

The deliberately modular design of the tool in the initial phase of the development, paid well back in the accomplishment of powerful features. With the use of *External Requests* the user can call DDV by pointing in the same time a configuration file that is pre-saved in the server. All the requested information is kept in the configuration file and in this way a single call accompanied with the configuration file location as a url parameter, is enough to pop up a new browser window with the DDV page and the chart already populated.

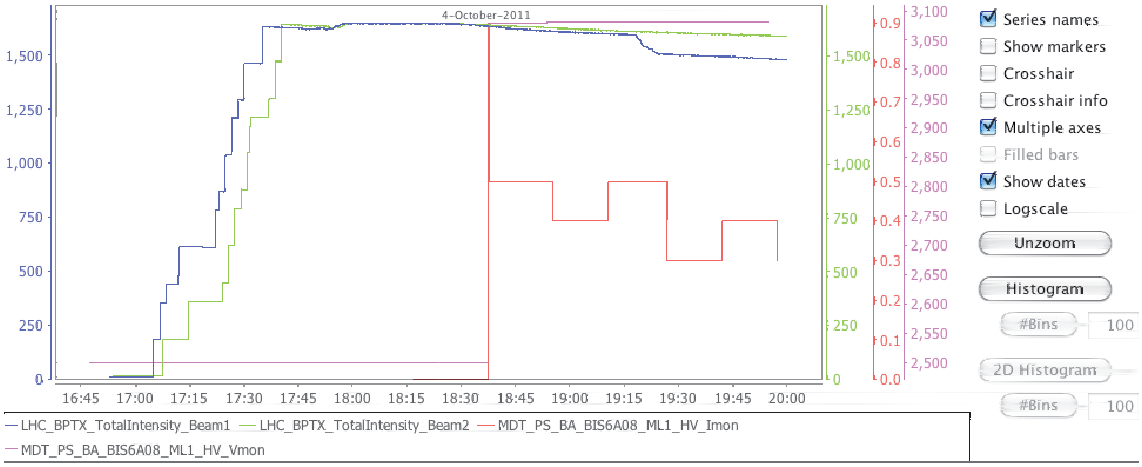

Figure 3: JfreeChart, the default output of DDV. Except of displaying graphically the data, it offers several features like multiple axes, logarithmic scale, markers display, crosshair information, projection of data to 1D and 2D histograms (where applicable).

RESULTS

DDV is officially released as ATLAS central service since the January of 2011. Security constrains have been taken into account in the implementation allowing the access of the application by collaborators worldwide. Currently the number of DDV users is about 150 and it grows constantly.

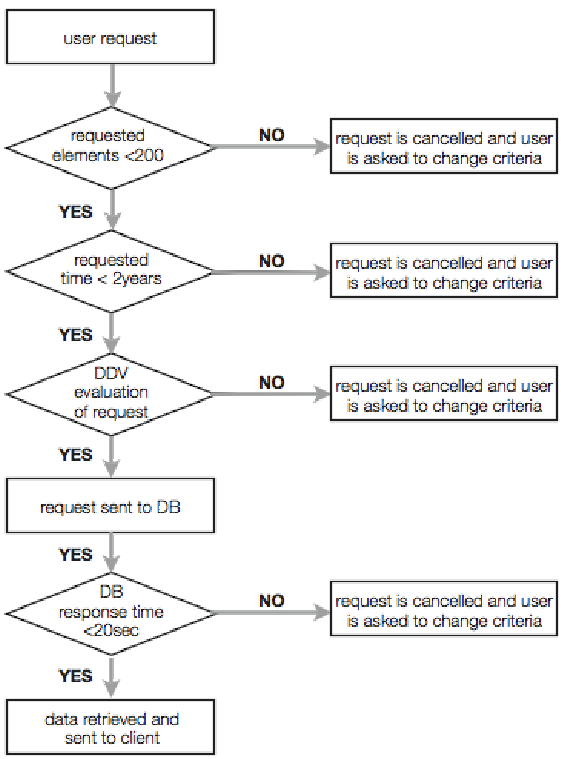

Figure 4: DB protection mechanism.

REFERENCES

[1] A. Barriuso Poy et al., "The detector control system of the atlas experiment", JINST 3 (2008) P05006.

[2] SQLite documentation, http://www.sqlite.org/docs.html.

[3] cxOracle Documentation, http://cx-oracle.sourceforge.net/html/index.html.

[4] CherryPy Documentation, http://docs.cherrypy.org/stable/.

[5] GWT API Reference, http://code.google.com/webtoolkit/doc/latest/RefGWTClassAPI.html.

[6] David Gilbert, JfreeChart documentation, http://ktipsntricks.com/data/ebooks/java/jfreechart-0.9.1-US-v1.pdf.

[7] ROOT Documentation, http://root.cern.ch/root/doc/RootDoc.html.