PDV – a PVSS Data Viewer Application

Dirk Hoffmann, Olivier Pisano – Centre de Physique des Particules de Marseille, case 902, 163 avenue de Luminy, F-13288 Marseille CEDEX 09

The PVSS Data Viewer (PDV) has been developed to access environment and control data of the Pixel detector of the ATLAS experiment, with an effort to be sufficiently generic to provide access to data of other subdetectors and even data of other experiments or PVSS systems in general. Other important keys for the design were independence from any existing PVSS installation and universality regarding operation systems or user environments.

Design Criteria

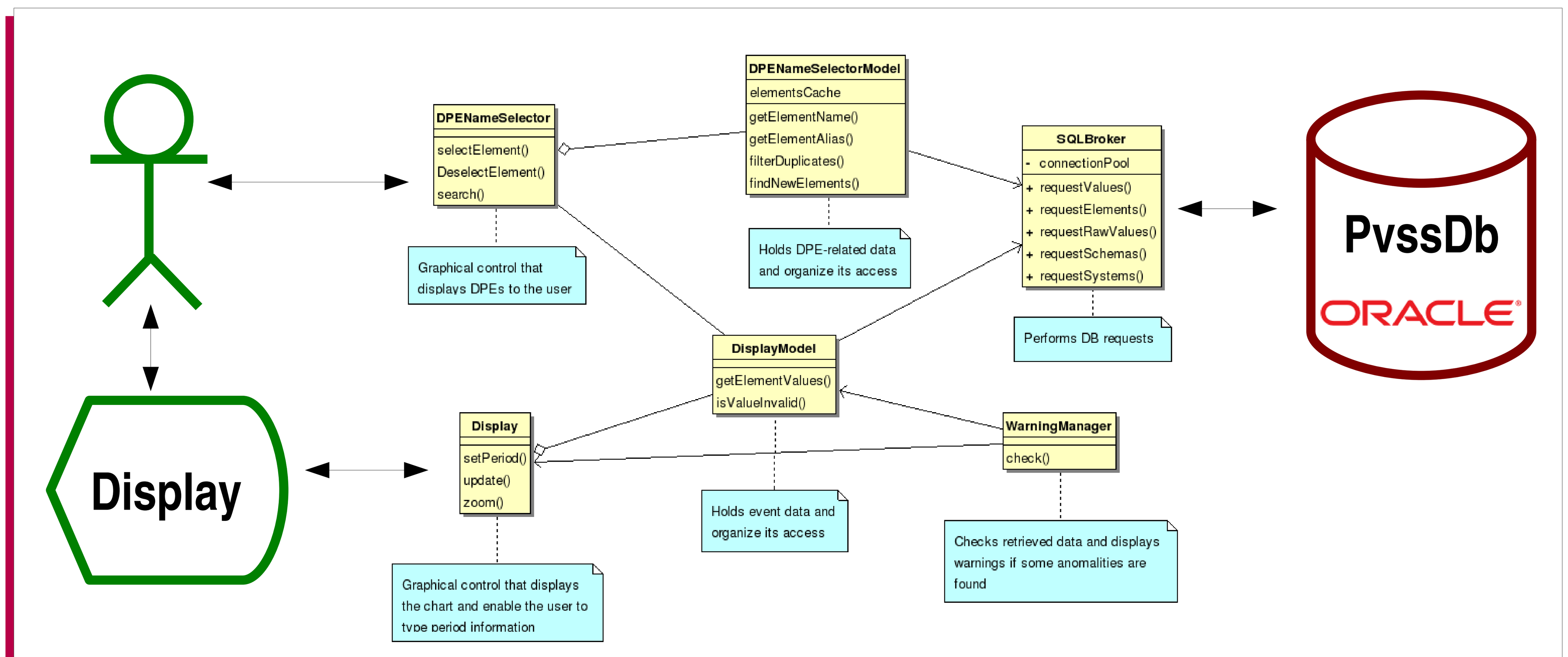
- **GenericPVSS/PvssDb interface**
  no detector or experiment specific features
- **Universal (Java VM)**
  No OS, environment or policy dependency
  WWW-able (Java Network Launch Protocol, JNLP)
  No dependency on PVSS installation
- **Decompression and clever display**
- **Basic analysis functions**
- **„All possible“ save-as options**

Technical Implementation

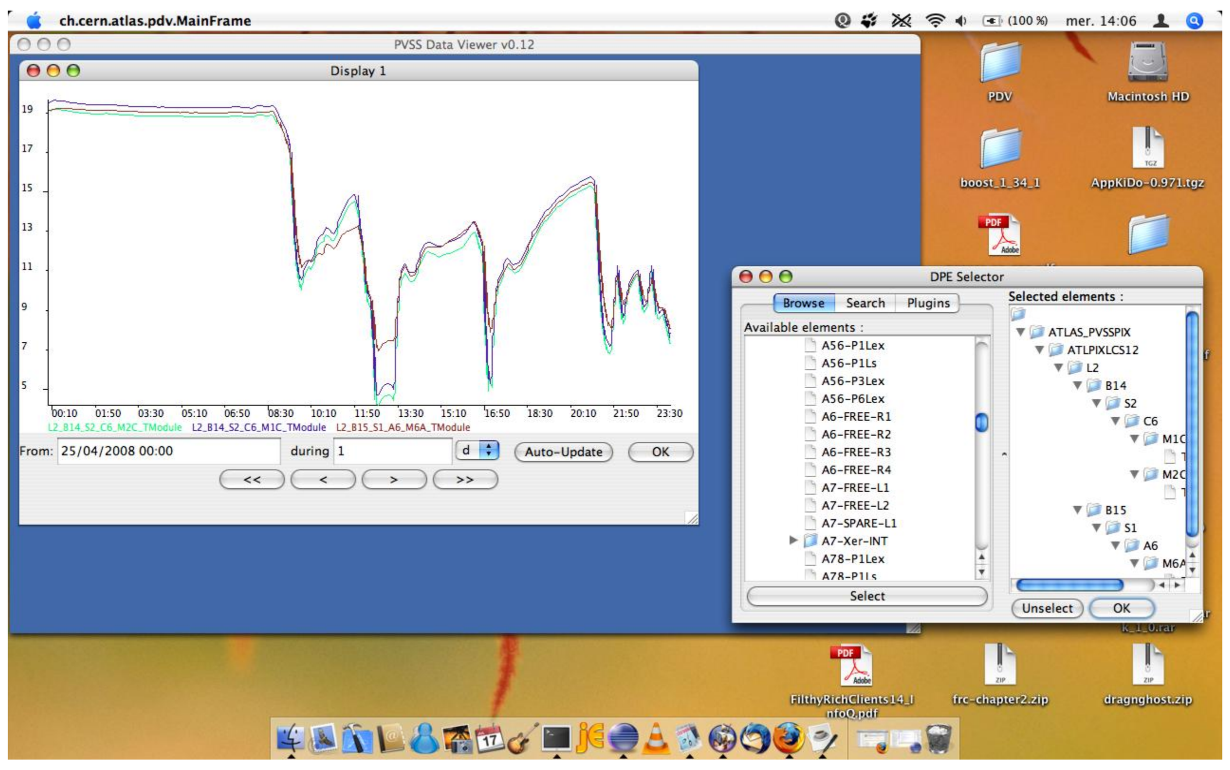
A data flow diagram embedding the URL for the software design is shown below. The user interacts with the application mainly through the DPE name selector dialog and the dialog part of the currently selected display chart, which selects the time interval (period) of data that are being queried.

The interna of the data storage and caching of DPEs for schemas which have already been used in a previous session, are completely hidden from the user. A typical screenshot from a user session is shown here. The user selects the data period for which data are going to be queried (window **Display 1**). DPE definitions can vary with time and are taken into account in a way that is transparent for the user.

The **DPE Selector** window shows the schema and DPE structure in tree form. The relatively long DPE names are cut intelligently in order to arrange them in the tree. The DPE name display can be changed at any time (for display and selector) from DPE names to the PVSS alias convention.



The PDV application links user and database: The user interacts through the Display and DPE Selector components of the application with the PvssDb.

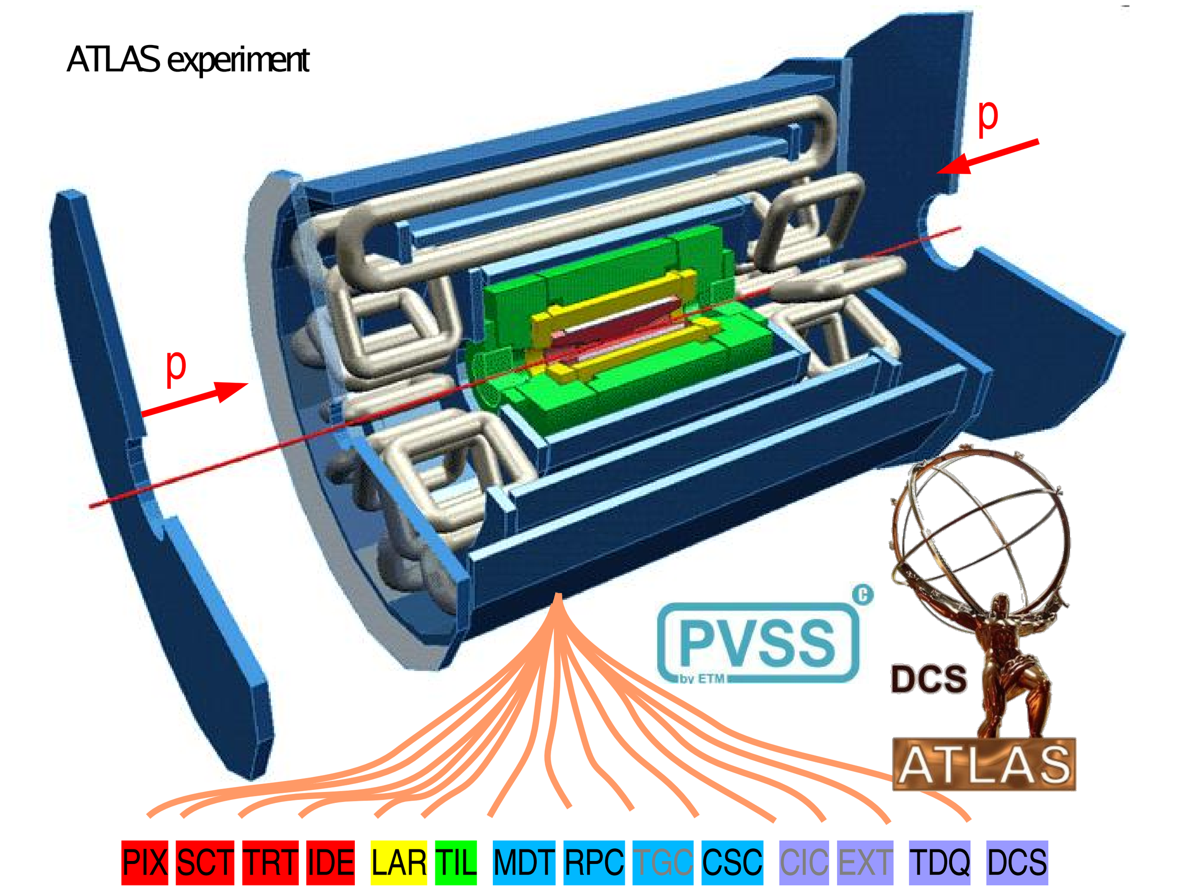


Screenshot of a typical user session on Mac OS X

Context and Origin of Data

The ATLAS experiment uses PVSS, an industrial SCADA application, for its detector control system (DCS). A custom driver establishes a connection to a central Oracle instance for the purpose of archiving the recorded data in the PVSS Database (PvssDb).

The ATLAS DCS consists of 14 different subsystems, associated roughly to subdetectors, which contain a total of some hundred computing nodes in form of Windows and Linux PCs. The PvssDb of the Oracle instance is presently filled by 11 out these subsystems with an allowed average rate of 3 GB per day, which translate into $5 \cdot 10^6$ values. This average daily data volume of 30 GB needs to be handled efficiently by any application that implements user queries.
As opposed to physics data which undergo a systematic reconstruction process, DCS data are typically queried on demand, whenever an untypical event has happened or is suspected to have happened.



Atlas experiment and names (abbreviations) of the PVSS subdetector schemas

PVSS Data Compression

PVSS applies a data compression algorithm called *smoothing* prior to writing online data into PvssDb:

- New data that have not changed by an amount greater than a predefined *deadband* are ignored.
- New data which are received from the front-end after expiry of a predefined *timeout* interval are written, whether the deadband is exceeded or not. Thus the credibility of stable data measurements is maintained at a sufficient level of confidence.

This algorithm is applied for all data values (data point elements, DPEs) independently. As the front-end data are in general not synchronous, this leads to a continous data flow into the PvssDb.

Data Reconstruction (and recompression)

As the PDV software aims at a completely transparent generation of displayed and extracted data for the user, a reconstruction of the compressed raw data in the PvssDb in performed at query time on SQL level, and inside PDV in order to fill all time slices that would be left empty due to compression of relatively stable DPE values.

Some effort has been made to optimize long-term queries (days or weeks, even months, if DB load restrictions allow). The actual resolution of the user display is used in order to determine the maximum useful resolution: The user will not be able to distinguish different values that are displayed in the same pixel interval of the abscissa. Therefore the query is optimized at SQL level to calculate a minimum and maximum per pixel time interval equivalent. Thus data transfer and reconstruction effort are optimized inside the Oracle instance.

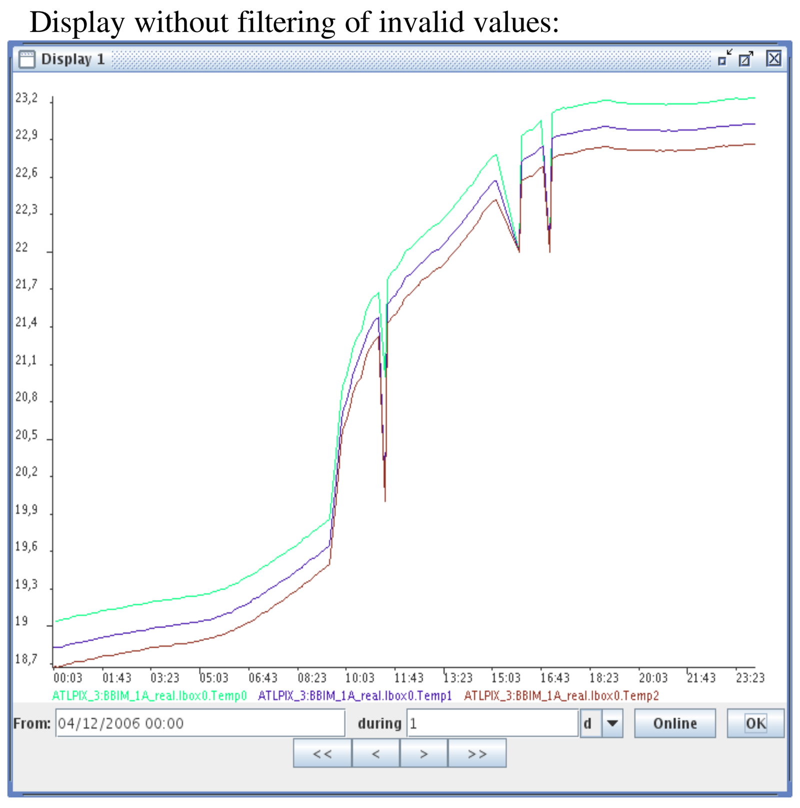
Universality and Compatibility

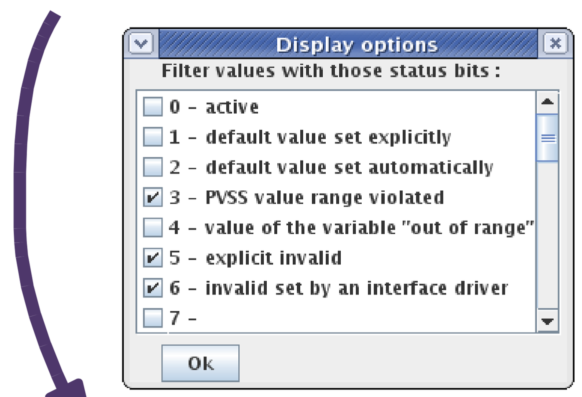
In order to allow access to PVSS data to users outside of the ATLAS DCS cluster and even without PVSS installation at all, the choice for the machine independent, byte code generating language JAVA has been made. Development is made in JDK 1.5, and applications have also been tested to run in JRE 1.6 on Linux (SL/SLC), Windows (2k, 2003, XP, Vista) and Mac OS X.

Application Features 1: Invalid data

- **Invalid data tagged by PVSS through a 32 bit mask**
- **Filtering of invalid data by PDV, fully customizable bit mask**

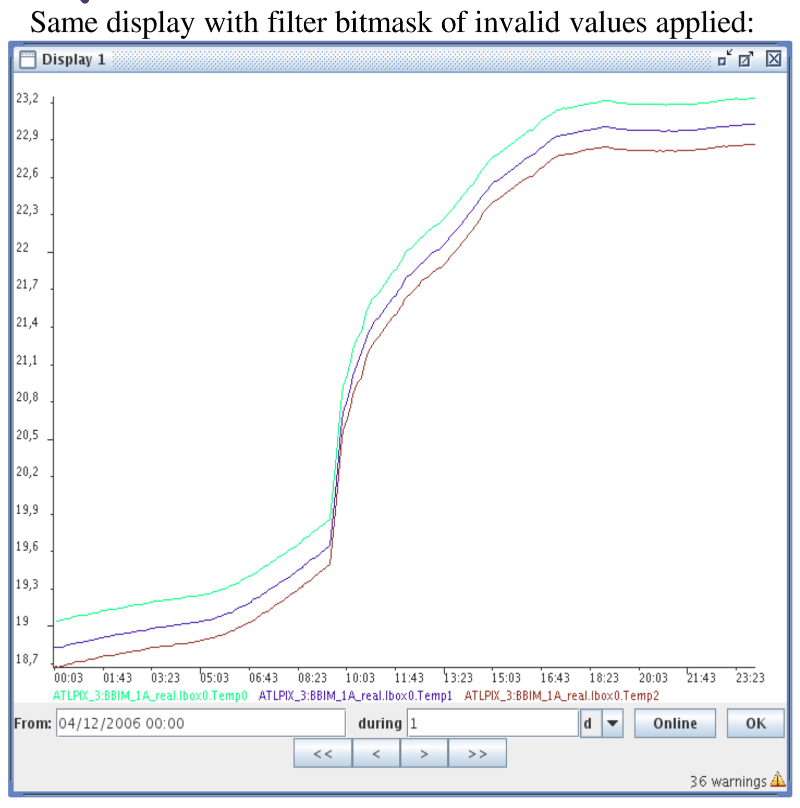
Display without filtering of invalid values:





Invalid bitmask selection dialog

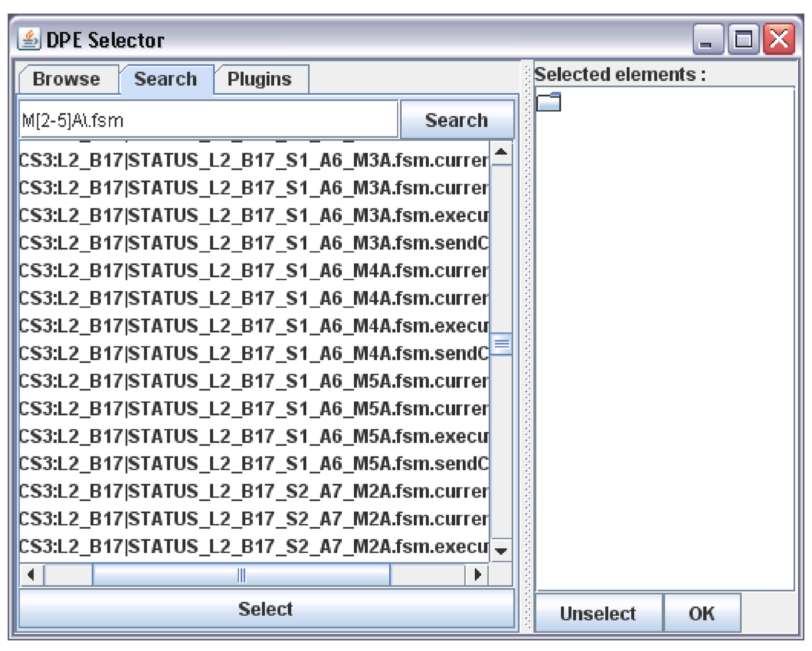
Same display with filter bitmask of invalid values applied:



Application Features 2: DPE search

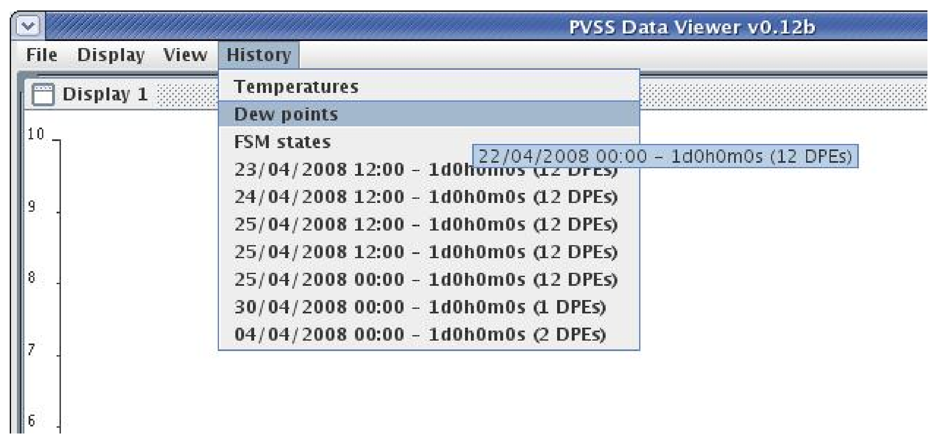
- **Selection of DPEs in DPE Tree**
- **Search of DPEs with *regular expression* patterns on**
  - **DPE names**
  - **DPE aliases**
  - **DPE comments**

Snapshot of **Search** tab in the DPE Selector window



Application Features 3: History

Users typically verify same groups of DPEs routinely or, more generally, want to re-make a display query that they have executed formerly. The PDV retains the last effected queries for each user in the user specific PDV database (situated in the „home“ directory).



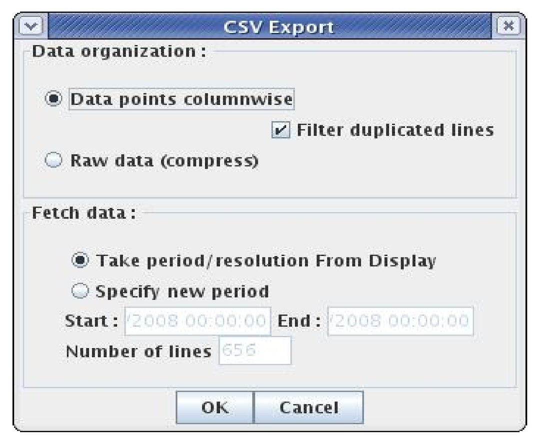
A screenshot of the corresponding **History** menu is shown above, which also illustrates an additional feature of this history mechanism: All queries are stored in form of **<History><Record>**s in an XML file. Thus users can, after acquiring some basic knowledge of the underlying XML structure, edit history files with a standard XML editor or just any text editor, or generate a history XML automatically.
PDV enables the user to select any of the history records to be displayed (thus re-queried) at startup, which allows to use PDV for some pre-programmed data plots semi-automatically without initial user intervention. The menu above makes use of individual identifiers (Temperatures, Dew points, …) for easy retreival of aditionally generated history records; otherwise the data period is used for identification.

Application Features 4: Export and Save Data

The possibility of exporting displayed data after a query has been explored exhaustively and is probably the most complete feature of the PDV in its present state. Besides the standard graphical export,

- as printout (or PostScript as print-to-file) or
- as PNG bitmap graphics file,

the user can export the underlying data of a display as comma separated value (CSV)



Depending on his level of understanding or needs, he has the choice between multiple options, like

- raw data format (uncompressed from PvssDb, no reconstruction)
- reconstructed format (min/max per time slice)
- optimized reconstructed format where duplicate lines (stable values) are eliminated.

An interface to root is foreseen, but difficult to implement due to the requirement of machine independency, which excludes JNI interfaces by principle.

Application Features 5: Plugins

- **Accomodate subdetector/user specific requests**
  - **Selection of DPEs by specific algorithm or interface (possibly graphical)**
  - **DPE name/alias splitting for tree construction**
  - **Data Export in user specific formats**
- **User provided code, stored in `jar` or `class` file in user directory**

Software management

The source code of the PDV application has been successfully managed with CVS as repository and Savannah for user feedback, bug tracking and task management.

Outlook

After the first year of development of the PDV application, many additional requests in addition to the basic functions have been made by the user community.
Some subdetector specific requirements have come up with the extension of its use beyond the border of only the Pixel detector. And we expect further small functions to be needed, if users outside the ATLAS detector start to use the application.

The essential part of all requests is programmed for implementation until the LHC start. The product is supposed to go into maintenance phase then.
The plugin mechanism will allow users to contribute their own specific interfaces and to exchange them among others.

The relatively low level access of the PvssDb tables in Oracle is to be replaced by a common API provided by CERN IT-CO. Our experience gained with the interface to the Oracle instance and the PvssDb tables is presently leaving its mark on this central development as well.

Dirk.Hoffmann@CERN.Ch, April 2008