Data Access Interface and Implementation at Wendelstein 7-X

T. Bluhm[1], P. Heimann[2], Ch. Hennig[1], G. Kühner[1],

H. Kroiss[2], J. Maier[2,], H. Riemann[1], M. Zilker[2]

[1] Max-Planck-Institut für Plasmaphysik, Euratom-IPP Association, Wendelsteinstr. 1, 17491 Greifswald, Germany; torsten.bluhm@ipp.mpg.de

[2] Max-Planck-Institut für Plasmaphysik, Euratom-IPP Association, Boltzmannstr. 2, 85748 Garching, Germany

The increasing number of data acquisition stations and setups in laboratory environments at Wendelstein 7-X already produces a considerable amount of data by now. Diagnosticians want to view the acquired data in a comfortable way and use it for calculations in their own test and analysis algorithms. To prepare for full operation of W7-X a reasonable approach is to provide data access methods that stay as close as possible to the final data access concept. This requires to pay attention for the special needs of W7-X regarding continuously acquired data, segment based parameter switching and time based synchronization of data from different diagnostics. Therefore a data access interface has been designed and implemented considering continuous and steady-state data acquisition as well as usability and performance issues. The interface and its usage in different software environments (e.g. high level scientific programming languages) will be explained. Also the integration of user defined off line analysis algorithms will be described. Additionally the **DataBrowser** will be presented. The DataBrowser is a java application that can browse continuously acquired data of different kind, provides several types of plots including zooming functions to display the data and the corresponding parameters and offers export functions to save the data locally.

Keywords: Data Access, Data Viewing, User Interface, Continuous Acquisition

1. Introduction

Reliable data access methods for acquired data of laboratory installations at Wendelstein 7-X become more and more important. The implementation of the WEGA stellarator as a test bed for the control and data acquisition system of W7-X and diagnostic prototypes makes this need even more demanding. [1]

There are different challenges that have to be met by such a data access interface:

As a consequence of continuous operation time intervals of arbitrary length have to be handled. The size of a time interval can reach from less than a second (which will be the length of a discharge at the startup of W7-X) up to half an hour. It may also be necessary to observe data from constantly running diagnostics so that the time interval to be observed can span several days.

Another challenge is the huge amount of data that will be produced by some diagnostics. As it should be possible to access the stored data from every standard PC the fact that the data can not be loaded into memory at once has to be considered. Strategies to load only the data that is actually used and caching techniques have to be developed.

It is also an essential requirement that the methods have to be as efficient as possible and easy to handle. That means that a clear interface has to be provided that can be used in many different environments to let the scientists do their analyzes in the environment they are used to.

Finally there are two use cases for the access of data that have to be implemented:

- basic data access methods for concrete time intervals that will be used in analysis algorithms
- data browsing functionality

A solution for the first use case is presented in chapter 3 and an implementation of the second use case will be described in chapter 4.

2. Data organization

To understand the way data is accessed at W7-X one has to have basic knowledge of how data is organized first. All data at W7-X is stored in a central archive database in so called streams. Streams are sequences of data items that can have different types. Every item of a sequence is stored along with a 64bit time stamp. [2,3,4]

There are three types of streams:

- data streams that contain raw measured data
- parameter streams that contain parameter logs
- analysis streams that contain analyzed data

The type of an item in a data stream depends on the type of the stream and of course on the producer. Generally data streams contain sequences of single values or arrays with basic data type while parameter streams contain sequences of complex parameter structures. Analysis streams are very similar to data streams except that they additionally support versioning. All streams are identified by a group name (which is the name of the data acquisition station they were produced on) and the name of the data module that was responsible for acquiring the data values.

Streams in the database are logically linked. For every data stream in the database there are several parameter streams that describe the parameters for the device that delivered the data values at the instant they were acquired. Parameter streams are divided into two categories:

- configuration parameters:
  parameters that change whenever a change in the configuration of the experiment occurs (e.g. addition or removal of a hardware device) - this will likely happen at most once per experiment run
- segment parameters:
  parameters that can change during the discharge (e.g. the value for the gain of an ADC)

Configuration parameters change less frequently while segment parameters will change more often. So for every configuration parameter set there are one or more segment parameter sets that are valid during the same time interval (see Fig. 1).

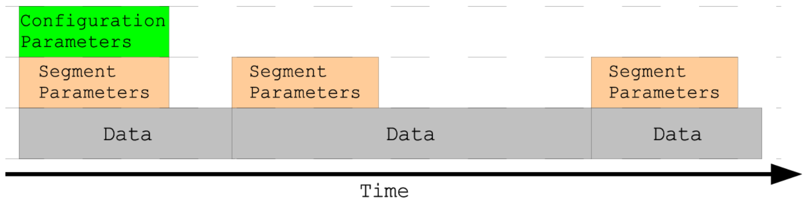


Figure 1

3. Data access interface

The software interface that has been developed is specially adapted to fit the requirements described above. It provides a set of classes and methods that allow easy access to any kind of data in the database. This chapter will describe these methods and the classes that are used as arguments.

3.1 Access functions

The data access process can be divided into two parts: the initialization part and the actual data access part (see Fig. 2).
The initialization part mainly consists of three function calls: the initialization of the database connection, the selection of input streams and the selection of a time interval.
The data access part consists of three nested loops. The first two loops are used to loop over configuration and segment intervals and the corresponding parameters as described in chapter 2. The third loop was introduced for performance reasons and is used to loop over time slices. The time slice concept will be explained in more detail in the next chapter.

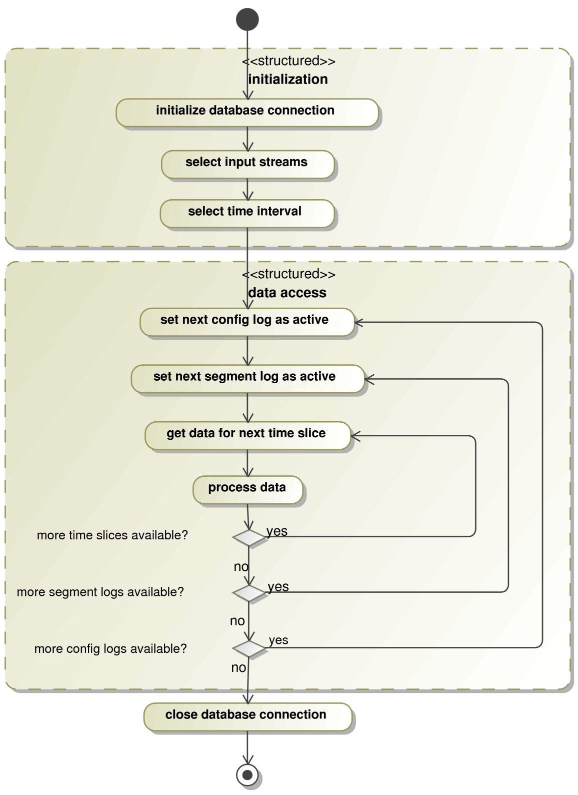

<<structured>>
initialization
initialize database connection
select input streams
select time interval
<<structured>>
data access
set next config log as active
set next segment log as active
get data for next time slice
process data
more time slices available?
yes
no
more segment logs available?
yes
no
more config logs available?
yes
no
close database connection

Figure 2

3.2 Time slices

When specifying a time interval during the initialization part it is possible to specify, besides the start and end of the interval, the size of a time slice. A time slice is a fraction of the actual time interval to be accessed (for instance the duration of a discharge) that can be handled efficiently. That

means it easily fits into memory and can be transferred over the network quite fast. The total time interval is thus divided into several subintervals according to changes of the configuration and segment parameters and the time slice size. In the data access part the software will then loop through these subintervals.
Although time slices slightly increase the complexity of the interface it is easy to see that they help to improve performance in software projects that want to handle long time intervals containing data measured with high frequency.

3.3 Selectors

To select the input streams a special set of classes has been defined that share a common interface (see Fig. 3). There are different concrete subclasses to select a specific stream or a set of streams that belong together.

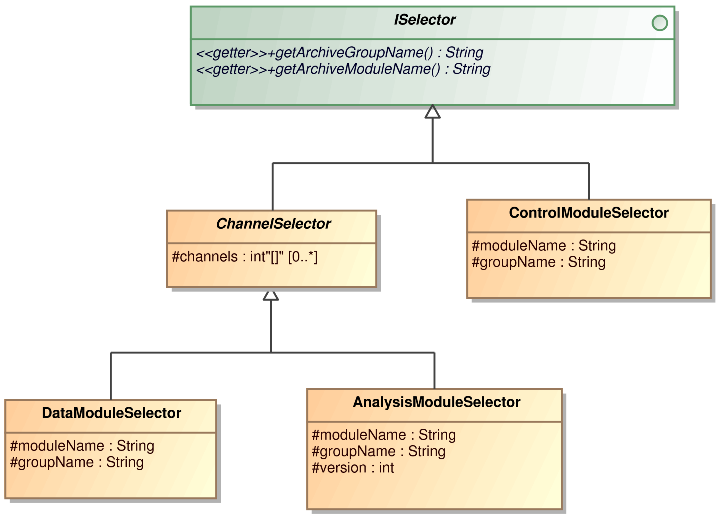


Figure 3

A *ControlModuleSelector* simply selects the parameter stream that belongs to this module while selectors of type *DataModuleSelector* and *AnalysisModuleSelector* select the data stream for the module and the corresponding parameter stream.
Every selector class uses a module name and a group name to identify the streams respectively the module that created the stream (see chapter 2). For data and analysis streams it is furthermore possible to select specific channels via the channels attribute of their common super class *ChannelSelector*. This attribute contains a list of indices that define the channels to be loaded. That way it is possible to avoid unnecessary processing of data values from channels that are not required for an analysis.
When selecting analysis streams the corresponding *AnalysisModuleSelector* provides a version attribute to select a specific analysis version. This is required because there may be different results for an analysis module when the implementation of the analysis algorithm changes. So as the version of the code changes the version of the calculated results has to change as well.

3.4 Dispensers

The data structures stored in the database are specially designed to be suited for fast and high-performance data acquisition. For this reason the structures are not always easy to understand and to handle for the normal user. Therefore the interface provides a set of adapter classes that allow easy access to the complex database structures.
These classes are all derived from a common super class called *DataDispenser*. The super class provides access to the time vector values and contains a static function to generate the correct

dispenser subclass for a given database object. At the lower inheritance levels there are several classes that represent the different data structures. At the moment there are mainly classes to access database objects that are structured as 1-, 2- or 3-dimensional fields of values with the same type (see Fig. 4). Here 1-dimensional fields are to be understood as structures that provide exactly one value for every time stamp, 2-dimensional fields as structures that provide an array for every time stamp (for example the values from different channels) and 3-dimensional fields as structures that provide a 2D array for every time stamp.

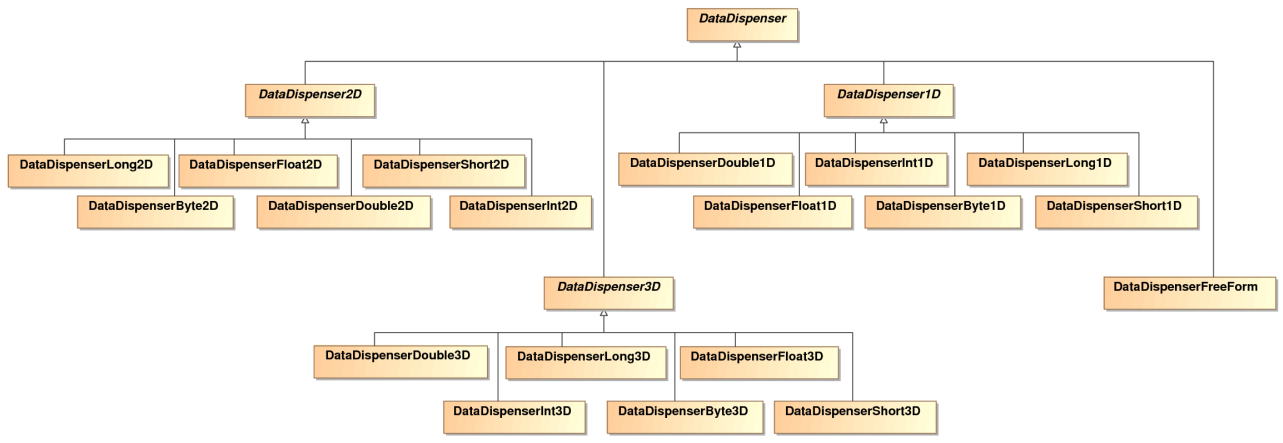


Figure 4

For database objects that contain a series of values with different types for every time stamp the class *DataDispenserFreeForm* can be used. It is also planned to provide dispenser classes to access image and video data as soon as this becomes necessary.

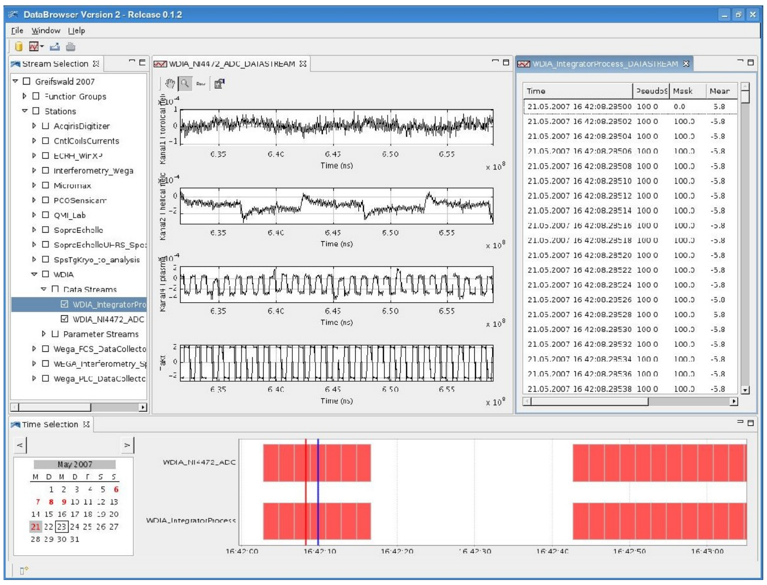
3.5 Usage in different environments

As mentioned before one of the goals during designing this interface was to make it accessible from many different platforms. This was one reason to implement the whole interface in Java. Thus it is not only possible to use these methods in other Java programs but also in many high level scientific programming languages which often provide a straight forward access to software implemented in Java. Two examples are IDL and MATLAB which have already been successfully used to access data via the described interface.
An example code in Java which looks very similar in other languages is shown below.

```java
// connect to database
TimeSlicedDataAccessRO dataAccess = new TimeSlicedDataAccessRO("Greifswald", "ThomsonScattering");
// select input streams
ISelector[] selectors = new ISelector[] {
  new DataModuleSelector("AcqirisDigitizer", "Acqiris_DataModule")
};
dataAccess.selectInputStreams(selectors);

// select time interval
long t1 = XDVTime.convertYYYYMMDDhhmmssToNanoseconds(2007,3,1,15,46,0,0);
long t2 = XDVTime.convertYYYYMMDDhhmmssToNanoseconds(2007,3,1,15,48,0,0);
long dt = (long)(60*1e9);
ConfigLogSet[] configLogs = dataAccess.queryData(t1, t2, dt);

// process data
for (ConfigLogSet configLog : configLogs) {
    dataAccess.setConfigLog(configLog);
    SegmentLogSet[] segmentLogs = configLog.segmentLogs();
    for (SegmentLogSet segmentLog : segmentLogs) {
        dataAccess.setSegmentLog(segmentLog);
        for (int islice=0; islice<segmentLog.nslices(); islice++) {
            DataDispenserByte3D dispenser =
                (DataDispenserByte3D)dataAccess.getDataDispenser(selectors[0], islice);
            analyze(dispenser.getTime(), dispenser.getByteValues());
}}}

dataAccess.close();
```

4. DataBrowser

As already mentioned the described interface provides a good way to access data in analysis algorithms when the time intervals to be processed are well known. To find interesting time intervals and get a rough overview of the acquired data a special browsing tool is required. To fulfill this task the DataBrowser has been implemented.
Because many features required by such a tool are standard features (e.g. wizard dialogs, help pages, internal window management, communication mechanisms between different GUI parts) the software has been implemented as a plugin to the Eclipse framework that already provides solutions for many of these requirements. [5]



Time	PseudoS	Mask	Mean
21.05.2007 16 42:08.28500	100 0	0.0	5.8
21.05.2007 16 42:08.28502	100 0	100.0	-5.8
21.05.2007 16 42:08.28504	100 0	100.0	-5.8
21.05.2007 16 42:08.28506	100 0	100.0	-5.8
21.05.2007 16 42:08.28508	100 0	100.0	-5.8
21.05.2007 16 42:08.28510	100 0	100.0	-5.8
21.05.2007 16 42:08.28512	100 0	100.0	-5.8
21.05.2007 16 42:08.28514	100 0	100.0	-5.8
21.05.2007 16 42:08.28516	100 0	100.0	-5.8
21.05.2007 16 42:08.28518	100 0	100.0	-5.8
21.05.2007 16 42:08.28520	100 0	100.0	-5.8
21.05.2007 16 42:08.28522	100 0	100.0	-5.8
21.05.2007 16 42:08.28524	100 0	100.0	-5.8
21.05.2007 16 42:08.28526	100 0	100.0	-5.8
21.05.2007 16 42:08.28528	100 0	100.0	5.8
21.05.2007 16 42:08.28530	100 0	100.0	-5.8
21.05.2007 16 42:08.28532	100 0	100.0	-5.8
21.05.2007 16 42:08.28534	100 0	100.0	-5.8
21.05.2007 16 42:08.28536	100 0	100.0	-5.8
21.05.2007 16 42:08.28538	100 0	100.0	-5.8

Figure 5

4.1 Data selection

For browsing acquired data similar to the interface described before the two orthogonal selections of the input streams to be viewed and the time interval must be made. The difference to the basic software interface is that the selection parameters are not initially known but will be picked from the amount of all available information.
That means that the software has to display all streams and all valid time intervals stored in the database first and then provide selection methods to choose the interesting streams and time intervals interactively.
To realize these selection mechanisms the views extension point of the Eclipse framework has been implemented two times. The first implementation displays all streams available in the database in a tree view. Here the root node is the database that is browsed. Below the root node the groups that combine streams belonging together are displayed. For a better overview the group nodes are split in two categories corresponding to their origin: All streams produced by a distinct station are put in the group node of their station below a node called “Stations”. All streams not produced by a specific station but by software components distributed over the network (for instance the control program of a function group) are put in the group node of the function group they belong to below a node called “Function Groups”.
Inside the group nodes another separation is made to enhance the view: All data streams are put in a common node called “Data Streams” and likewise all parameter streams are put in a node called “Parameter Streams”.
To mark a stream as active to the other parts of the program the checkbox next to the tree node representing the stream has to be selected. A whole set of streams can be selected by using the checkbox next to their common parent node. So all streams belonging to a station can be activated

by clicking on the checkbox next to their station node.
The second selection view, the time interval selection, is implemented using a calendar view and a task diagram. The calendar view displays the month that was chosen using the forward and back buttons on top of the calendar view. All days at which data for the selected streams is available are marked red. If a specific day or a set of days is selected with the mouse the corresponding time interval will be displayed in the task diagram.
Every selected stream is displayed in the task diagram as one task. The time intervals that contain acquired data are marked as red horizontal bars. It is possible to zoom the time axis with the mouse to get a closer look at the valid time intervals. A click with the left mouse button inside the task diagram sets the beginning of the time interval selection and a click with the right mouse button sets the end. Start and end of the selected time interval are shown as a red and a blue vertical line.
To inform other program parts about selection changes the selection service provided by the Eclipse framework is used. Thus all Eclipse parts interested in the stream or time selection can listen for the corresponding selection change event without the selection view knowing each of the listeners. [6]

4.2 Data viewing
After a selection is made the selected data has to be presented to the user. As we have different data types we also need different plots to display them. It is also sometimes reasonable to provide more than one possibility to view the same type of data. For instance one may want to view an acquired time trace one time as a graph and another time as a table containing the concrete values.
Several plots implementing the "editors" extension point of the Eclipse framework were created to accomplish this task. There is currently one plot to display 1D and 2D data (which as before means one value or a vector of values per time stamp), one plot to display 3D data and one "plot" that displays the concrete values in a table view. The implementation of a "plot" to display image and video data has been started.
The extension point concept of the Eclipse framework allows easy addition of further plots. As soon as the requests arise more plots will be added.

4.3 Data export
Sometimes it may be necessary to save data that has been read from the central database into local files to process these files with external programs. For this reason the Eclipse "export" extension is used to provide special export functions. At this time only the export to text files has been implemented but it is easily possible to add other export functions.

5. Conclusion
The described concepts have been implemented and tested from a developers point of view. The browsing functionality still needs improvement when handling huge data intervals and the data representation in the graphical user interface has to be optimized to fit the requirements of the diagnosticians. Although there are already some physicists that use the provided tools in laboratory setups these requirements will become more clear when the W7X control and data acquisition prototype will start operation. Hopefully the diagnosticians will take advantage of the possibility to use this test bed installation as preparation for W7X by learning the usage of the provided software on the one hand and improving the software by giving experienced feedback on the other hand.

6. Acknowledgment
The authors would like to thank the control group and W7-X software group for their support and for all the helpful discussions on the complex topics.

References

[1] J. Schacht et al., WEGA as a test-bed for the WENDELSTEIN 7-X control system (this conference)

[2] P. Heimann, S. Heinzel, Ch. Hennig, H. Kroiss, G. Kühner, H. Kühntopf, J. Maier, J. Reetz, M. Zilker, The data acquistion system W7-X, Proceedings of the 3rd IAEA Technical Commitee Meeting on Steady-State Operation of Magnetic Fusion Devices, 2-3 May 2002, Greifswald, Germany, 6-7 May 2002, Arles, France

[3] P. Heimann, S. Heinzel, Ch. Hennig, H, Kühntopf, H, Kroiss, G. Kühner, J. Maier, J. Reetz, M. Zilker, Status report on the development of the data acquisition system of Wendelstein 7-X, Fusion Engineering and Design, 71 (2004), 219-224

[4] P. Heimann, T. Bluhm, Ch. Hennig, H. Kroiss, G. Kühner, J. Maier, H. Riemann, M. Zilker, Database Structures and Interfaces for W7-X (this conference)

[5] Eclipse Homepage: http://www.eclipse.org

[6] B. Daum, Java-Entwicklung mit Eclipse 3.2. Anwendungen, Plugins und Rich Clients, 2006