A service based interface for scientific data retrieval

Torsten Bluhm[a,*], Sven Jacob[c], Andreas Werner[a],
Peter Heimann[b], Christine Hennig[a], Georg Kühner[a],
Hugo Kroiss[b], Heike Laqua[a], Marc Lewerentz[a], Josef Maier[b],
Heike Riemann[a], Jörg Schacht[a], Anett Spring[a], Manfred Zilker[b]

31st August 2010

[a] *Max-Planck-Institut für Plasmaphysik, EURATOM Association, Teilinstitut Greifswald, Wendelsteinstraße 1, D-17491 Greifswald, Germany*
[b] *Max-Planck-Institut für Plasmaphysik, EURATOM Association, Boltzmannstraße 2, D-85748 Garching, Germany*
[c] *Fachhochschule Stralsund, Zur Schwedenschanze 15, D-18435 Stralsund, Germany*
[*] *Corresponding author; Tel.: +49-3834-88-2238 E-mail address: Torsten.Bluhm@ipp.mpg.de*

Abstract

Retrieval of scientific data at large fusion experiments is an important but surprisingly complicated task. There are many different data storage systems that have to be accessed using different software interfaces on varying platforms. With MDS+ a respectable approach towards a more standardised and simplified data access interface was started. However, it has its drawbacks regarding continuously acquired data and provides a whole data storage system rather than only an interface, which is not always demanded. Therefore a data access interface based on service oriented technologies has been developed aiming for a more standardised and platform independent solution. The contribution will describe the current status of this development. It will explain the interface itself including the WSDL (Web Service Description Language) definition and show currently existing service implementations for the Wendelstein 7-X and the Wendelstein 7-AS data storage systems. Furthermore different possibilities to access the services on the client side will be discussed.

Keywords: data access, web service, SOA, steady state

1 Motivation

1.1 Requirements

A continuous data acquisition system like the one developed for Wendelstein 7-X (W7-X) is demanding for the corresponding data retrieval methods. On the one hand retrieval of short time intervals (<1s) must be provided, where data is acquired with high sampling rates. On the other hand it must also be possible to handle very long time intervals that span several days or even weeks and give an overview for slowly developing values. For diagnostics of the operational management it is also necessary to access arbitrary time intervals completely independent from some discharge identifier like a shot number. Hence, the data values must be addressed using an absolute time scale.

Steady state and long pulse data acquisition requires consideration of changing hard- and software parameters during data analysis.[1, 2] It must be guaranteed that data values are always processed together with the parameters that have been active at the time of the data acquisition. This means data acquisition intervals have to be synchronised with the validity intervals of the corresponding parameters.

Another problem with processing data from large intervals is that there is always only limited RAM available on the data processing unit. Therefore, it is necessary to divide the intervals into smaller portions and process them one by one during data retrieval and analysis.

Furthermore there are more technical requirements concerning the hard- and software that is used to analyse data. This is even more challenging if the goal is to generalise the interfaces not only for one experiment but for multiple fusion experiments. There are different runtime platforms involved (single PCs, clusters, grids) with different operating systems (Windows, Linux, Solaris, AIX). Different programming languages are used to access different data storage systems. Satisfying all these requirements is a quite complex task and also forces some compromises.

Finally the data acquisition and storage system at W7-X has some special requirements. At the moment it is not possible to access more than one of the so called Objectivity federations which are used for data archiving at the same time within one process.[3] This makes it difficult to switch between productive and test systems for example. Besides that users also have to include a number of different libraries into their analysis codes that are necessary to access an Objectivity federation. It must be taken care of that the libraries have the correct version and that they are accessible in different runtime environments. Moreover there are many analysis codes that are developed based on data measured at Wendelstein 7-AS (W7-AS), Asdex Upgrade or even WEGA[4]. It would save a lot of work if these codes could be developed using the same data access interface like W7-X. This would make the effort of porting these codes as minimal as possible and also allow the validation of codes with data from multiple experiments.

1.2 Services - the solution?

The concept of Web Services [5] seems to meet all of the requirements described above.[6] It provides an abstract interface definition independent of runtime platform, programming language and data storage system. The usage of web services does not produce any additional dependencies and it still allows very flexible programming due to the ever improving support by programming languages and the corresponding development environments.

2 Data Access Service Interface

A basic principle during the development of the described interface was the Contract-First approach. This means that the service interface was first specified in the form of a WSDL (Web Service Description Language) specification [7] independent from any programming language. All implementations on the client side as well as on the server side have then been derived from this specification. The WSDL specification describes the available methods for the service and the data types used by these methods. The following chapters will give a brief overview about the provided methods.

2.1 Logic of accessing continuous data

To understand the task of the provided methods it is necessary to understand the internal logic of the described data access interface first.

Handling configuration & experiment parameter changes

As mentioned before accessing continuously acquired data requires some special conditions to be met. One problem is that due to the segmented control concept parameters that are relevant for the acquisition and measurement of data may change during a time interval. However, for analysis algorithms it must be assured that the correct parameters for every data interval are accessible. It is therefore necessary to subdivide the demanded time interval and iterate over these sub intervals in order to work with consistent parameters.

Generally we assume that there are two kinds of parameters:

- Configuration parameters
  These parameters change on a slow time scale and concern mostly hardware related settings like cabling, board identifiers etc. They will not change during a discharge.

- Experiment/segment parameters
  These parameters change more frequently and can also change during a discharge (to set a new gain for example).

To read correctly parametrised data this implies that we have an outer loop to iterate over the configuration changes and an inner loop to process the segment changes inside a configuration interval. For the combined access of more than one data source the configuration and segment intervals must be synchronised and provided as one common sequence of parameter intervals. Thus we can ensure that always a data array with constant parameters is processed in the innermost loop.

Handling large data intervals

As we expect intervals to be accessed that cover long time periods or contain measurements with high sample rates and channel numbers, it is possible that not all data can be loaded and stored into memory at once. To avoid memory overflow errors the data access interface provides a mechanism to split a data interval into smaller sub intervals (slices) that can be retrieved from the data archive one after the other. To accomplish this a third loop inside the segment loop described above is necessary which iterates over the slices of one segment interval.

2.2 Service Methods

The data access concept described above leads to a set of functions that define the interface of the data access service. The most important of these methods will be described below.

open()

The *open* method initialises a new service session. The result of this method is a data structure that holds a reference to the current session. This reference has to be passed to all subsequent method calls to identify the session. Because the Web Service Addressing specification of the W3C provides such a standard in the mean time this concept will most likely be replaced with this more standardised technique.[8]

close()

The *close* method is the counterpart to the open method and cleanly ends a service session. No further method calls for the specified session will be accepted after a call to *close*.

select()

This method selects the data sources that will be accessed during the session. The specification of the data sources prior to the actual data access is necessary to synchronise potential parameter changes as described before. Because there is no general identification for data sources the arguments to this method are simply hierarchic structured key/value pairs. It is the task of the service implementation to interpret these descriptors. Some examples will be given in chapter 3.1 and 3.2.

getConfigurationIntervals()

With this method the configuration intervals inside a given time interval are retrieved. The parameters for this method are two 64 bit long values specifying the start and the end of the interval of interest. These values are the nanoseconds since 1.1.1970. The result of this method is a list of data structures that describe intervals with unchanging configuration parameters. This list will be used for the outer data access loop.

getSegmentIntervals()

This method also returns a list of data structures similar to the method before with the difference that these structures describe intervals with unchanging segment parameters inside a configuration interval. Therefore the arguments of this method include the id of one configuration interval. Furthermore the size of one data slice has to be passed to this method. This argument can be used to control the amount of memory that is necessary for one analysis step. Similar to the start and end parameters above this value is a 64 bit long value specifying the size of one data slice in nanoseconds.

accessData()

accessData finally is the actual method to retrieve the data values from the data archive. The arguments of this method are three IDs that specify the configuration interval, the segment interval and the slice index and one of the selector arguments already used with the select method to define the data source. It returns a data structure that serves as a container to the transferred data. The container holds a vector of time values, a (multidimensional) array of data values and some meta information that give additional information about the structure of the data values. For performance reasons the data values are transferred as binary values. This means that the client has to decode the binary data block into the concrete type using the provided meta data.

getConfigurationParameters() & getSegmentParameters()

Besides the method to retrieve data from the data archive these two methods can be used to retrieve specific configuration and segment parameters for a given time interval. Their arguments include ids to define the configuration and segment intervals and a description of the parameter that is to be retrieved. The result is an abstract parameter description similar to the arguments of the *select* method.

tagToInterval()

As described earlier the data access methods use absolute time values to specify experiment phases. Because this is not common practise at most fusion experiments the *tagToInterval* method provides a way to map shot numbers or similar identifiers for experiment phases to time intervals. The argument to this method is simply a string which represents a shot id for example. The service must be able to interpret this string and map it to a time interval respectively its start and end time. These two time values are returned by the method and can be used as arguments to *getConfigurationIntervals*.

3 Data Access Service Implementation

Currently there are two service implementations that are already in use at W7-X. There is one implementation to access the data that is currently acquired at prototype installations and test setups at W7-X. The other implementation accesses the shot files of the W7-AS experiment and provides access to these kind of data with the same interface that will be used for later W7-X experiments.

3.1 For Wendelstein 7-X

The service implementation for W7-X is a Java based application that accesses the central Objectivity archive database. It was developed using the JAX-WS framework.[9]

Because the data archive is structured by so called data streams and groups of these data streams the selection of data sources is defined by two unique ids that describe the stream group and the stream. These ids are passed as string values with specific keys to the select method of the data access service.

As all data at the W7-X experiment is measured together with an absolute time value a translation of a discharge descriptor to a time interval is not necessary. However, it might be necessary in the future to introduce descriptors for discharges and sub intervals of a discharge to identify specific experiment phases. In that case such a mapping method could be added easily.

3.2 For Wendelstein 7-AS

For the implementation of the data access service for W7-AS shot files the gSOAP framework was used.[10] Thus it was easily possible to integrate the Fortran/C++ based methods to access the shot files.

Unlike the W7-X service implementation the W7-AS data storage system uses diagnostic and module names for the identification of data sources. Again these two names are passed as strings with corresponding keys to the select method.

Because W7-AS is a short pulse experiment most of the parameters do not change during a discharge. Those parameters are handled as configuration parameters and hence all shot intervals are handled as configuration intervals. However, there are some timing related parameters that can change during a shot. These parameters are used as segment parameters and changes of these parameters mark segment intervals.

To realise the mapping between shot numbers and absolute time values the start time contained in every shot file has been used. Because it is not possible to dynamically search the start time of a shot on every data access a mapping table was created that contains all shot numbers and the corresponding start times.

4 Data Access Clients

4.1 The standard way

The usual way to access data via one of the service implementations is to:

1. retrieve the WSDL file from a well known server
2. generate the client stub using the WSDL file
3. implement the client functionality using the generated stub methods

There are stub compilers for many programming languages such as Java, C/C++, C#, Python, Matlab and others. Unfortunately not all implementations are completely W3C conform yet. There are still problems using the SOAP message style document/literal in Matlab for example.

Because data values are transferred as binary data for performance reasons the client has to decode the binary data field. This can be accomplished using the meta information transferred together with the data values. The meta information contains values like the basic data format (byte, short, float, ...) and the number of channels.

4.2 Prepared clients for easier access

The handling of stub compilers and of the code they produce can be quite difficult for the inexperienced user. Furthermore most users will create identical stubs because they are working on similar platforms and use similar development environments. We provide, therefore, a collection of small libraries containing a prepared version of the client stub and some utility methods for common platforms. At the moment this has been done for Java and C++.

Java client library

Java has been chosen because it is widely spread and can be used in Java applications and also in higher level scientific languages like IDL or Matlab. The client library is developed using the JAX-WS framework and provides wrapper methods for most of the service methods. The wrapped methods already include the decoding of the binary data fields and provides special container classes for the different data types that have to be handled. Besides that they simplify the selection of data sources by providing special selector classes with an easy-to-use interface.

C++ client library

The C++ client library was developed using the gSOAP framework. Like the Java library it provides wrapper methods for the service methods that already provide decoding of the binary data values. The gSOAP framework can also be used to create a plain C client. With such a client it would be possible to also support legacy codes implemented in Fortran.

5 Summary & Tests

The implemented services have been installed on servers at W7-X and run very stable. First user experiences were made accessing data from the W7-X Objectivity archive using the Java client implementation in Matlab or IDL. There was no negative feedback although it has been observed that users have to get used to the way of accessing data with the developed interface (loading data in smaller slices and analysing the slices step by step rather than loading a whole shot and analyse the data all at once). The complete independence of the client from any data archive related library has been proven to be very useful. It was not necessary to install any database client or similar software on the client systems. The next step in development will be to extend the interface with methods that allow the user to store data.

To test the performance of the interface a few measurements have been made using the W7-X service and the provided Java client library. For these tests data volumes of different sizes have been transferred from the data archive to the test client and the time consumption of the operation has been measured. The data volume used to calculate the data rate was the actual data volume which means the sum of the 64bit timestamps and the 4 byte data values for each channel. The overhead produced by the service and the protocol was ignored. As a result an actual data rate of about 3 MB/s could be measured without any optimisation which is a quarter of the total bandwidth (100 Mbit/s ~ 12 MB/s). The data rate differs depending on the slice size. The higher the size of the data slices the higher was the measured data throughput. Although the data rate could be better up to now there was no real need to optimise the data transfer because analysis algorithms often include long running calculations anyway.

References

[1] P. Heimann, T. Bluhm, Ch. Hennig, H. Kroiss, G. Kühner, J. Maier, H. Riemann, M. Zilker, Database structures and interfaces for W7-X, Fusion Engineering and Design, Volume 83, Issues 2-3, Proceedings of the 6th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research, April 2008, Pages 393-396, ISSN 0920-3796, DOI: 10.1016/j.fusengdes.2007.08.008.

[2] H. Riemann, T. Bluhm, P. Heimann, C. Hennig, G. Kühner, H. Kroiss, H. Laqua, M. Lewerentz, J. Maier, J. Schacht, A. Spring, A. Werner, M. Zilker, From a physics discharge program to device control–Linking the scientific and technical world at Wendelstein 7-X, Fusion Engineering and Design, In Press, Corrected Proof, Available online 15 January 2009, ISSN 0920-3796, DOI: 10.1016/j.fusengdes.2008.12.012.

[3] Objectivity Technical Overview Release 9, Copyright 1993–2009 by Objectivity, Inc., available at `http://www.objectivity.com`

[4] J. Schacht, D. Aßmus, T. Bluhm, A. Dinklage, S. Heinrich, C. Hennig, U. Herbst, R. König, H. Laqua, M. Lewerentz, I. Müller, M. Otte, S. Pingel, J. Sachtleben, A. Spring, A. Werner, A. Wölk, Stellarator WEGA as a test-bed for the WENDELSTEIN 7-X control system concepts, Fusion Engineering and Design, Volume 83, Issues 2-3, Proceedings of the 6th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research, April 2008, Pages 228-235, ISSN 0920-3796, DOI: 10.1016/j.fusengdes.2007.09.016.

[5] W3C Web Service Activity: `http://www.w3.org/2002/ws/desc`

[6] Sven Jacob, Entwicklung einer einheitlichen Schnittstelle für den Zugriff auf Fusionsexperimentdaten, Master Thesis, Fachhochschule Stralsund, May 2009

[7] W3C Web Service Description Language Specification: `http://www.w3.org/2002/ws/desc`

[8] A. Werner, J. Svensson et al., Integration framework for W7-X using service oriented GRID middleware, this issue

[9] Java API for XML Web Services: `https://jax-ws.dev.java.net`

[10] gSOAP Toolkit for SOAP Web Services and XML-Based Applications: `http://gsoap2.sourceforge.net`