Fusion Engineering and Design 127 (2018) 106–110

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

MDSplus yesterday, today and tomorrow

T. Fredian[a,*], J. Stillerman[a], G. Manduchi[b], A. Rigoni[b], K. Erickson[c], T. Schröder[d]

[a] Massachusetts Institute of Technology, 175 Albany Street, Cambridge, MA, 02139, USA
[b] Consorzio RFX, Euratom-ENEA Association, Corso Stati Uniti 4, Padova,35127, Italy
[c] Princeton Plasma Physics Laboratory, Princeton, NJ, 08543, USA
[d] Max-Planck-Institut für Plasmaphysik, D-17491, Greifswald, Germany

ARTICLE INFO

Keywords:
Data acquisition systems
Data management
Data formats
MDSplus

ABSTRACT

MDSplus is a data acquisition and analysis system used worldwide predominantly in the fusion research community. Development began 31 years ago by a collaboration of software developers who were charged with providing a data acquisition system for three new fusion experiments under construction: CMOD at MIT, ZTH at LANL and RFX at Padova, Italy. The design of MDSplus combined the functionality of MDS (MIT/Model Data System developed at MIT for the Alcator and Tara fusion experiments) with new features suggested by the developers from the other laboratories. The development of MDSplus used a RAD (rapid application development) approach before RAD became a mainstream methodology. MDSplus was implemented and ready for the initial operation of CMOD in 1991. Since that time, many other fusion facilities started using MDSplus for data acquisition and/or for exporting their data to other sites. Today MDSplus is still used around the world for fusion energy research, space exploration and other fields of science and technology. Work on MDSplus continues to enhance its capabilities, support more platforms, and improve its reliability. It is anticipated that MDSplus will continue to provide valuable tools for the fusion energy research community. This paper describes some of the history of the MDSplus software, the work that is currently underway, and the plans to enable MDSplus to continue to be available and supported long into the future.

1. Introduction

MDSplus [1,2] is a Framework for data acquisition and management of scientific data. It consists in a collection of libraries and applications used for acquisition, access and storage of scientific data. MDSplus functionality can be summarized as follows:

- Data organization and storage: the MDSplus data access layer defines a variety of data types for scientific applications. Data types such as scalars and arrays are enriched with derived types such as signals, to represent both samples and timestamps. Metadata can be defined, e.g. to associate a description or a validation procedure to a given data item. Hierarchical collections of data items can be stored in pulse files, possibly distributed across multiple computers. Pulse files are created by cloning a template model, called experiment model, that defines the data structure and contains as subset of data items, that is, the description of the experiment configuration and all the information about the experiment that is known in advance. The cloned pulse file will be enriched with experimental data collected during the experiment sequence. Other data systems, such as HDF5, provide support for hierarchical data organization for a variety of data types, but MDSplus provides two unique features: the use of generic expressions data representation and Multi Reader/Multi Writer (MRMW) ability. Unlike the other data system where a data item is an instance of a given data type, in MDSplus every data item is an expression, that is, a program that returns a given data type. Expressions can be as simple as the definition of a scalar value, or be represented by hundreds of lines of code specifying how the returned data item is built, based on other information, normally stored in the pulse file itself. Internally. Expressions are stored in pulse files as the parse tree of the corresponding program (in case of a simple data, the parse tree becomes the data item itself), and every time a data item is read in MDSplus, the corresponding expression is evaluated on the fly. Expressions add an expressive power in data representation that cannot be provided by other data systems and they are widely used to describe the whole acquisition chain for acquired signals and provide a way to dramatically reduce the size of pulse files by eliminating redundant data. For example, a set of signals acquired using the same timebase can refer to a single timebase item in their expression definition instead of storing the

* Corresponding author at: Massachusetts Institute of Technology, 175 Albany Street, Cambridge, MA, 02139, USA.
E-mail address: twf@psfc.mit.edu (T. Fredian).

https://doi.org/10.1016/j.fusengdes.2017.12.010
Received 6 June 2017; Received in revised form 30 October 2017; Accepted 11 December 2017

whole time array for every signal.

- MRMW ability is another unique feature of MDSplus. Data systems supporting structured data typically preserve integrity in data access by limiting file access to a single writer, and often even multiple readers are not allowed. This represents a serious limitation in large data acquisition systems where different actors store data concurrently in the pulse file. MRMW is provided in MDSplus using a locking mechanism that is implemented in the lowest data access layers and is therefore available to all the MDSplus components.
- Access to the data and metadata from a wide variety of programming languages and utilities. MDSplus provides Application Programming Interfaces (APIs) in a variety of programming languages including C + +, Java, Python and Matlab. The same data semantics are defined in all APIs, expressed as a set of classes providing methods that allows all the expressive power of MDSplus to be exposed to users. Users can access data in a straight way, just calling the data() method that evaluates the corresponding expression on the fly, getting the data item they are interested in. More experienced users can develop their own expressions to efficiently organize data semantics.
- Remote data access using a variety of transport mechanisms. This is another unique feature of MDSplus, that is, the possibility of acting on remote data preserving the same interface as for local data. In this way the implementation of remote data acquisition systems is greatly simplified. The developer does not need to worry about data transport. This is achieved by implementing in the lowest data access layers a transport mechanism based on a specific high level protocol, called mdsip, initially built over TCP/IP and currently available in a variety of transmission protocol. Remote data access using mdsip turns out to be much more efficient than using other protocols for distributed files such as NFS. The reason is that, unlike general purpose distributed file systems, network traffic in mdsip is optimized and targeted to the specific MDSplus requirements in remote data access. This configuration, called *Distributed Client*, is enabled by setting a few environment variables that specify whether data access is local or remote and where data are stored. Another remote data access configuration supported in MDSplus is called *Thin Client*. Applications send expressions to a server for I/O and evaluation and results are sent back to the client. This optimizes the number of network transactions for data access and it is best suited for remote data access in Wide Area Networks where latency in transaction becomes an issue. Thanks its flexible and efficient remote data access layer, MDSplus is now the standard de facto for remote data access in fusion experiments.
- Data-driven workflow engine for data acquisition and automated analysis. Unlike traditional data systems that store configuration data and acquired signals in separate databases, in MDSplus the same pulse file stores all the experiment-related information, not only including configuration and experimental data, but also the description of the actions carried out during the experiment sequence and of the associated scheduling information. The Action data type specifies *what* should be executed (a program, a procedure, and I/O routine, etc), *when* it should be executed (i.e. its relationship with other actions in the sequence) and *which* actor should execute this action.
- Integration of user-provided components. MDSplus provides the *device* pattern that is a set of classes and rules that allow users integrate specific data acquisition components. Such components, normally interacting with specific hardware, can be integrated in the framework and becomes an integral part of MDSplus. For example, they are recognized by the workflow engine and the corresponding actions scheduled during the experimental sequence.

This paper will discuss the history leading to the development of the MDSplus data system, the current work underway to improve the quality and functionality of the software and the prospects of MDSplus continuing to be a useful tool for fusion energy research in the future.

2. The history of MDSplus

2.1. First there was MDS

In 1982, two MDSplus authors, Tom Fredian and Josh Stillerman, joined the Alcator C fusion experiment team at the Plasma Science and Fusion Center (PSFC) at the Massachusetts Institute of Technology. They were tasked with providing computing assistance for engineers and scientists of the two experiment groups.

The techniques for acquiring and storing data performed by scientists and engineers at the Alcator C experiment were very primitive compared with today's standards. The main source of experimental data was through digitization of 128 signals recorded on a large analog storage drum using a single CAMAC digitizer to digitize four signals at a time and write the data to a disk file. In addition to the analog drum a few other CAMAC digitizers had been used to record measurements. For this data the scientist responsible for the measurements developed or copied and modified a specialized program that would read the digitizer and store the data into a disk file in a format only known by the scientist who wrote the program. The other commonly used mechanism for storing measurements was the use of a Polaroid instant camera to take photos of oscilloscope displays which were then often digitized by hand by scientists.

Based on the previous experience with process control system software they started developing a data system where the data could be stored in a central database and the scientists and engineers could use a common set of tools to record and access experimental data. This new system was later given the name MDS [3] which originally stood for Model Data System but was later called MIT Data System.

The MDS data system was developed entirely using the FORTRAN programming language to be used on the Digital Equipment VMS operating system. Digitized data from CAMAC modules along with timestamp information were stored in a datafile and could easily be accessed using a unique signal name given to each measurement. Software modules written to read a particular type of CAMAC digitizer were developed to enable a scientist to configure data acquisition with the digitizer simply by specifying a few device specific settings, the signal names to use for the channels being recorded and some timing information to construct a timebase for the measurements. In addition, the system provided a simple mechanism to do a linear conversion with a coefficient and offset to convert digitizer counts into measured voltages. With MDS, all of the scientists and engineers could access all of the measurements recorded after each pulse of the experiment using the same software and interactive tools for visualizing the recorded signals. Some other features of MDS are worth mentioning. The data access code of MDS supported concurrent multiple data writers and readers. This allowed new data to be stored by multiple programs simultaneously at the same time as previously stored data was being accessed by users. MDS also provided a lossless compression of the data greatly reducing the disk storage requirements. The system relied on dynamic image activation and call by name to provide extensibility, avoiding the need to recompile/relink the distributed code.

The MDS system developed in less than two years turned into a very useful tool for the scientists and engineers at MIT and soon many other fusion research facilities around the world learned of MDS and began using it on their experiments as well. By 1987 MDS was in use at PPPL, ORNL, UCLA, SNL, U of Washington, Columbia University and U of Wisconsin (United States), ANU (Australia), NIFS (Japan), KTH (Sweden), and NFRI (South Korea).

Much of the acceptance and success of MDS was that it was a collection of tools that could be used by any fusion experiment facility requiring only site specific configuration setting modifications without modifying the MDS software.