EPICS V4 EXPANDS SUPPORT TO PHYSICS APPLICATION, DATA ACSUISITION, AND DATA ANALYSIS*

L. Dalesio, Gabriele Carcassi, Martin Richard Kraimer, Nikolay Malitsky, Guobao Shen,
Michael Davidsaver, BNL, Upton, Long Island, New York, U.S.A
Ralph Lange, Bessy, Berlin, Germany
Matej Sekoranja, Cosylab, Ljubljana, Slovenia
James Rowland, Diamond Light Source, Oxfordshire, England
Greg White, SLAC, Menlo Park, California, U.S.A.
Timo Korhonen, PSI, Villigen, Switzerland

Abstract

EPICS version 4 extends the functionality of version 3 by providing the ability to define, transport, and introspect composite data types. Version 3 provided a set of process variables and a data protocol that adequately defined scalar data along with an atomic set of attributes. While remaining backward compatible, Version 4 is able to easily expand this set with a data protocol capable of exchanging complex data types and parameterized data requests. Additionally, a group of engineers defined reference types for some applications in this environment. The goal of this work is to define a narrow interface with the minimal set of data types needed to support a distributed architecture for physics applications, data acquisition, and data analysis.

INTRODUCTION

Version 3 of EPICS [1] consisted of three key components: the narrow client interface that presented channels of various scalar data types with metadata for display and control, a robust and high performance network protocol for these data types, and a process database that allowed the definition of control logic to achieve steady state control. Many clients were developed to provide batch, SCADA, and DCS capabilities to physics research facilities. Version 3 offered little to help develop application for data acquisition, model based control, or experiment control. Many facilities developed monolithic programs with some general libraries to build toolsets to accomplish these functions. Version 4 of EPICS, features an extended data type definition and a protocol that can serialize and de-serialize these data types to extend the narrow interface to develop services above the instrumentation level to support these functions. This paper will explore the extended interface that is being presented to the services and clients to support this functionality and the middle layer services that are planned to work in this protocol to support a modular architecture to support data acquisition, experiment control, and model based control.

VERSION 3 CHANNEL TYPES

The version 3 channel types included the basic scalar data types along with metadata to make the value useful in a process control system. These meta data included a time stamp, information that allowed the channel to be displayed, alarmed, or controlled. The time stamp is the primary mechanism for identifying when the value was read. This enables us to look for correlations between various system parameters. It also included an alarm severity that identified a channel that was not valid, or in an alarm condition. The metadata also included information to inform a client how to display a value with either display limits, display ranges and engineering units for analog type values, state labels for discreet signals, and dimensions for strings or array [2]. It also included information to understand the limit of control for analog settings and discreet outputs. These types provided reasonable functionality for device integration. They provided little to no support for data acquisition, model based control, and experiment control where the channels needed to be more complex and dense. Version 4 core allows the definition of any data structure to be serialized and transported [3]. This version defines a set of general and specific normative types to support these functions. This paper will discuss this set of normative types and their application in the middle layer services that are being developed to support this new scope.

VERSION 4 MIDDLE LAYER USE CASES

When we study the physics toolkits that have been developed such as SDDS [4], Matlab Middle Layer Toolkit [5], and XAL [6], it becomes clear that several utilities are always available. There is always a description of sets of EPICS Process Variable that are predefined for lists of channels that need to be operated on or displayed. This is specified as two services in version 4: Directory Service and Gather Service. These tools also have need to acquire information from a static data store for alignment data, magnet mapping data, and other static data that is in a relational database. These sets of data fall into several categories: named value pairs, or sets of values such as coefficients. In addition, data acquisition and model based control required large vectors of data or multidimensional arrays, or images. Data over large periods of time are also useful in this environment. It is these use cases, that drive the creation of the extended set of data types that will be exposed through the version 4 PVAccess API In addition to the extension of the data types to support these functions, The version 4 PVAccess server has been connected to the

EPCIS version 3 database to serve this data on the version 4 protocol as the first set of General Normative Types: NTFloat, NTDouble, NTEnum, NT String, etc. See Figure 1.

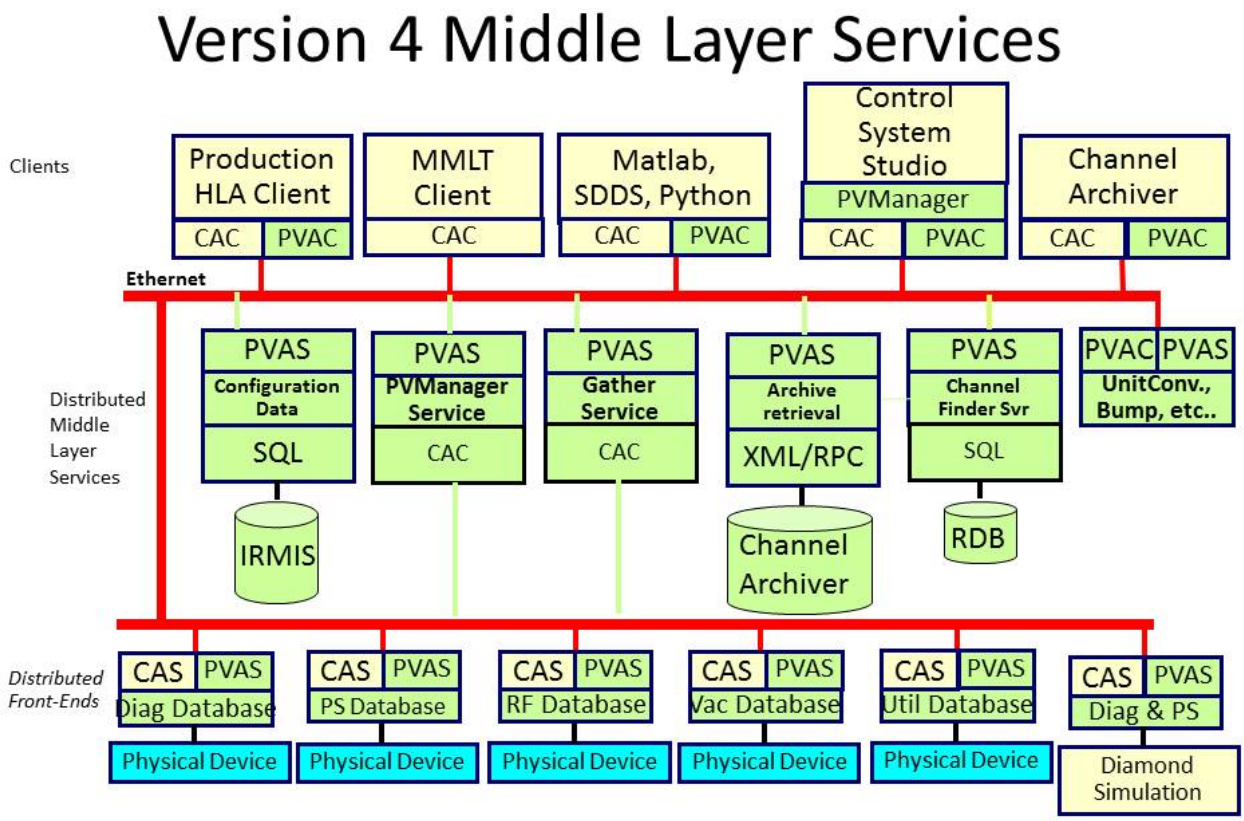

Figure 1: An EPICS Version 4 architecture drawing showing the new middle layers services that can be done with this extended set of data types. Many of these tools are being presented in this conference.

VERSION 4 NORMATIVE TYPES

Several normative data types have been defined to extend the set supported under EPICS version 3. All of the following normative types include a definition for time stamp called: NTTimeStamp , and alarm information called: NTAlarm. The general normative types are:

NTPVList, NTTimeArray, NTFreqArray, NTHistogramArray, NTMultiChannelArray, NTNDArray. NTStatisticalSamples, NTImage, and NTTable,

NTPVList

The PVList is a data structure that contains a list of process variables. In addition to the list of process variable names, this channel contains information needed to connect to this channel and provide the group as an order list. The fields may include connection information such as IP Address, port, priority, and an indication if it is currently available on the network. The field to create an ordered list could be the physical position along a beam line or storage ring.

NTTimeArray, NTFreqArray, NTHistogram

The arrays types need to be different as they represent very different data. The type informs the clients what types of information they present and the operations that can be done on them. NTTimeArray is from a digitizer and must have the time between samples to understand how it relates to other waveforms taken relative to it. NTFreqArray is an FFT of a NTFreqArray that requires the delta frequency between samples. The NTHistogram is an array of counts for different ranges of a scalar. This array can count values on either a linear distribution or log distribution..

NTMultiChannelArray

The multichannel array is a container for aggregating a set of scalar values into a single vector that is typically from the same time such as an orbit, or vacuum profile.

NTNDArray

An N dimensional array is used to transport a vector of data taken over time or a two dimensional array taken over time. This extends to an N dimensions.

NTStatisticalSample

A statistical sample is used to compress the data being sampled at a higher time frequency, into a lower frequency into a statistical sample that includes: high value, low value, mean, time of first sample, first sample, time of last sample, last sample, standard deviation,:number of samples and the or'd alarm severity of all of the samples. This data type is useful for reducing data from a longer period to a shorter period such as: raw bpm data samples at 10 KHz and reported at 1 Hz or record data being sampled at 10 Hz and being archived once every minute. It can also be used to compress a very long range of data coming back from the data archiver where the browser only has 500 pixels for displaying the data. Compression on the server side would greatly reduce the amount of network traffic and data transfer time.

NTImage

An image is a multi-dimensional array that needs additional information to know how to display it or do math operations on it. The NTImage must know the encoding and compression to be able to integrate, perform background subtraction, or display the image.

NTTable

A table is a fine way to collect a set of named parameters that are of a different type. This is a way to catch any arbitrary set of named elements that have a fixed number of values. If there is only 1 value (or row), it is a set of named value pairs. If there is more than 1 row, it is a set of named vectors. This can be used to capture a set of configuration parameters such as: experimenter name, company name, camera used, shutter speed, focus, etc… It can also be used to return the Twiss parameters for a set of magnets, where each column is one of the elements of the TWISS parameters and each row is a magnet.

CONCLUSION

PVData in Version 4 presents a set of normative types that can be aggregated, displayed, and archived. This include the version 3 normative types along with a server connected to version 3 databases. This set can be used to create general purpose servicers such as a directory service or a time synchronous vector from a distributed set of scalar values. PVAccess supports the serialization and de-serialization of PVData so that no changes are needed in this code to extend the set normative data types. More complicated services can collect the data needed from middle layer services as collections of these types and serve the results to high level applications or other middle layer services that will use them to create other results that may be served in turn to other clients or middle layer services.

This infrastructure along with this set of normative types are being used to develop the high level physics application environment for NSLS II in collaboration with the NSLS II control group, the NSLS II physics group, the SLS Control Application group and the Diamond Beam line Control Group. Early results show good performance. However, it remains to be demonstrated that this approach works and improves the overall development and maintenance of high level applications.

REFERENCES

[1] L. Dalesio, Kraimer, et. al., "EPICS Architecture", invited talk at the ICALEPCS, Tsukuba, Japan, 1991.
[2] J. Hill, et. al., "Channel Access Reference Manual", URL: http://www.aps.anl.gov/epics/base/R3-14/12-docs/CAref.html,, 2009.
[3] M. Kraimer, et. al., "pvAccess, pvData, pvIoc, pvService overview and status", EPICS Collaboration Meeting, Villigen, Switzerland, 2011.
[4] M. Borland, "SDDS Information"m URL: http://www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/SDDSInfo.shtml, 2001.
[5] G. Portmann, J. Corbett, A. Terebilo, "Middle Layer Software Manual for Accelerator Physics," LBNL Internal Report, LSAP-302, 2005. MATLAB MIDDLE LAYER TOOLKIT
[6] *XAL*, http://neutrons.ornl.gov/APGroup/appProg/xal/xal.htm