Brief History Of EPICSV4

EPICS Collaboration meeting Fall 2016 ONRL

Marty Kraimer

Table of Contents

1. History 2005 through 2009: javaIOC

1.1 Genesis

July 2005 Meeting

The focus of this meeting was to think about the future of EPICS. As I recall there were about 6 to 8 people who attended the meeting. I also recall that each person offered at least one vision and normally more than one.

At the end of the meeting I remember thinking:

> How can we possibly mesh all these visions?

Leave ANL Feb 2006

In December 2005 APS asked for people to volunteer to leave ANL. The offer included up to six months severance pay.

I had been thinking about my vision for the future of EPICS ever since the July meeting. I also thought that, if I stayed at APS, I would not have time to implement my vision. I took the offer and left ANL and started work on the vision.

When I left APS, I thought it would be about three years before I had users.

Obviously I made a really bad estimate!!

1.2 Vision

A brief statement of the vision was to implement an IOC similar to a V3 IOC but with support for fully structured data. This included the components:

pvData
: Structured Data

pvAccess
: Network support for pvData

pvDatabase
: Memory resident database of PVRecords

: PVAccess Support For PVRecords

The initial implementation language was chosen to be Java. The reasons were:

1. Java took care of memory management.
2. Java had recently added proper support for multi-threading.
3. I had been coming up to speed on programming in Java. (Maybe this was the most important reason ?)

pvData

pvData defines both introspection and data interfaces for the following types:

scalar
Since the implementation language was Java the scalar types were the Java primitives and string.

scalarArray
A 1D array of scalar.

structure
An ordered set of fields where each field has a name and one of the supported types. Note that since structure is an allowed field type this is a recursive definition.

structureArray
A 1D array of structure fields.

pvAccess

This is network support for transporting pvData.

Just like EPICS channel access, pvAccess provides network communication between a client and a server. The difference is that pvAccess uses pvData as the data definition.

pvAccess implements the following interfaces:

channelProvider
A channel provider allows a client to connect to channels provided by a server. Just like channelAccess, each channel must have a channelName that is unique on the local network. A channelProvider must implement interface channel.

channel
A channel provides the following interfaces:

getField

channelGet

channelPut

channelPutGet

channelArray

monitor

channelRPC

pvIOCJava

A pvIOC is a memory resident database of PVRecords. Each PVRecord has a name and a top level PVStructure. A record is "smart" because it has code associated with it.

pvIOC implements what today is called pvDatabase. In addition it had other, currently abandoned, components that are briefly described at the end of this talk.

1.3 Early Days

When I left ANL (Feb 1, 2006) , we (my wife Irma and I) moved to Osseo Michigan. I made one of the rooms in our house into an office.

The rest of the history through 2009 will be described via a brief description of what was reported at various EPICS Meetings.

1.4 Meeting APS June 2006

At an EPICS meeting at APS in June 2006 I presented the first progress report on the javaIOC. At this time what was implemented was the pvData component and the beginning of pvRecord and pvDatabase.

1.5 Meeting DESY April 2007

By this meeting a lot more was implemented. It had support for pvDatabase and a channelProvider for pvDatabase.

Financial Support

At this meeting Matthias Clausen (DESY) and Bob Dalesio offered to provide financial support for javaIOC development.

For the last half of 2007 through the first part of 2015, I was paid to work on the javaIOC and then on EPICS V4.

Matthias provided initial support (about 1 Man Year). Bob, via DOE SBIR grants and BNL, provided the remaining support.

Matej becomes responsible for pvAccess

A major missing component for the javaIOC was network support for pvAccess. During the EPICS Meeting Matthias, Bob, Matej Sekoranja, and I had a private meeting. During this meeting we discussed making Matej responsible for the network support for pvAccess.

Matthias had Matej and me come to DESY for a couple of days. During the day Matej and I met with Mathais and other people from DESY. After the daily meetings, Matej and I took a long walk and then had dinner together. We got to know and respect each other. Matej and I became close collaborators. From then until about the end of 2010 Matej and I were the main developers for pvData, pvAccess, and pvDatabase.

The support for Matej, who works for cosyLab, was initially from Matthais and Bob. In recent years he has also received support from DLS, PSI, SNS, and ESS.

1.6 Meeting INFN October 2008

By this meeting even more of what was descibed in the April 2007 meeting was working. This included the first version of network support for pvAccess.

2. 2009,2010 Beginning of C++ Implementation

2.1 Bob Dalesio Vision

Going back to the July 2005 meeting, remember that many visions were presented, not just mine.

Amoung these was Bob Dalesio's vision. Bob has presented his vision at several EPICS meetings so let me just say that an important part of his vision is Middle Layer Services between clients and data. An important data source is IOC records.

In order to provide really efficent Services it is desirable to be able to run them as part of the IOC that has the records. This means that pvData, pvAccess, and pvDatabase must be implemented in C++ as well as Java.

When Bob said that we must provide a C++ implementation my thoughts were:

- He is providing the funding so I better take this request seriously.
- This is going to take a lot of work.
- Great Vision!! Now we can develop code that lives in a V3 IOC. Truly powerfull!!

2.2 Implementing pvData, etc in C++.

Deciding to provide a C++ implementation required some pvData extensions.

2.3 New Types

Unsigned integer types

Scalar types for unsigned integer types of length 8, 16, 32, and 64 bits were added.

union and unionArray

Adding these two new types arose from work at Diamond on areaDetetor. The issue was to provide a pvData implementation of the NDArray used by areaDetector. Several approaches were tried but all were ugly. Union and unionArray provided a nice solution. The NTNDArray normative type (implemented by David Hickin) is the pvData implementation of NDArray.

2.4 Normative Types

Whenever I made the statement that an arbitrary stucture of data of the basic primative types could be created via pvData, Bob would say:

> We **MUST** somehow limit the types in order to implement services.

The solution is normative types. A normative type is a structure with a well defined set of fields.

2.5 Start of C++ development

At the beginning of the C++ development C++11 was not yet a standard and I was not familar with Boost C++. It was obvious that some sort of reference counting memory management was needed. My first effort was to implememt it myself. This led to several months of wasted effort.

Then Michael Davidsaver convinced Matej and me to use shared_ptr from Boost. Michael also gave us lots of help with implementation for both pvData and pvAccess. Michael also implemented the code to enforce "Copy On Write" for arrays in pvData.

3. History Since 2010

3.1 Additional Developers

In addition to Matej and me, the following people have managed and/or developed code that is in EPICS V4 release 4.6:

> David Hickin, Michael Davidsaver, Ralph Lange, Andrew Johnson, Greg White, Sinisa Veseli, Guobao Shen, Timo Korhonen, Kay Kasemir, Steve Hartman.

3.2 pvData

Only minor changes since 2010

3.3 pvAccess Enhancements

Two major enhancments since 2010 are:

pipelineService
: Lossless monitor service

codec support
: Support for other network protocols.

3.4 pvDatabase instead of pvIOC

pvDatabase implements database of "smart" records via a process method attached to each record. In addition it implements a channel provider for accessing the records.

3.5 IOC: V3 + V4

Since pvDatabase is now implemented in C++ as well as Java, a pvDatabase can be part of the same process as a V3 database.

A lot of effort since 2010 has involved interoperability between V4 and V3.

3.6 pvaSrv

Channel provider for connecting to V3 records.

3.7 pvaPy

Python support for pvData and pvAccess.

3.8 pvaClientJava and pvaClientCPP

A synchronous interface to the client side of pvAccess.

3.9 exampleJava and exampleCPP

A set of examples for using pvData, pvAccess, pvDatabase, and pvaClient.

3.10 developerGuide.html

Analogous to the "EPICS Application Developer's Guide".

4. javaIOC: Abandoned Components

4.1 pvIOCJava

Field support

Each field of a record instance can optionally have code attached to the instance.

Scalar expression

Code similar to the V3 calc record.
An expression variable can be a Java primitive or String.
Several operators are supported.

Link to another Record

Database and pvAccess links to other records is supported.

XML parser

XML can be used to create PVRecord instances.
Support is provided to attach code to a record.

Scan Threads

Similar to scan threads in V3 IOC.

4.2 portDriverJava

This is a Java implementation of asynDriver.

vxi11

Support was implemented for the vxi11 protocol (successor to GPIB protocol). This means that a large set of network based hardware devices are supported.

Industry Pack

Support was implemented for IP itself and for serveral IP modules.

Bacnet

Support was developed for interfacing to BACNET devices.

4.3 swtshellJava

This is a "probe" like tool for pvAccess/pvData.