EPICS VERSION 4 PROGRESS REPORT*

Timo Korhonen#, PSI, Villigen,
Leo Dalesio, Michael Davidsaver, Martin Kraimer, Nikolay Malitsky, Guobao Shen, BNL, Upton, Long Island, New York, USA
Matej Sekoranja, Cosylab, Ljubljana, Slovenia,
Andrew Johnson, ANL, Argonne, USA
David Hickin, James Rowland, Diamond, Oxfordshire, England
Ralph Lange, HZB, Berlin, Germany
Greg White SLAC, Menlo Park, California, USA

Abstract

EPICS Version 4 is the next major revision of the Experimental Physics and Industrial Control System, a widely used software framework for controls in large facilities, accelerators and telescopes. The primary goal of Version 4 is to improve support for scientific applications by augmenting the control-centered EPICS Version 3 with an architecture that allows building scientific services on top of it. Version 4 provides a new standardized wire protocol, support of structured types, and parametrized queries. The long-term plans also include a revision of the IOC core layer. The first set of services like directory, archive retrieval, and save set services aim to improve the current EPICS architecture and enable interoperability. The first services and applications are now being deployed in running facilities. We present the current status of EPICS V4, the interoperation of EPICS V3 and V4, and how to create services such as accelerator modelling, large database access, etc. These enable operators and physicists to write thin and powerful clients to support commissioning, beam studies and operations, and opens up the possibility of sharing applications between different facilities.

INTRODUCTION

EPICS version 3 supports a flat set of records and access to those records via a protocol that supports a few predefined structures [1]. While this serves well the needs of device integration, scientific applications require wide interfaces to access complex data and support for data acquisition. As there has been no standard way to do this, different mutually incompatible solutions have been developed. A closer look at these middle layer toolkits reveals a number of common patterns which have guided us in the development of the EPICS 4 facilities.

The work on JavaIOC [2] to create a new IOC (Input/Output Controller) that supports hierarchical records to describe devices and physics data constructs provided a starting point for the development. As the development proceeded, the goal to write a new IOC gave way to providing an extended set of structured data types on top of the existing infrastructure to better support various data sources for machine control and data acquisition applications. Thus in EPICS version 4 the version 3 database structures will be kept essentially untouched and the records will be served through a new network protocol. This makes the introduction of the new facilities even in existing system easy, as all the previous infrastructure will still be present.

Figure 1 is a simplified architecture drawing of an EPICS 4-based system. Such a system can contain client applications that only use Channel Access, only pvAccess or applications that can use both. The middle layer services would all need to use pvAccess for publishing their services but can serve data from various sources. The front-end IOCs contain both Channel Access and pvAccess servers so that all participants of the system have access to their resources.

The new features in version 4 extend the domain of EPICS beyond embedded IOCs by enabling several types of services like relational database stores, model services, directory services, data aggregation services, etc., to be built. These services can interoperate with traditional IOCs to combine static data with live control data. One of the main goals is also to bring EPICS into the domain of data acquisition, not only by the support for structured data but by optimizing the performance so that EPICS can be used as a part of data processing chains and handling big data volumes.

In the last two years the version 4 infrastructure has been developed to a state that allows it to be merged into the regular EPICS release series. At the moment it is planned that the current 3.15 will be last release under the version 3 series and the next major release will be carrying the version number 4. At the time of writing, EPICS 4 consists of a number of modules (pvCommon, pvData, pvAccess, pvaSrv, etc.) that are compiled and used on top of an EPICS version 3 base release.

NEW FEATURES

Version 4 new features target the implementation of a controls infrastructure based on services that aggregate data from various sources and publish it in a way that fits a control system environment. The services use the same concepts as EPICS in general: services are published by name and knowing the name, any EPICS client software can use that service as if it was a traditional IOC, without having to differentiate between services and IOCs in any way.

#timo.korhonen@psi.ch

ISBN 978-3-95450-139-7

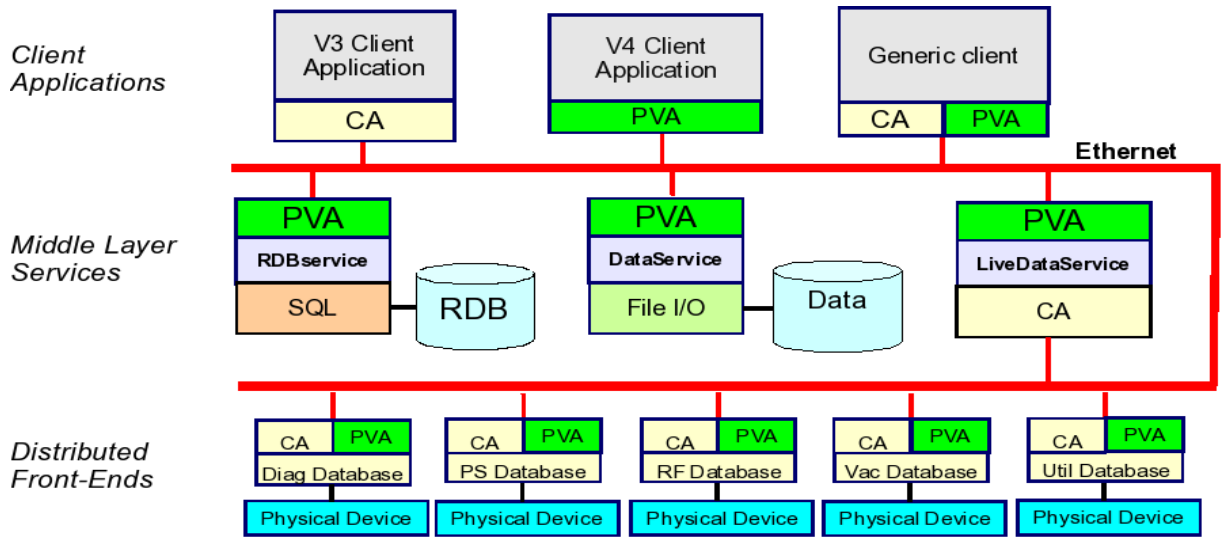

Figure 1: Architectural view of an EPICS 4 setup.

Structured Data

A module called pvData provides the support for data structures instead of single scalar values or simple arrays. With pvData one can create data containers where the basic elements of data are units and structures. pvData defines how the data are typed, interfaced and exchanged. The types may be complex, and care is taken in memory management and interfacing to make data I/O from memory as high performance as possible. pvData provides APIs for creating and updating pvData objects and for accessing meta-data descriptions of the data structures, the so called introspection interface.

pvData has four types of entities: scalar, scalarArray, structure, and structureArray. Scalars are entities of primitive data types, and scalarArrays are one-dimensional arrays of those. Structures can contain scalar types and further nested structures or arrays of structures. All typical primitive data types like signed and unsigned integers, single and double precision floating point numbers, strings, boolean arrays, etc. are supported.

```
structure beamOrbit
    alarm_t alarmStatus
        int severity 0
        int status 0
        string message
    time_t timeStamp
        long secondsPastEpoch
        int nanoSeconds
        int userTag
    structure [] positionData
        string  bpmName
        double zPos
        double X
        double Y
        double I
```

Figure 2: A pvData structure example.

Individual pieces of data can be combined to complex structures of related data. For instance, combining the x,y and intensity data from a beam position monitor into one coherent set, or combining BPM readings from several IOCs into a multidimensional array of positions (x,y,z) that define a beam orbit (Figure 2).

Wire Protocol - pvAccess

A new protocol, pvAccess [3], supports the efficient transfer of structured data over the network. It borrows a number of concepts from its predecessor (ChannelAccess [4]) like publishing a service name, name resolution with broadcasts, etc., but adds a number of new features. In addition to the traditional get, put and subscription (monitor) operations, pvAccess also supports queries with parameters ("channelRPC") and a put-get operation where after writing a value to a channel, a processed result is returned to the caller.

As the data structures behind the channel are not predefined, a client must find out the data structure via introspection and create storage for the data structure before it can ask data. At connection time introspection information can be passed from server to client and each side can create a data instance. The data is transferred between these instances. The data transmitted on the network does not have to be self describing since each side has the introspection information, thus saving data bandwidth.

To implement data services, the possibility to query with parameters is essential. The "channel RPC" operation is used for queries with parameters. For instance, a client application in a pulsed accelerator could request beam orbit data from a certain part of the machine for a range of pulse numbers. In a channel RPC operation, the query parameters are packed in a pvData structure and the service again returns a structure. The query and return structures are by convention of predefined types ("Normative Types") to make interoperation of applications possible.

A "channel put-get" set of messages are used to set (put) data to the channel and then immediately retrieve data from the channel. Channels are usually "processed" or "updated" by their host between put and get, so that the get reflects changes in the process variable's state.

ISBN 978-3-95450-139-7

PvAccess also provides queuing of monitored values which had only a limited support in EPICS 3. The support allows for configurable queue sizes and notifications to the client if the buffers have overflowed.

Usage of network bandwidth is optimized by the already mentioned separation of data and introspection, sending in chunks to avoid large latencies and preallocation of large buffers. In addition, pvAccess supports transmission of only those parts of a data structure that have changed.

Normative Types

EPICS 4 pvData is able to support unlimited types of data structures. However, it is not possible for an application to handle the data without knowing what the data represents, which would make it impossible to write generic applications like display managers. For that reason a number of normative types [5] have been defined. Each defines both a standard structure and its semantics, i.e., what is the type intended to present. When the structures have well-defined meanings, applications can be written to process the data without knowing anything about the source of that data. For instance a display manager can format data of known types in a meaningful way. Standard types also make it possible to build processing chains by combining generic processing elements; the elements can be software or processing implemented in a FPGA or a GPU.

INTEROPERABILITY

A major concern in an upgrade of an infrastructure with a big install base like EPICS is interoperability with the previous versions. In this case between pvAccess and Channel Access, plus the possibility to access IOC database records from pvAccess clients.

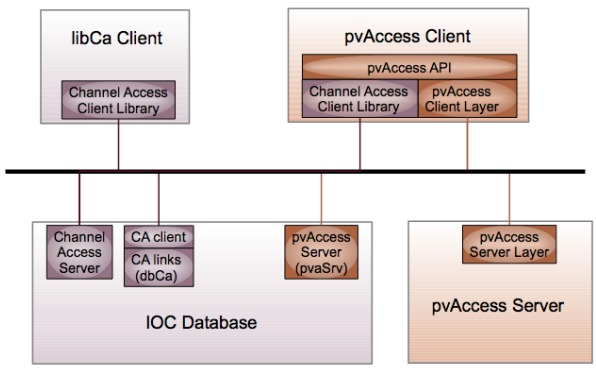

Figure 3: Interoperability diagram of EPICS 3 and 4.

Channel Access clients can use the "classic" infrastructure without any modification. They cannot of course access any pure pvAccess facilities like services that are not IOCs. On the other hand, pvAccess client API includes both Channel Access and pvAccess as "providers". That way, a single client library and API can be used to communicate using both protocols.

PERFORMANCE

One very preliminary performance graph for pvAccess floating point acquisition is shown below, together with comparable Channel Access acquisition. The setup is using C++ clients and servers doing GET operations, on a double array value varying the value array size. This result demonstrates the fact that pvAccess has been optimized for large data transfers. Note that in addition to raw network encoding and deserializing performance, in a real world implementation pvAccess could achieve much higher apparent performance since it additionally has the capability to transfer only actually changed structure field values [6].

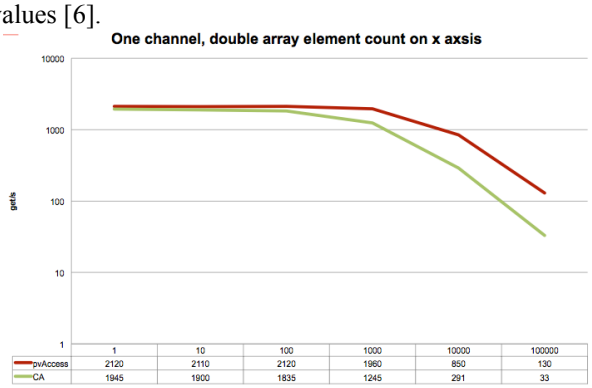

Figure 4: Performance comparison of ChannelAccess and pvAccess for a single channel acquisition.

SERVICES

The extensions in EPICS 4 enable support for several types of services that add value to the control system infrastructure by allowing easy integration of different data sources to the system.

IOC Level – pvaSrv

As the record structure from EPICS 3 will not be changed, a service providing access to the IOC records is required. This service, called pvaSrv is a part of the V4 package. At the time of writing pvaSrv can access records in the same way that Channel Access does, namely connecting to a single field of a record and returning a NT-type structure to the client. Work is underway to extend pvaSrv to allow collecting values of multiple EPICS Pvs into pvData structures. The collections can be configured via configuration files or dynamically on the fly via channelRPC calls. A proof-of-principle implementation of this is already included in the release version 4.3.0.

ChannelFinder

EPICS databases have a flat record namespace. While direct access to primitive data enables the building of lightweight controls applications without the overhead of complicated structures, it makes it difficult to provide a device-oriented view of the facility for high-level applications. ChannelFinder [7,8] is a service that provides a method to organize a flat list of channels into structures and hierarchies, by attaching metadata

ISBN 978-3-95450-139-7

("properties" and "tags") to the channels to describe their function. A client application can query the service using the properties and tags and get a list of corresponding process variables. For example, a tag could describe all channels belonging to a certain device, a further tag could describe all devices of a certain function like focussing quadrupoles.

Data Aggregation Service (Gather)

Another common requirement for building high level applications is a service that aggregates data from several sources into structures that are useful for client software. In a large facility measurement values or control points are usually located in distributed front-end IOCs. An application that needs access to these values needs to collect the data from several sources. Having a centralized service is efficient as data collection is done only once and the results served to many clients. In addition it also makes management of the system much easier as the client programs just need to ask the service for the data instead of retrieving the values separately from each involved IOC. A gather service can also combine static data to the data entities it delivers, for example combining the positions of monitors along the beamline with the live data from the IOCs. At the moment only a prototype implementation of a generic Gather service exists, but such a service is planned, as well as application-dependent Gather services for particular use cases like serving a beam orbit.

Archive Service

As many institutes have and are still using the Channel Archiver [9] for archiving control system data, creating a service that gives access to archived data with standard pvAccess client tools will give immediate value to those sites. Archive data can be requested in the same way as live data and possibly combined, to give an uninterrupted view of the evaluation of a PV in time.

Saveset Service

Machine Snapshot and Retrieve (MASAR) is a service for saving the values of a set of predefined channels, comparing their values to the live machine data and restoring them to the live system [8,10]. The service uses the channelRPC method of communication.

Relational Database Services

A relational database service that returns pvData structures has been implemented that listens on the pvAccess to incoming requests. Internally the service has a database table with names and corresponding predefined SQL queries. The parameters in the query contain the name for the query which is then executed by the server and the results formatted and returned to the client as NTTable structures.

SUMMARY AND OUTLOOK

The basic structures for EPICS 4 have been developed and are nearing completion. First deployments are underway and the results are very encouraging. EPICS 4 facilities have proven to provide an easy way of integrating control system data with services that support the needs and abstraction level of scientific applications. Services infrastructure is starting to take shape, and especially interesting prospects are developing in the experiment support area. The services enable a whole new range of applications, many of which will first be developed in the coming years. The integration of EPICS 4 structures in data acquisition applications will take EPICS into a whole new domain. EPICS version 4 provides a lot of new functionality while at the same time preserving the functions and the ecosystem with drivers and record structures of version 3. This enables a smooth transition to the new version, which is important for the large installation base of EPICS systems.

ACKNOWLEDGEMENT

Many people in the EPICS and wider controls community outside the core developer group have contributed to the success of this development by reviewing the documents, participating in the discussion and providing ideas and feedback. These and deserve our sincerest thanks.

REFERENCES

[1] L.R.Dalesio, G.Carcassi, M.A.Davidsaver, M.R.Kraimer, R.Lange, N.Malitsky, G.Shen, T.Korhonen, J.Rowland,M.Sekoranja, G.White, "EPICS V4 Expands Support to Physics application, Data Acquisition, and Data Analysis" ICALEPCS'11, Grenoble, October 2011, FRBHMULT06.

[2] M.Kraimer, "JAVAIOC ", ICALEPCS'07, Knoxville, October 2007, MOPB04.

[3] pvAccess Protocol Specification, http://epics-pvdata.sourceforge.net/pvAccess_Protocol_Specification.html

[4] Channel Access Reference Manual, http://www.aps.anl.gov/epics/base/R3-14/12-docs/CAref.html

[5] Normative Types Specification, http://epics-pvdata.sourceforge.net/alpha/normativeTypes/normativeTypes.html

[6] EPICS Version 4 web site, http://www.epics-pvdata.sourceforge.net

[7] "Device Definition and Composite Device Views on Top of the Flat EPICS Namespace", ICALEPCS'13, TUCOCB05.

[8] "Development Progress of NSLS-II Accelerator Physics High Level Applications",Lingyun Yang, Jinhyuk Choi, Yoshiteru Hidaka, Guobao Shen, Guimei Wang, IPAC2012, New Orleans, USA, THPPR018.

[9] "Overview of the Experimental Physics and Industrial Control System (EPICS) Channel Archiver", K.U.Kasemir, L.R. Dalesio, ICALEPCS'01, San Jose, October 2001, THAP019.

[10] "NSLS II Middlelayer Services", Guobao Shen, Yong Hu, Marty Kraimer, Shroff Kunal, Dejan Dezman, these proceedings, MOPPC155.

ISBN 978-3-95450-139-7