ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WECPR01

EPICS 7 CORE STATUS REPORT*

A. N. Johnson†, G. Shen, S. Veseli, Argonne National Laboratory, Lemont, Illinois, USA
K. Shroff, Brookhaven National Laboratory, Upton, New York, USA
T. Korhonen, European Spallation Source ERIC, Lund, Sweden
H. Junkes, Fritz Haber Institute, Berlin, Germany
M. G. Konrad, FRIB, East Lansing, Michigan, USA
R. Lange, ITER Organization, St. Paul lez Durance, France
S. M. Hartman, K. U. Kasemir, Oak Ridge National Laboratory, Oak Ridge, USA
M. A. Davidsaver, Osprey DCS LLC, Ocean City, USA
M. R. Kraimer, Osseo, USA
K. H. Kim, SLAC National Laboratory, Menlo Park, USA

Abstract

The integration of structured data and the PV Access network protocol into the EPICS toolkit has opened up many possibilities for added functionality and features, which more and more facilities are looking to leverage. At the same time the core developers also have to cope with technical debt incurred in the race to deliver working software. This paper will describe the current status of EPICS 7, and some of the work done in the last two years following the reorganization of the code-base. It will cover some of the development group's technical and process changes, and echo questions being asked about support for recent language standards that may affect support for older target platforms, and adoption of other internal standards for coding and documentation.

EPICS 7

The first version of the EPICS Control System Toolkit [1] that was officially named "EPICS 7" was released in December 2017, fulfilling a request from the EPICS community for a single downloadable software package that contained the core EPICS Base code and the extra "EPICS Version 4" C++ modules (now called "PVA modules") that support structured data and the pvAccess network protocol.

In the two years since that release, members of the development team have been adding features to the core, debugging and re-engineering the new modules, and improving the integration between the different parts of the code-base. Some code that will not be developed any further has been unbundled from the core and published separately as stand-alone modules.

Release History

The new 7.0 release series replaced the 18-month old Base-3.16 series, and prompted the closing of the 16-year old Base-3.14 series to reduce the maintenance burden on the limited number of core developers. The Base-3.15 series will continue to be maintained for sites using older hardware and operating systems that cannot support newer C++ compilers.

* Work supported by U.S. Department of Energy Office of Science, under Contract No. DE-AC02-06CH11357
† Andrew Johnson <anj@anl.gov>

Table 1 lists the versions of EPICS Base that have been released in the last two years.

Table 1: EPICS Releases from December 2017

Version	Release Date	Description
3.14.12.7	2017-12-15	Stable, bug fixes
3.14.12.8	2018-09-14	Final 3.14 release
3.15.6	2018-10-11	Stable, bug fixes
3.16.2	2018-12-12	Final 3.16 release
7.0.1.1	2017-12-15	First 7.0 release
7.0.2	2018-12-17	Modules rejoined, APIs
7.0.2.1	2019-03-20	Bug fixes
7.0.2.2	2019-04-23	Bug fixes
7.0.3	2019-07-31	Bug fixes, API changes

Source Code Repositories Each release series is maintained on its own branch of a Git [2] distributed version control system repository, but the new PVA modules each have their own separate Git repository which is integrated into the main one as a Git submodule. The master repository is on Launchpad.net [3] along with the main EPICS bug-tracker, but is also mirrored to a repository on GitHub [4] where the PVA submodules are kept.

Continuous Integration During initial development of the PVA modules they were built and tested in multiple configurations against the different release series of Base using a Jenkins [5] Continuous Integration server running on a commercial cloud-based service, and also by a Jenkins server hosted at Argonne which tested builds on different operating systems. The commercial service has now been replaced by Travis CI [6] and Appveyor [7], which are both free for use by open source software projects and integrate well with GitHub.

Release Frequency After EPICS 7.0.2 was published the developers agreed to try releasing new versions more often, generating bug-fix releases instead of publishing patch files when important fixes became available. This policy is getting some push-back from sites that prefer to apply patches to their existing installations than to import and build a completely new version.

SOFTWARE DEVELOPMENT

Just over 600 non-merge commits have been made to the 7.0 repository branch since December 2017, and over 1400 counting commits to the PVA modules as well.

Repository Restructuring

An important early change involved restructuring the main Git repository to simplify modifications that affected two or more of the core submodules. When the 7.0 branch was first created the individual core modules (the Common library 'libCom', Channel Access 'ca', and the process database 'ioc') were checked out in the source tree as Git submodules from separate branches of the main repository.

Unfortunately, this arrangement made it impossible to commit or review changes that modified code in more than one module at once, which is a fairly common requirement when working on the core modules. As a result, these branches were soon recombined into the main 7.0 branch, although for several releases they continued to be configured as if they were still separate stand-alone modules. This independent module configuration has now been removed which reduces some file duplication and may speed up the build slightly.

Unbundling the Portable CA Server

The Base 3.15 series included the code for the Portable Channel Access Server (PCAS) and its associated General Data Descriptor library (GDD) which provided the ability for other software to publish process variables (PVs) over the Channel Access (CA) network protocol. This code was unbundled in the 7.0.1 release as it was not easy to use or work with and the core developers want to encourage the use of the new PV Access protocol instead.

The GDD and PCAS software have been made available as a separate support module available from GitHub [8]. Version 4.13.2 was released in September 2018, the major version number referring to the CA protocol version it implements.

Common Library Developments

The epicsThread OS-interface API gained a new routine on all platforms called epicsThreadJoin(), which allows a parent thread to wait for a specific child thread to exit. This functionality has always been available in the epicsThread C++ class but wasn't provided for C code until now.

The General Time service that allows multiple time providers to be registered has learned a short-cut when only one-time provider is actually present. This configuration is common on workstation operating-systems, and the short-cut significantly reduces how long it takes to read out the current wall-clock time.

Various timer implementations have been changed to use the monotonic OS clock API that was recently added to all platforms. This should prevent these timers (and hence the CA library and the IOC's periodic scan threads) from freezing if time synchronization problems occur.

A third-party JavaScript Object Notation (JSON) [9] streaming parser has been included in Base since the 3.15 series, but the original code could only handle 32-bit integer values and EPICS now supports 64-bit numbers. The parser has been updated to the latest version which supports 64-bit integers, but its author appears to have abandoned it and has not responded to bug reports or code contributions though the GitHub project [10] since 2015. Several enhancements to the original library have been developed for future use by EPICS so the parser can now support most features of the newer JSON5 standard [11]. These changes have been offered upstream in a pull-request [12], but no response has been received from the author.

IOC Database Developments

The implementation of the IOC doesn't often see major changes that affect how it behaves, although several parts have been refactored in recent years to allow for enhancements and optimizations to the code. One recent behavior change altered how records handle field modifications made from outside the IOC while a record is busy.

If a put to a record field arrives from a Channel Access client and is processed while that record is busy (i.e. its PACT field is non-zero) the field value is changed as requested and the put operation also sets the record's PUTF field to a non-zero value. Later when that record processing gets completed, the IOC checks the PUTF field, sees that it's non-zero and arranges for the record to get processed again, ensuring that the effect of the field's update has been properly reflected in the state of the database.

The behavior described in the previous paragraph has been in place since 2002 or before, but the code failed to handle cases where the busy record was not the direct target of the put but was downstream of it, through one or more database links – in that case no reprocessing would occur. This short-coming has now been rectified by also propagating the PUTF state to other records processed by the target record through database links.

Another significant change which has not appeared in a released version yet strengthens the implementation of the IOC's Access Security module, which limits who can access and make changes to the Process variables (PVs) in an IOC that has this optional feature turned on. Previously an IOC would check the hostnames in its security rules against a hostname provided in a network packet sent by the client itself, which could easily be spoofed. The change lets an IOC be configured to check its client's IP addresses instead, doing a DNS lookup of the names in its security rules and comparing the resulting addresses with the source addresses of the packets sent by the clients.

New JSON link types have been added to allow reading and writing of dbState values and inject calculations into the read or write path to/from other link types, and also link types to enable debugging and tracing during link type development work. The development of these triggered some improvements and fixes to the extensible link type APIs that had been added for the Base 3.16 series but have not been used much yet.

The Macro Substitution and Include (MSI) program has been revised to clean up some memory leaks and converted from C to C++ code. An earlier fix for a bug reported in the program was later discovered to cause it to reject some valid and previously accepted input files. This was fixed and checks for the affected substitutions file syntax were added to the regression tests for it.

The IOC code has also gone through a number of smaller changes including cleaning up the implementation of some internal subsystems and adding more checks to some IOC shell command arguments, to prevent IOCs from crashing if commands are given bad parameters or are run at times when the IOC isn't ready to accept them.

PVA Module Developments

As implied above there were actually more commits made in the last two years to the PVA modules than to the original Core software modules. This follows a change of maintainer for the main pvData and pvAccess modules from Matej Sekoranja of CosyLab to Michael Davidsaver of Osprey DCS.

Shortcomings were discovered in the `shared_ptr` object ownership rules implemented by the C++ classes making up the main modules, which were causing memory leaks. A major task was undertaken to document and fix the rules, which also resulted in additional associated fixes and API changes in the downstream modules. Helpers were added to track and debug ownership issues and help identity reference loops. A number of unused internal utilities were removed, or in some cases deprecated in one release and removed in a later one. In other cases, a few classes were moved downstream to the only modules that actually made use of them.

Interfaces were added to allow pvData structures to be converted to and from JSON-encoded strings. A number of APIs were enhanced to simplify their use or make them thread-safe, and some new interfaces were introduced that were designed to significantly reduce the amount of boilerplate code needed to access structured data and to write both clients and servers for pvAccess. New classes were generally designed to be able to take advantage of features added in the C++11 language standard when the compiler supports that, although that standard remains optional and older compilers still work.

Some major rework was also needed to the internal functionality of both pvData and pvAccess to fix problems discovered with their implementations. Since the 7.0.1 release no incompatible changes have been made to the network protocol, although clients can no longer make connections to servers running the earliest versions of pvAccess. As happened with the older Channel Access protocol, care will be taken in the future to ensure that new versions of the pvAccess client and server library are able to communicate with older versions.

Command-line Clients The pvAccess module provides a handful of command-line client programs for performing simple requests against PVA servers. Most of these programs ('pvget', 'pvput', 'pvmonitor' and 'pvinfo') have been rewritten; one more has been added ('pvcall'); and another ('eget') was unbundled — it is still used at some sites, but will no longer be maintained as part of the EPICS core software.

The pvAccess APIs were designed to allow other network protocols or local channel providers to be accessed using the same API, and a Channel Access transport layer has been part of the pvAccess module for many years. During some of the developments described above it was found that the Channel Access interface layer could dead-lock in some circumstances. These issues have been fixed.

The PVA Access Security interface classes have been completely redesigned, and an implementation added that sends the information used by the IOC's Access Security API over the network channel, thus replicating the functionality that the Channel Access protocol provides.

Support for Access Security has been integrated into the IOC's new PVA server interface 'QSRV' which replaced the older pvaSrv code, and into the new p2p gateway application that allows traffic to cross between IP Subnets. The QSRV module makes IOC record fields accessible over PVA just as pvaSrv did, but it adds atomic access to groups of records, and provides a JSON link type that uses the pvAccess protocol to get or put data.

Normative Types This module provides helper classes for software to use to generate data structures that comply with the data type specifications recommended for use with PVA. Older versions of this module included some quite strict compatibility structure checks which have since been loosened; now any convertibly-equivalent structure is considered to be compatible with the specification. Unfortunately, a new module that sends data to older software that is still using the stricter checks might have difficulty communicating, although no site has reported having this issue yet. One solution for this would be to rebuild the older code against the newer version of EPICS 7.

The synchronous pvaClient module has seen a number of changes to improve reliability and ease of use. Its APIs now support access to any top-level field of a pvStructure, not just to a top-level field called 'value'. The module now provides multi-channel classes that allow clients to access multiple pvAccess PVs at once. In the future it will also support converting data to or from JSON-encoded strings.

The pvDatabase module was designed to make it easier to implement servers in C++ code. It has undergone similar changes to improve its APIs, and recently added support for plugins and record processing. The code is also more robust against requests from rogue clients.

PYTHON BINDINGS

The Python language has become increasingly popular in recent years for writing scientific and control-system applications, and this trend has also applied inside the EPICS community. Several different bindings for Channel Access were written by different people over the years, often with different purposes or user-groups in mind, although there was some competition involved as well. This tradition has continued with PVA protocol bindings.

Most EPICS Python modules can now be installed from the Python Package Index (PyPI) using the 'pip' program, and in some cases from channels available using the Conda package management system.

The pvaPy module that was included with the EPICS Version 4 release packages continues to be updated along with the EPICS core, although the code is now being maintained and released separately [13]. Several new versions of this module have been released in the last two years, the latest version adding support for the multi-channel APIs from the pvaClient module to do I/O against multiple PVs at once.

A newer module named p4p [14] was written by Michael Davidsaver when he took over as the pvAccess maintainer, to help him document and debug the protocol. This eventually became a fully-functional set of bindings for easily accessing the core PVA functionality from Python code.

Python programmers can use either module to write PVA client or server programs (or even both at the same time), or may call or respond to Remote Procedure Calls over PVA. Both modules support NumPy for fast access to large arrays.

DOCUMENTATION UPDATES

A workshop called an “EPICS Documentathon” was held at the European Spallation Source in Sweden in early September 2019 where the attendees worked on a number of enhancements to the EPICS documentation, which is currently spread over many different documents, Git repositories and websites.

The individual EPICS record types were originally documented in the EPICS IOC Record Reference Manual [15] which in 2005 was converted into a set of Wiki pages on the EPICS Wiki [16] hosted at Argonne, under the theory that it would be easier for community members to maintain them that way. Before the EPICS 3.15 release it became clear that an alternative approach was needed, and tooling was added to some of the core software to extract and generate HTML documentation from the DBD files that describe the individual record structures at build-time, as this would allow the field descriptions to be taken directly from the record type sources, guaranteeing they would always be up to date. The work of converting the Wiki pages into the DBD files in Perl’s “Plain Old Documentation” (POD) format was completed at the Documentathon, although the results have not yet all be merged into the Base repository.

Other IOC reference documentation has been maintained in the EPICS Application Developers Guide [17] which had been converted from FrameMaker into LaTeX format in 2010. The most useful parts of this document are now the descriptions of the APIs provided by the Common Library (libCom) module, and modern tools make it easier to maintain these kinds of descriptions by generating them from annotated source code. Another project worked on at the Documentathon was to extract API descriptions from the Guide and turn them into source code annotations for use by the Doxygen [18] documentation generator. This project is expected to take a number of years to complete.

The PVA modules have been annotated for Doxygen as they were being developed, but some additional documentation was published on the EPICS V4 website which is no longer being maintained and will eventually be archived. The useful files have been rescued, reformatted and integrated into a new EPICS Documentation website [19] that can publish documentation from any EPICS-related project that is properly configured to use the public “Read The Docs” [20] website and service. In addition to publishing the Doxygen output from the PVA modules a number of the more useful and up-to-date “how-to” documents from the EPICS Wiki are also being maintained as Restructured Text documents in a separate GitHub repository [21] and are automatically published to the new website.

FUTURE DIRECTIONS

The documentation update work described above will continue as effort is available to contribute to it, with more Doxygen annotations being added to the core header files, and the descriptions of the individual record types being brought up to date with the code. Plans need to be developed on what to do with other sections of the IOC Application Developers Guide.

Not described in this paper is a project currently under way for the Java implementation of pvData and pvAccess which involves rewriting and simplifying a significant proportion of the data container and networking code. The original APIs and implementation of these do not follow modern Java conventions, and they are seen as hard to understand and use. A project has been proposed to do a similar rewrite of the C++ pvAccess implementation, since its internal structure has equivalent problems that the piecemeal reworking done to date are insufficient to fix.

The IOC Database code currently supports a wider set of data types than the Channel Access protocol (64-bit integer types were added for the Base 3.16.1 release for example), but it doesn’t currently have a way to handle structured data types. With the addition of the PVA modules to the EPICS core there is a desire for the IOC database to be able to store and transport the larger and more complex data types they handle, although how this can be done remains to be devised.

The question has been asked by developers more than once about being allowed to use features available in the modern C++ language standards C++11 and C++17. The answer to this affects the target architectures that EPICS can be compiled for. The architecture with the oldest compiler is currently VxWorks 6.9 which has a gcc-4.3.3 cross-compiler that provides a few C++11 features. The latest VxWorks 7 release appears to offer C++17 compatibility, but no EPICS sites are known to be using any VxWorks 7 release yet. The community will have to take this step at some point but some core developers are reluctant to do so as it could cut off a significant number of existing users, or force them to expend effort upgrading the operating system for older systems to be able to run the most recent versions of EPICS.

CONCLUSION

Since its first release in December 2017, the EPICS 7 software and documentation has seen a significant amount of development work, bug fixes and improvements. There is still technical debt in the code-base, from both the older core libraries and the PVA modules, but by making it easier for those outside of the core group to contribute we hope to grow the number of developers and users who can help to continue the 30-year collaboration that is EPICS.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contributions of the EPICS Council, all EPICS developers, Documentathon participants and our users, who all contribute in one way or another to the continued development and popularity of the EPICS 7 toolkit.

REFERENCES

[1] New EPICS website, `https://epics-controls.org`

[2] Git version control system, `https://git-scm.com`

[3] EPICS Base project pages on Launchpad, `https://launchpad.net/epics-base`

[4] EPICS Base main repository on GitHub, `https://github.com/epics-base/epics-base`

[5] Jenkins Continuous Integration system , `https://jenkins.io/`

[6] Travis CI service, `https://travis-ci.org/epics-base`

[7] Appveyor CI service, `https://ci.appveyor.com/project/epics-base/epics-base`

[8] PCAS, the Portable Channel Access Server, `https://github.com/epics-modules/pcas`

[9] JavaScript Object Notation, `http://json.org/`

[10] Yet Another JSON Library project on GitHub, `https://github.com/lloyd/yajl`

[11] JSON5 Data Interchange Format, `https://json5.org/`

[12] GitHub Pull Request: Add Support for JSON5, `https://github.com/lloyd/yajl/pull/211`

[13] GitHub Project for the pvaPy module, `https://github.com/epics-base/pvaPy`

[14] PVAccess for Python (P4P) project on GitHub, `https://github.com/mdavidsaver/p4p`

[15] Janet B. Anderson and Martin R. Kraimer, "EPICS IOC Record Reference Manual," EPICS Release 3.12, Advanced Photon Source, Argonne National Laboratory, April 1996

[16] EPICS Wiki, `https://wiki-ext.aps.anl.gov/epics/index.php/Main_Page`

[17] Martin R. Kraimer *et al.*, "EPICS Application Developers Guide," EPICS Base Release 3.16.2, October 2018

[18] Doxygen documentation generator website, `http://www.doxygen.nl/`

[19] EPICS Documentation website, `https://docs.epics-controls.org/en/latest/index.html`

[20] Read The Docs service, `https://readthedocs.org/`

[21] How-to pages on the EPICS website GitHub project, `https://github.com/epics-docs/how-tos`