EVOLUTIONARY PLANS FOR EPICS VERSION 3*

Andrew Johnson, Argonne National Laboratory, Argonne, IL 60439, USA
Ralph Lange, Helmholtz-Zentrum Berlin / BESSY II, 12489 Berlin, Germany

Abstract

With the Experimental Physics and Industrial Control System (EPICS) toolkit being used around the world, modifications to the core software must be very carefully designed to avoid breaking existing applications; this often limits the changes and new functionality that can be introduced. A new way to extend the EPICS input/output controller (IOC) was discovered recently that should be fully compatible with most channel access (CA) client programs; if the IOC supports optional postfix modifiers to the process variable (PV) field names it publishes, it becomes possible to add several features to the EPICS toolkit. However, if those field modifiers can be written in standard JavaScript Object Notation (JSON) syntax, they can encode a complex data structure and become very powerful, permitting client-specific configuration of individual CA channels without necessitating any changes to the network protocol. This paper will describe how EPICS Base is adding support for JSON encoding and field modifiers, and will discuss other features such as record aliases that have been introduced to help control systems evolve.

INTRODUCTION

EPICS [1] is mentioned as a keyword in 23% of the abstracts announced for this conference†, demonstrating that the toolkit has achieved significant penetration in control systems for experimental physics installations worldwide. The core EPICS developers understand the importance of this software to the user community, and they work hard to avoid jeopardizing future uses by only releasing software that can interoperate with existing installations.

Almost all existing EPICS installations use various releases of EPICS Base 3.13 or 3.14. This paper discusses new features for future releases of this code-base that will retain the ability to communicate with current versions and provide a route for the migration of existing IOCs and client programs.

One key to the evolution process will be the ability to continue to use existing Channel Access client applications without having to make any significant modifications to their source code. This prevents us from changing the Application Programming Interface (API) to the Channel Access client library, although enhancements can be added providing they are optional. The main way in which we propose to provide access to some of the enhanced functionality described in this paper is by modifying and extending the syntax of the Process Variable names that are transported by Channel Access.

* Work supported by U.S Department of Energy Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.

† A keyword search on the ICALEPCS JACoW website for 'EPICS' retrieved 98 abstracts out of a total of 419 on 2009/09/15.

PROCESS VARIABLE NAMES

An IOC Process Variable (PV) name is made up of two parts, a *record name* and a *field name*, combined using the following syntax:

record name
record name **.**
record name **.** *field name*

When parsing a PV name within the IOC, the first space character terminates the name, thus neither the *record* nor *field name* may contain spaces. Similarly a period marks the end of the *record name* part, so record names cannot usefully contain periods.

There are documented restrictions [2] on the characters that may appear in a *record name*, but these are not enforced by the IOC software, and it would be unwise to rigidly enforce them at this point. As a result, we assume only that a *record name* never contains control characters, spaces, or periods.

The *field name* part is more limited. A record type's field names are defined in its database definition (DBD) file and must all be valid C identifiers — this restriction is enforced by the C compiler when building the record support code. We can thus be confident that no existing IOC applications are using other characters in their field names, which makes it possible to use this part of the PV name as an extension point for future enhancements.

If no *field name* part is included in a PV name, the record's VAL field will be used by all existing versions of EPICS Base, although future releases may make this behavior configurable.

Field Modifiers

With EPICS Base version 3.14.11 some fields can now be specified using this PV name syntax:

record name **.$**
record name **.** *field name* **$**

The dollar symbols are called "field modifiers" because they modify the metadata that describe the named field (VAL if no *field name* is given) or other aspects of the data accessed through this channel.

The postfix dollar sign field modifier is allowed on fields of type DBF_STRING or any of the DBF_LINK types, and it changes the native type reported for the field into an array of DBF_CHAR elements. This modifier in conjunction with changes to the field conversion code inside the IOC makes it possible for existing CA clients to transport strings longer than 40 characters without having to make any change to the CA protocol. Both the MEDM and EDM graphical display tools have supported using character arrays for text transport for many years.

Future Field Modifiers

A field modifier that has already been implemented experimentally is an array offset, a long-requested addition to the CA protocol. As a field modifier, however, no protocol change is needed to provide this functionality, although a protocol extension would give more flexibility (this solution only permits the offset to be changed by disconnecting and reconnecting to the record again). The field modifier consists of a postfix integer inside square brackets, giving the following syntax:

record name **.[** *array offset* **]**
record name **.** *field name* **[** *array offset* **]**

The result will permit CA client programs to access any subset of an array field without transporting the whole array. Currently this feature has to be implemented in an IOC application by including subArray records to extract the needed subsets from the array data.

Field modifiers do not have to be limited to simple postfix flags or parameter definitions. Some EPICS users have expressed the desire to be able to access multiple fields of a record simultaneously, which could be achieved with this syntax:

record name **.** *field name* **,** *field name*

Several fields from the same record could be named, and the channel metadata would describe the channel as an array. The obvious data type of the array would be that of the widest numeric type of the fields included, or for simplicity, the implementer could declare that all elements must be converted to the epicsFloat64 type. However, this does prevent the use of string fields in the list, and a more flexible solution is obviously desirable.

JavaScript Object Notation (JSON)

The abstract to the IETF memorandum RFC-4627 [3] on JSON states:

> JavaScript Object Notation (JSON) is a light-weight, text-based, language-independent data interchange format. It was derived from the ECMAScript Programming Language Standard. JSON defines a small set of for-matting rules for the portable representation of structured data.

This format was designed to encode structured data, but unlike XML, the resulting strings are compact and both easy to read and create by hand. JSON could be a suitable representation for the list of mixed field data called for by the field list modifier described above, although using it would require any CA client programs to understand the JSON result. The intention is not to introduce the format merely to solve this problem though. In addition to the uses described in this paper, JSON will be useful in many other ways in the future.

The main development branch of the EPICS Base CVS repository now contains a copy of a JSON encoding and decoding library YAJL [4], which was chosen for its Open Source license terms, event-driven parser API, reluctance to allocate memory, and ease of porting to the EPICS build environment.

JSON Field Modifiers

JSON can be used to encode complex field modifiers in an extensible fashion. The PV name restriction that no space characters are included can easily be complied with, and the only difficulty foreseen is that in JSON names and value strings must be enclosed in double quotes, which would have to be escaped inside the current database file field value syntax. The latter could be modified to accept single quoted strings to alleviate this problem.

A PV name with a JSON field modifier would use this syntax:

record name **.{** *JSON object* **}**
record name **.** *field name* **{** *JSON object* **}**

The *JSON object* consists of a list of name/value pairs that specify one or more server-side configurable plug-ins to be applied to this channel. Within each pair the name selects a particular plug-in by name, and the value would be passed to the plug-in as its configuration parameter. Some plug-ins might only need a single string or numeric parameter, but others might expect a nested JSON object containing multiple name/value pairs.

A few examples may clarify the above description. The first plug-in suggested here is drawn from the ideas being developed by Ralph Lange and Bob Dalesio [5]:

```
Current.VAL{"rate":0.5}
```

The above PV name requests the rate filter plug-in and provides a single parameter, the maximum rate at which updates should be sent through this channel. The same plug-in might also accept multiple parameters using the JSON object syntax as follows:

```
Current.{"rate":{"min":0.1,"max":0.5}}
```

The examples above are for a filter to be included in a monitor event data stream. Other plug-ins for CA get or put operations could provide the ability to perform atomic write-read or read-write functions:

```
Slew.{"putget":{"RA":18.8931,"DEC":33.0175}}
```

This example shows an atomic put of target angles for both axes of a telescope slew command. The plug-in would check and save its parameter data; then when a CA get operation is requested on the channel, it sets the RA and DEC fields of the record from the saved parameters, processes the record, and returns the record's VAL field as the result of the CA get.

RECORD ALIASES

As experimental facilities are modified and extended over time, their control systems need to be changed to match the underlying hardware. It can be very difficult to make some types of changes due to the widespread nature of the modifications needed. One such place where this is common is in the record names, which can quickly spread to many higher-level applications.

In order to alleviate some of these problems, IOCs built with EPICS Base version 3.14.11 support the ability to define aliases for their record instances. This permits an engineer to add a new name for an existing record while it continues to answer to its old name. As a result, it is not

immediately necessary to find and fix all the client applications that still use the old name; the facility will continue to function while the name change is slowly propagated through to all the higher-level applications.

It is possible to discover remotely whether a particular record name is an alias by fetching its NAME field as a string (or a long string using the **$** field modifier), which will always return the canonical name for the record.

STATUS

This project is still in the design phase, although the field modifier technique is already used in the recently released version 3.14.11 of EPICS Base. Integration of the JSON parser into the IOC's PV name lookup code will require some internal changes that could take several months, but no major difficulties are expected from the programming perspective.

CONCLUSION

Despite its age, EPICS can still be modified to add new functionality without breaking compatibility with existing control systems. Record aliases and field modifiers are the latest weapons in this arsenal, and by introducing an Internet standard format (JSON) and an externally written library (YAJL) to parse it, we are reducing the effort needed to implement this.

The possibilities that this technology opens up are significant and will permit the implementation of several features that have been desired for many years.

REFERENCES

[1] The Experimental Physics and Industrial Control System website, http://www.aps.anl.gov/epics/

[2] Martin R. Kraimer et al., "EPICS Application Developer's Guide, EPICS Base Release 3.14.11," August 2009, http://www.aps.anl.gov/epics/base/R3-14/11-docs/AppDevGuide.pdf

[3] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)," RFC 4627, July 2006, http://www.ietf.org/rfc/rfc4627.txt

[4] Lloyd Hilaiel, "Yet Another JSON Library," v1.0.5, http://lloyd.github.com/yajl/

[5] Ralph Lange, Andrew Johnson, Leo Dalesio, "Advanced Monitor/Subscription Mechanisms for EPICS," THP090, this conference.