pvaPy vs p4p

Patrik Marschalik

February 13, 2018

This is a short comparison of the two competing python packages pvaPy and p4p, that provide an interface to the pvAccess library of EPICS v4.

1 Overview

For EPICS v3, which only supports Channel Access, there is only the pyepics package that provides an interface to Channel Access. This library seems to be relatively mature. Since EPICS v4, there also exists the pvAccess interface that is different from Channel Access and is therefore not accessible from pyepics. Currently, there are two competing packages that aim to close the gap. One is the pvaPy package which is developed by Sinisa Veseli, the other one is p4p developed by Michael Davidsaver. It seems that both packages only have one developer.

2 Requirements

Both packages require at least EPICS Base 3.14.12 and the EPICS v4 modules pvDataCPP and pvAccessCPP. For this review, EPICS v7.0.1.1 is used that already contains all necessary modules. Both packages require Python 2.7 or 3.4 and later. p4p explicitly requires Numpy 1.6 or higher. For pvaPy the situation is more complex. It requires Boost ($\geq$ v1.41.0) built with Python support. The additional library Boost.numpy is optional. Starting with Boost v1.63.0, Numpy support is already included in the Boost library, at least as option at compile time. Here, the Boost library in version 1.66 was used. It was compiled twice, once with support for Python 2.7 and once with support for Python 3.5.

3 Building and installation process

Once EPICS with all needed modules is compiled, including Boost for pvaPy it was possible to build both packages both with Python 2.7 and Python 3.5 support. In both cases, the environment variables had to be carefully adjusted in order to find all necessary dependencies. The same was needed for installation. In particular, PYTHONPATH has to be set correctly, depending on which Python version is used.

4 Documentation

Both packages seem to have a full API documentation. For p4p this is available online, for pvaPy it can be built using Sphinx. In addition, p4p has a getting started page. An example folder is included in the sources of both packages. There are three and 27 examples for p4p and pvaPy, respectively. A detailed tutorial is missing in both cases.

5 Comments

What I like for the p4p packages is the implementation of a context manager that allows the use of the with-statement. What I dislike a lot is that it seems that p4p has its own client rather than wrapping the existing c++ client.

6 Conclusion

With both packages it is easy to read and write PVs and subscribe callback functions.

Regarding the installation process both packages leave quite a lot do be desired. A Python package should provide a Setup.py file and the possibility to be installed with all dependencies via the pip command, whether it is available on PyPi or not.

Also, an introduction or tutorial that is a bit more detailed than the given examples and getting started page would be nice for the future.

Not only pvAccess but also Channel Access is possible with both packages. A thorough comparison of either one of the two packages with the established pyepics packages, for Channel Access, has still do be done.

p4p is more modest concerning the dependencies. Boost itself has a lot of dependencies. On the other hand, it is an actively maintained and well established library and not just a fancy dependency. I guess that the dependency on Boost could make pvaPy more performant, I haven't tested this extensively, but that was my first impression. Although it is preferable to have as few dependencies as possible, I would advise to not consider this as a disqualifying factor, above all if it comes with better performance.

Both developers were available for questions and answered all questions quickly. So we can thank both Sinisa and Michael for their work in developing a Python interface for EPICS.