Python Channel Access Bindings

Review of a conversation
on EPICS tech-talk

4 different interfaces

CaPython (FNAL)

http://www-d0online.fnal.gov/www/groups/ctl/epics/epics_python.html

A faithful reproduction of the low level channel access API.

PythonCA (KEK) / NPEI (PSI)

http://www-acc.kek.jp/EPICS_Gr/products.html

http://controls.web.psi.ch/cgi-bin/twiki/view/Main/NewPythonEpicsInterface

Also based on a low level API.

EpicsCA (UChicago)

http://cars9.uchicago.edu/~newville/Epics/Python/

Apparently inspired by EZCA (“easy CA”) with higher level API

Cothread (Diamond)

http://controls.diamond.ac.uk/downloads/python/cothread/

Also EZCA inspired, with integrated coroutine based threading model.

Important Differences

- SWIG vs ctypes (vs raw C)
  - ctypes is only relatively recently mature enough to be useful. Advantages are no need to compile, API can be version independent.
- Documentation
- Abstraction provided by API

Notes on `ctypes`

- Easy interfacing to dynamically loaded `.so` files
- Need to translate C declarations into `ctypes` definitions, no automatic tools for this.
- If `.so` API remains stable then same Python source will work across multiple versions of library and Python.
- No compilation required, installation is easy!
- Interfacing to C++ is a total nightmare, probably easier to write a C++ Python wrapper.

Tech Talk

- High level vs low level API
- Is it worth creating a (another?) standard Python CA wrapper library?
- Datatype support is important: character arrays as strings, for example!
- Things need to work on 64-bit Linux!
- Channel access length handling needs work:
  - It would be good if waveform.NORD generated updates when it changes.

Cothread API – an “abstract” API

```
caput(pvs, values,
    repeat_value=False,
    timeout=5,
    wait=False,
    throw=True)

caget(pvs,
    timeout=5,
    datatype=None, format=FORMAT_RAW,
    count=0,
    throw=True)

camonitor(pvs, callback,
    events=DBE_VALUE,
    datatype=None, format=FORMAT_RAW,
    count=0,
    all_updates=False,
    notify_disconnect=False)
```

Issues to Discuss

- Do we need a “standard” Python API?
  - Would allow shared tools to be built on top of it.
- What kind of API do we want? Abstract or low level? Abstract hides details which may be wanted in certain applications.
- Toolkit / framework / GUI integration?
- Threading: asynchronous threads or cothreads? Affects framework integration.

What now?