AN UPDATE ON CAFE, A C++ CHANNEL ACCESS CLIENT LIBRARY, AND ITS SCRIPTING LANGUAGE EXTENSIONS

J. Chrín, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

ABSTRACT

CAFE (Channel Access interFacE) is a C++ client library that offers a comprehensive and easy-to-use interface to EPICS (Experimental Physics and Industrial Control System). Functionality is provided for the synchronous and asynchronous interaction of individual and groups of low-level control data, coupled with an abstraction layer to facilitate development of high-level applications. The code base has undergone major refactoring to make the internal structure more comprehensible and easier to interpret, and further interfaces have been implemented to increase its flexibility, in readiness to serve as the CA host in fourth-generation and scripting languages for use at the SwissFEL, Switzerland’s X-ray Free-Electron Laser facility. An overview of the structure of the code is presented, together with an account of newly created bindings for the Cython programming language, which offers a major performance improvement to Python developers, and an update on the CAFE MATLAB Executable (MEX) file.

THE CAFE MODEL

The C++ Channel Access Client library serves as host to scripting and domain-specific languages, and event processing agents that aggregate and analyze data.

C++ CAFE

THE ADVANTAGES

- The inherent simplicity and convenience of maintaining a single CA interface code.
- New CA functionalities from future EPICS 3 releases need only be integrated into a single base library.
- A uniform response to errors and exceptions that facilitates trace-backs.
- The CA class is well separated from the internals of the domain language meaning that bindings to other scripting and domain-specific libraries are vastly simplified.

The C++ Channel Access Client library serves as host to scripting and domain-specific languages, and event processing agents that aggregate and analyze data.

CyCafe: Syphoning CAFE with Cython

Code Listing: PyCafe Read/Write/Monitor Examples

```python
import PyCafe
cafe = PyCafe.CyCafe()
cyca = PyCafe.CyCa()

#handlePv=handle/'pvName'>
dict=<'int','float','str','native'(default)>

#handlePV=handle/'pvName'>
status,statusList = cafe.getAsyn(hList)

#Get cached ctrl data
pvCtrl = cafe.getCtrl(handlePV)
cafe.set(handlePV,[values])

#struct
PvData= cafe.getPV(handlePV,dt='struct')

#return list of structured data
valList = cafe.getList (handlePV)

#return numpy.ndarray
memview = cafe.getArray(handlePV,dt='float',asnumpy=True)
npArray = cafe.getArray(handlePV,asnumpy=True)

#list,
#set waveform; input [values] may be any of
#s gives overall status, sList is a status list
pvList=
handlePV=<handle/'pvName'>

#close cafe
cafe.terminate()
```

C++ IMPLEMENTATION

- Management of client-side CA connections.
- Memory optimization, particularly when connections are restored.
- Separation of data retrieval from its presentation.
- Strategies for converting between requested and native data types.
- Caching of pertinent data related to the channel and its state.
- Aggregation of requests for enhanced performance.
- Adaptive correction procedures, e.g. for network timeouts.

The outcome of each method invocation is captured with integrity in every eventuality, ensuring reliability and stability.