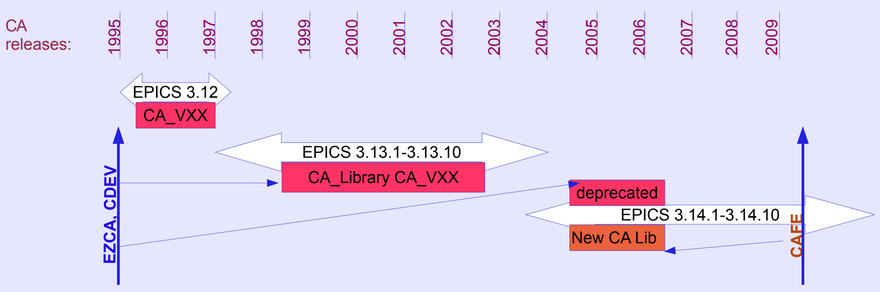
A TASTE OF CAFE

J. Chrin, G. Prekas, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract

CAFE (Channel Access interFacE) is a new C++ library that provides a multifaceted interface to the latest CA functions released with EPICS version 3.14. Functionality for both synchronous and asynchronous interactions has been implemented for individual, groups and collections of related channels. An abstract layer that addresses requirements dictated by beam dynamics applications has also been added. An XML-based configuration mechanism provides a convenient framework for users to define and initialize CAFE objects, e.g. for data analysis and/or visualization. Rules to flag members of a group/collection of CAFE objects, effectively modify a transaction to a selected subset, thereby allowing users to readily adapt to changes in a system during operation. CAFE is intended for use in C++ frameworks, such as Qt or ROOT, and presents itself as a candidate for event processing agents that, for example, capture machine physics data for inter-shot analysis at the SwissFEL. In this respect, the role of CAFE in aggregating low-level hardware events to produce events that supply summarized data to a Data Distribution Service (DSS), is demonstrated. Python bindings to CAFE are also in preparation for rapid application development with basic read/write functionality already implemented.

Why CAFE?



Modest changes to channel access (CA) over the past many years safeguard compatibility between old and new client/server connections. However, many C/C++ extensions (or wrappers) to CA are either frozen or not rigorously maintained and often do not reflect recent advances in channel access, such as multithreading and handling of lost connections.

Enter CAFE

- hooks into latest CA client library
- keeps in step with latest CA releases
- synchronous, asynchronous interactions for individual and groups of channels
- intricate interfaces tailored towards beam dynamics applications
- collections view related devices as a logical software entity
- rules to flag collection/group members to reduce collection/ group to a selected subset
- fast DAQ for inter-shot analysis (<10ms) at the SwissFEL
- bindings to scripting and 4th generation languages possible
- PyCafe (CAFE interface to Python) in preparation

XML Configuration

DBPM Collection definition

Collection name

Collections of related devices (e.g. all DBPMs) are defined in XML and generated automatically from a master XML file that offers an hierarchical view of the SwissFEL in Standard Machine Format (á la XAL).

```
<cafe:collections xmlns:cafe= "http://fel.web.psi.ch">
    <cafe:collection id="cDBPM" >
        <cafe:description>250 MeV Injector DBPM
Collection
        </cafe:description>
        <cafe:attributes>
            <cafe:attribute> X          </cafe:attribute>
            <cafe:attribute> Y          </cafe:attribute>
            <cafe:attribute> I          </cafe:attribute>
            <cafe:attribute> ENABLE     </cafe:attribute>
              ...
        </cafe:attributes>
        <cafe:member>
            <cafe:device> FINSS-DBPM10
</cafe:device>
        </cafe:member>
        <cafe:member>
            <cafe:device> FIND100-DBPM10
</cafe:device>
        </cafe:member>
        <cafe:member>
            <cafe:device> FINSB01-DBPM10
</cafe:device>
        </cafe:member>
        ...
    </cafe:collection>
</cafe:collections>

<cafe:config xmlns:cafe= "http://fel.web.psi.ch">
    <cafe:group id="gDBPM" >
        <cafe:description>250 MeV Injector DBPM Agent
        </cafe:description>
        <cafe:collection>
            <cafe:id>          cDBPM      </cafe:id>
            <cafe:attribute> X            </cafe:attribute>
            <cafe:datatype>CA_DOUBLE</cafe:datatype>
        </cafe:collection>
        <cafe:collection>
            <cafe:id>          cDBPM      </cafe:id>
            <cafe:attribute> Y            </cafe:attribute>
            <cafe:datatype>CA_DOUBLE</cafe:datatype>
        </cafe:collection>
        ...
        <cafe:collection>
            <cafe:id>          cDBPM      </cafe:id>
            <cafe:attribute> ENABLE       </cafe:attribute>
            <cafe:datatype>CA_SHORT  </cafe:datatype>
        </cafe:collection>
    </cafe:group>
</cafe:config>
```

DBPM group name

Agent Configuration File

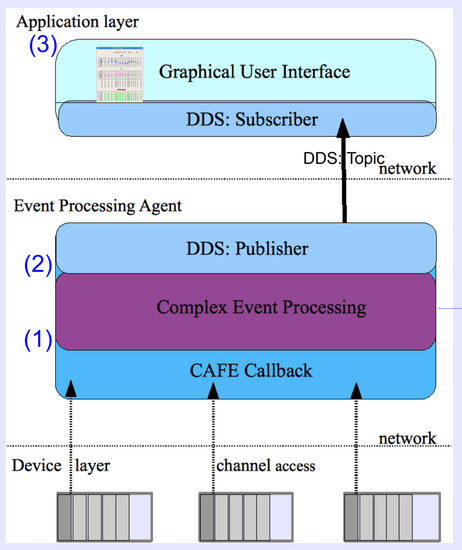
Initialization of CAFE objects in EPAs (and other applications) is simplified through the use of collections in the XML configuration file.

A CAFE method reads the configuration file, expands a collection into its members, and initializes the corresponding CAFE group for optimal(*) interaction with the low-level EPICS hardware

(*) single message

An Event Driven Application with XML, CAFE, DDS and Qt

Object Management Group's Data Distribution Service (OMG's DDS) implementation from OpenSplice



- Publish/Subscribe through “topics”
- Basic data structure expressed in IDL
- 'Readers' and 'Writers' of topics
- Language bindings for C/C++/Java
- Unbounded strings, sequences
- Fault tolerant data persistence
- High performance: shared memory use

```
class  EPA_DBPM
in input_execution_type
out  output_execution_type {

   Event Pattern Rules
   (Trigger, body of action)

   Local Variables
   ....

} EPA_Class Map (Aggregates and
                 correlates posets)
```

(1) The Event Processing Agent (EPA) is configured from XML and uses the CAFE API to establish a callback mechanism to EPICS

(2) The EPA monitors its input to detect instances of the rule trigger; when a match is detected (i.e. data transfer of DBPMs is complete) the agent executes the action of the rule's body causing the EPA to change its local state variables and output its event to the DDS

(3) The Qt based client application displays summarized DBPM data received through DDS

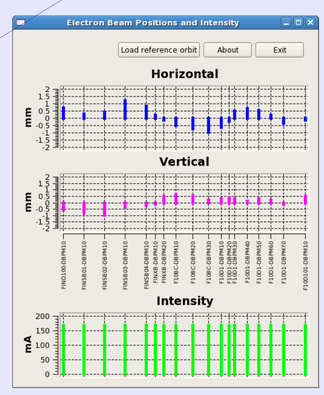
Establishing channel access Callbacks in EPAs is made simpler with CAFE functions

Parses XML configuration file

Initialize CAFE group: sequence of CAFE Objects

```
#include <cafe.h>

int main(int argc, char ** argv) {

   CAFE cafe;

   cafe.init(); // Initialize CA

   cafe.loadGroups(“cafeGroups.xml”);

   PVGROUP pvgroup = cafe.getPVGroup(“gDBPM”)

   pCallback callbacks[pvgroup.npv];

   evid * evids = new evid[pvgroup.npv];

   cafe.startGroupMonitor(&pvgroup,
                          &callbacks[0], evids);

}
```

```
struct PVGroup {
   char name[40];
   unsigned short npv;
   long statusGroup;
   PVDatum * pvdata;
}

struct PVDatum {
   char pv[60];
   char attrib[20];
   CA_DATATYPE dbrType;
   unsigned long nelem;
   long status;
   boolean rule;
   DBR_DATATYPE_UNION * val;
}

union DBR_DATATYPE_UNION{
   char str[40];
   short s;
   float f;
   unsigned short us;
   unsigned char ch;
   long l;
   double d;
}
```



Qt version: Qt-4.4.3
Qwt version: Qwt-5.1.1

Programming with C++ and Qt: Pros and Cons

- write once, compile anywhere
- memory management handled explicitly but better memory and runtime efficiency
- Fewer libraries but more tools (faster compilers, debugging, design)
- Qt toolkit for GUIs is fast and concise; also available for Python
- More programming experience required for application design

References

J. Chrin, “A Taste of CAFE”, SLS Internal Document, 1999.

N.T. Karonis, “EZCA Primer”, Internal Document, Argonne National Laboratory, Jan. 1995; http://www.aps.anl.gov/epics/extensions/ezca

J. Chen et al., “CDEV: an object-oriented class library for developing device control applications”, ICALEPCS 1995, Chicago, Illinois, USA, 29 Oct. - 3 Nov. 1995; http://www.jlab.org/cdev

M. Böge, J. Chrin, “An Event Service for Propagation of Data”, SLS Note: SLS-TME-TA-2004-0255, Dec. 1994

Object Management Group's Data Distribution Service, http://portals.omg.org/dds

OpenSplice, http://www.opensplice.com

International Conference on Large and Experimental Physics Control Systems, ICALEPCS'09, Oct. 12-16, 2009, Kobe, Japan