A TASTE OF CAFE

J. Chrin, G. Prekas, Paul Scherrer Institut, 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 provided. 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 or 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 and presents itself as a candidate for Event Processing Agents (EPAs) 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 (DDS), is demonstrated.

POURQUOI CAFE?

Channel Access (CA) is the communication protocol dedicated to EPICS and optimized for the transfer of large amounts of small data packets. The CA client library provides remote access to Process Variables (PVs) residing in the EPICS Input Output Controller (IOC). Application Programming Interfaces (APIs), such as EZCA (E-Z Channel Access) [1] simplify the task of accessing control data by suppling a user-friendly interface to the native CA API. Other APIs, such as CDEV (Common DEVice) [2], provide another layer of abstraction to the CA package and with it certain enhanced features.

However, many of these APIs are not rigorously maintained and do not necessarily reflect the advances in CA functionality. Some APIs appear to be frozen to deprecated functions with little likelihood of the code being brought into compliance with the new CA standard. The most recent EPICS release, for instance, implements multithreading and an improved handling of lost connections. Applications wishing to benefit from these new features would best be served through an in-house API that hooks into the new CA classes. Furthermore, such a development not only allows us to keep in step with future EPICS releases [3], but also presents the opportunity to provide abstract interfaces that are a better match to beam dynamics requirements set by next generation machines, such as the SwissFEL.

A TASTE OF CAFE

CAFE (Channel Access interFacE) [4] is a new C++ library that provides a multifaceted interface to the new CA functions released with EPICS version 3.14 and onwards. Functionality for both synchronous and asynchronous interactions, i.e. monitors, is provided for both individual channels and groups of channels. Operations with both implicit and explicit data types have been implemented and an error handling mechanism raises an exception in the event of a failed transaction. More intricate interfaces tailored towards the needs of beam dynamics applications have also been introduced. These include the use of collections that view related devices as a logical software entity [2]. Rules to flag certain members of a collection is a novel feature that can be applied to effectively reduce the collection to a selected subset. This allows users to readily adapt to changes in the system during operation.

The aggregation of channels into groups (of which a collection is a special type) allows several requests to be delivered with greater efficiency within a single method invocation. Such an approach would be usefully employed in retrieving a snapshot of selected machine data; in a pulsed machine such as the SwissFEL, this will further facilitate in capturing data correlated to a given pulse (or shot).

Data pertaining to a group are encapsulated within a CAFE defined data type, *PVGroup*, which holds a sequence of the CAFE object, *PVDatum*, itself a container for selected PV data (device, attribute, value(s), status, rule, etc.). The data holder for the PV value is a union of CAFE data types, which permits a common representation to be used for data types passed as arguments to CAFE methods.

XML based configuration files provide a convenient framework for defining collections and groups and storing their associated members. Table 1 lists an XML file that defines a collection formed from a series of Digital Beam Position Monitors (DBPMs) and advertises their device attributes. CAFE collection XML files are typically generated from a master XML file representing the entire accelerator. Transactions on CAFE collections are invoked through simple intuitive method invocations that reference collections through their identifier (*cDBPM* in Table 1).

Table 2 illustrates how a CAFE group may be defined in XML with a minimal set of tags. A group may be created from any combination of collections and individual channels. CAFE parses the XML group file, expanding collections into their *PVDatum* constituents, and initializes the resulting *PVGroup*. A failed operation on a group or collection member raises an exception only after all other operations on members have been attempted; data from successful transactions are returned with the exception.

Table 1: XML for CAFE Collections

```
<cafe:config xmlns:cafe=
          "http://fel.web.psi.ch">
  <cafe:collection id="cDBPM" >
    <description>Definition of a collection
    of DBPMs for the 250 MeV Injector
    </description>
    <attributes>
      <attribute> X </attribute>
      <attribute> Y </attribute>
      <attribute> I </attribute>
      <attribute>ENABLE</attribute>
    ...
    </attributes>
    <member>
      <device> FINSS-DBPM10 </device>
    </member>
    <member>
      <device> FIND100-DBPM10 </device>
    </member>
    <member>
      <device> FINSB01-DBPM10 </device>
    </member>
    ...
  </cafe:collection>
</cafe:config>
```

Table 2: XML for CAFE Groups

```
<cafe:config xmlns:cafe=
          "http://fel.web.psi.ch">
  <cafe:group id="gDBPM" >
    <description>Definition of a PVGroup for
    configuring the 250 MeV Injector DBPM EPA
    </description>
    <collection>
      <id> cDBPM </id>
      <attribute> X </attribute>
      <datatype> CA_DOUBLE </datatype>
    </collection>
    <collection>
      <id> cDBPM </id>
      <attribute> Y </attribute>
      <datatype> CA_DOUBLE </datatype>
    </collection>
    ...
    <member>
      <device> FIN-PCT </device>
      <attribute> CURRENT </attribute>
      <datatype> CA_DOUBLE </datatype>
    </member>
  </cafe:group>
</cafe:config>
```

The XML configuration mechanism for initializing CAFE objects allows complex interactions to be initiated with relative ease through specially drafted CAFE methods. The following describes a test application that demonstrates the role of the CAFE API in establishing asynchronous connections to a collection of devices. The monitored data forms the input to agents that aggregate and analyse the data for publication to high-level applications. The operating system in use here is Scientific Linux 5.

EVENT DRIVEN APPLICATIONS WITH XML, CAFE, DDS AND Qt

CAFE is anticipated for use in C++ frameworks such as Qt or ROOT and presents itself as a candidate for event processing agents (EPAs) [5] that, for example, capture machine data for inter-shot analysis at the SwissFEL. An example application is presented that aggregates simulated data from DBPMs to produce events that supply summarized data to interested clients using a publish/subscribe paradigm defined by the Object Management Group's Data Distribution Service (OMG-DDS) [6].

The software architecture is illustrated in Fig. 1 and comprises a CAFE/DDS based EPA for data aggregation, analysis and propagation, and an application layer for the display of summarized data. The OMG-DDS implementation is the OpenSplice (Open Source) Community Edition [7]. The use of DDS to transmit EPICS data is also the subject of other independent work presented at this Conference [8].

The EPA is configured from an XML file (Table 2) that defines the CAFE *PVGroup* object to be used as an input/output argument to the corresponding CAFE methods. The group is first instantiated by reference to it's identifier, *gDBPM*, and a callback mechanism to the group's associated PVs is then established with a single CAFE method invocation.

The EPA uses event pattern rules to aggregate and correlate partially ordered sets (posets) of DBPM data in its input, and creates an high-level event that summarizes the aggregated data for output to DDS.

The DDS is a middleware standard that provides an infrastructure for real-time data distribution between multiple publishers, "DataWriters", and subscribers, "DataReaders", that share an interest in a "Topic", which is the basic data structure expressed in OMG's Interface Definition Language (IDL). A Quality of Service (QoS) framework further provides management of a rich set of attributes that govern reliability, persistency, filtering and ordering of events. In the event delivery model of Fig. 1, the EPA publishes events to the DDS DataWriter residing in the DDS daemon on the publisher's node. The DataWriter publisher subsequently broadcasts (or multicasts as in our configuration) the events across the network. The events are intercepted by those nodes for which a local DDS daemon has the associated subscription. The event data are then passed to the subscriber's DataReader(s) from where they may be retrieved by the client application(s).

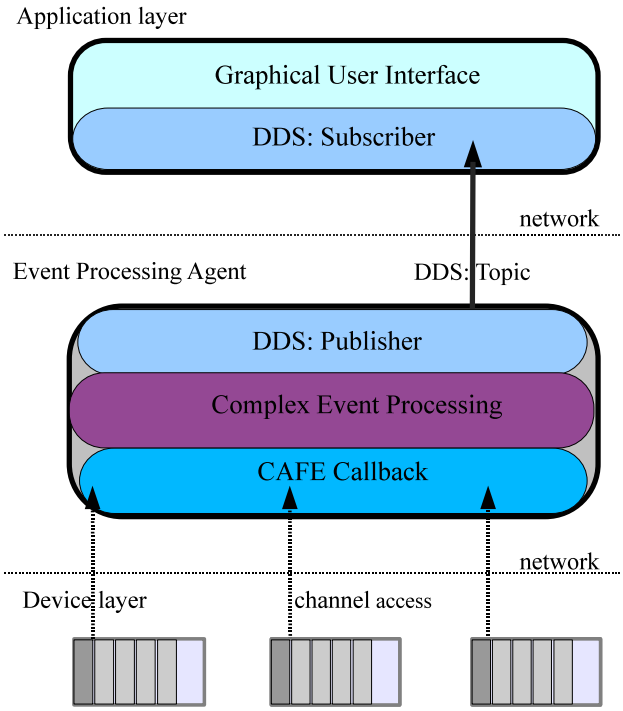

Figure 1: Aggregation of low-level hardware data with CAFE and propagation of summarized data through DDS.

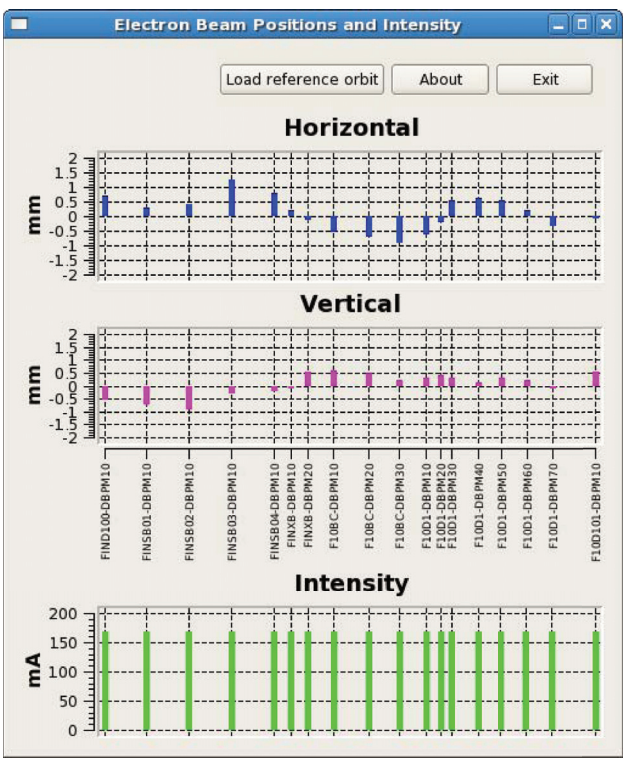

Figure 2: A graphical display of simulated DBPM data built from the Qt (v. 4.3.3) and Qwt (v. 5.1.1) libraries; data are received passively through the assigned DDS Topic. The application is dynamically configured through XML and DDS.

A Qt/Qwt based client application (Fig. 2) displays the summarized data received through DDS. The Qt toolkit is presently being investigated for the development of Graphical User Interfaces (GUIs) in C++. The Qwt class library provides a necessary extension for the development of scientific applications that require, for example, multi-dimensional plotting capability. The Qt/Qwt toolkit is fast and concise, and the test application has proven to be stable and robust even when placed under heavy load; indeed data continues to be displayed effortlessly at submission rates of 100 Hz, even as the application window is playfully resized by the user. The Qt/Qwt toolkit is also the basis for EPICS client application development at KSTAR [9].

APRES CAFE

A first serving of CAFE is scheduled for the commissioning of the 250 MeV Injector test facility for the SwissFEL, the first components of which will be online in December 2009. A subsequent refactoring of the code is anticipated with the intent of making the internal structure more comprehensible and easier to maintain; new functionality as dictated by user requirements will also be implemented. Further CAFE-DDS based EPAs will be added for the aggregation, analysis and propagation of data sets corresponding to the various accelerator nodes, such as magnet types and radio-frequency cavities. An EPA that provides data for online modeling is also foreseen.

Since CAFE is written in C++, bindings to declarative and 4^{th} generation languages are also made possible. PyCafe, a CAFE interface to Python is in preparation with basic read/write functionality already implemented. GUI applications can similarly be developed using PyQt and PyQwt, the respective Python bindings for Qt and Qwt.

REFERENCES

[1] N.T. Karonis. EZCA Primer. Internal Document, Argonne National Laboratory, Jan 1995.

[2] 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.

[3] A. Johnson and R. Lange. Evolutionary plans for EPICS version 3. In *These Proceedings*. ICALEPCS 2009, Kobe, Japan, Oct 2009.

[4] J. Chrin. A Taste of CAFE. SLS Internal Document, 2009.

[5] M. Böge and J. Chrin. An event service for the propagation of data. SLS Note: SLS-TME-TA-2004-0255, Dec 2004.

[6] OMG-DDS. `http://portals.omg.org/dds/`.

[7] OpenSplice. `http://www.opensplice.com/`.

[8] N. Malitsky et al. Prototype of a DDS-based high-level accelerator application environment. In *These Proceedings*. ICALEPCS 2009, Kobe, Japan, Oct 2009.

[9] S. Baek et al. KSTAR widget toolkit using Qt library for the EPICS based control system. In *These Proceedings*. ICALEPCS 2009, Kobe, Japan, Oct 2009.