le Café

A Taste of CAFE

CAFE (Channel Access interFacE):

A new C++ library for remote access to EPICS data

Jan Chrin

FELSI Meeting, 09/12/08

Outline

- Why CAFE?
- Code Listings
  - Synchronous Messages
  - Operations on Groups/Collections
  - Monitors
- CAFE Applications
- Further Developments
- Summary

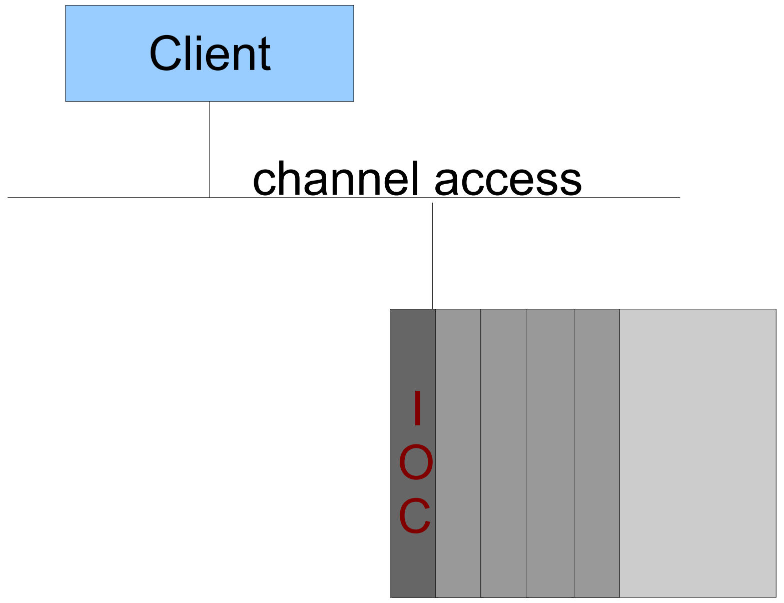
Channel Access (CA)

Dedicated EPICS Protocol providing remote access to records and fields residing on the Input/Output Controller (IOC)

Protocol optimized for transfer of large amounts of small data packets

CA Client Library

=> Initialize CA to receive IOC broadcast messages
=> Client search for Process Variable (PV)
=> Once host containing the requested PV
responds client establishes a "virtual circuit"
=> "Channel" is created (over virtual circuit) between
server and client thru which the PV is accessed
=> All subsequent messages sent thru virtual circuit
=> .....
=> Close CA connection



Standalone CA Clients: Alarm Handler, Archiver,
Motif Editor and Display Manager (MEDM), StripTool, ...

Process Variable (PV):
single value within EPICS host

Channel Access APIs

For High-Level Application development, several APIs simplify task of accessing controls data through user-friendly (and simple) interfaces to the native CA library

<u>EPICS Extensions</u>

C	EZCA: EaZy Channel Access	-> simple interface
C++	CDEV: Common DEVice	-> abstract layer to CA, enhanced features

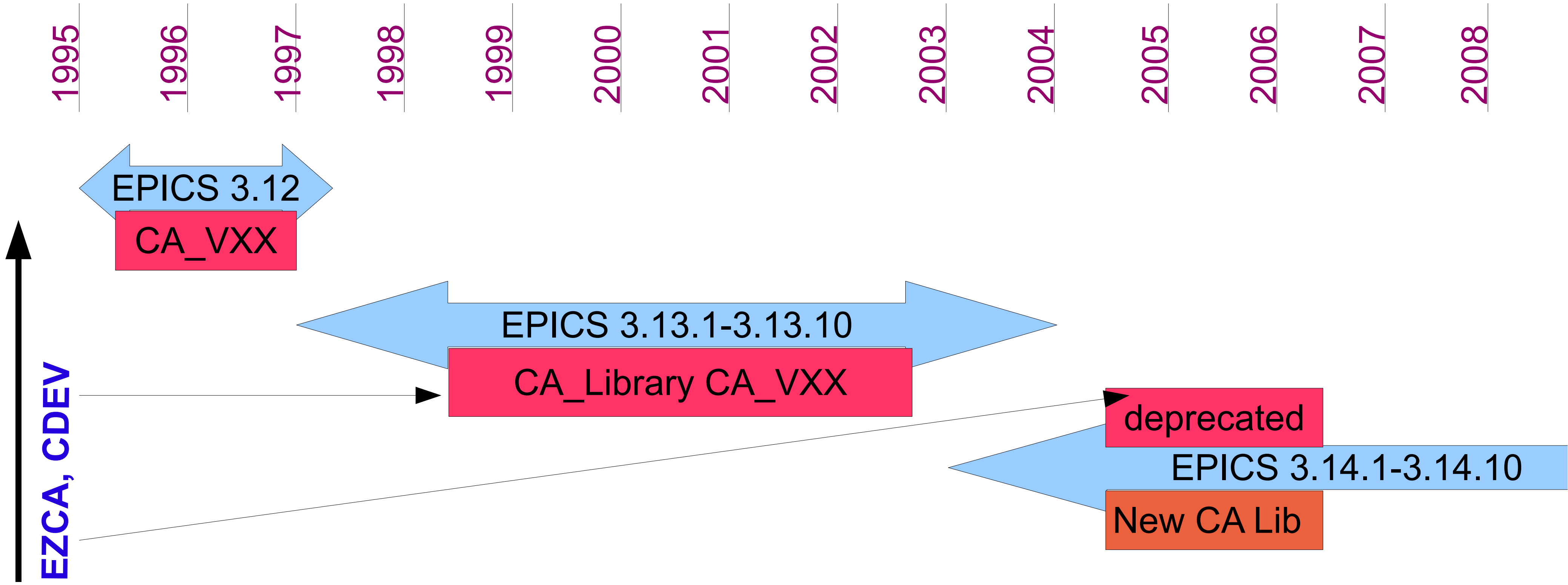
Interface for several other programming languages:

Java, Tcl, Python...

and 4th generation languages:

IDL (EZCA), MATLAB...

EPICS / Channel Access Releases



Only basic CA changes over 10 years guarantee compatibility between old and new client/server connections! Important for sites with legacy systems

Many extensions NOT rigorously maintained and often do not reflect recent advances in channel access (multithreading and handling of lost connections)

CAFE: an in-house API

- Hooks into new CA functions
- Allows to stay in step with EPICS releases
- Presents an opportunity to develop interfaces that are a better match for beam dynamics applications

CAFE provides a multifaceted interface to the new CA functions released with EPICS 3.14 and is tailored towards needs of scientific applications

Functionality for both synchronous and asynchronous interactions, i.e. monitors for individual channels and groups or collections of channels

Synchronous Operations with *Explicit* Data Types

```
#include <cafe.h>

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

   CAFE cafe;
   char st[40]=”1.1”; double d;
   long l[2]={1,2}; float f[2];

   cafe.init(); // Initialize CA

   try{
       // set/get
       cafe.set(“F-CALC”,CA_STRING, 1,st);
       cafe.get(“F-CALC”,CA_DOUBLE, 1,&d);
       // set/get 2 elements of a waveform
       cafe.set(“F-WF1” ,CA_LONG,   2, l);
       cafe.get(“F-WF1” ,CA_FLOAT,  2, f);
   }
   catch(CAFEException &e){
       cafe.printException(e);
   }

   //Release CA resources
   cafe.terminate();
}
```

Process Variable (PV)

CAFE Data type

No. elements

Data

CAFE Primitive Data Types

	C++
CA_STRING	char[40]
CA_SHORT	short
CA_FLOAT	float
CA_ENUM	unsigned short
CA_CHAR	unsigned char
CA_LONG	long
CA_DOUBLE	double

Synchronous Operations with *Implicit* Data Types

```
#include <cafe.h>

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

   CAFE cafe;
   char st[40]="1.1"; double d;
   long l[2]={1,2}; float f[2];

  cafe.init(); // Initialize CA

   try{
      // set/get String
      cafe.setString("F-CALC", st);
      cafe.getString("F-CALC", st);
      // set/get Double
      cafe.setDouble("F-CALC",  d);
      cafe.getDouble("F-CALC", &d);
   }
   catch(CAFEException &e){
      cafe.printException(e);
   }

  //Release CA resources
   cafe.terminate();
}
```

Process Variable (PV)

Data type is Implicit

No. elements=1 is implicit

Data

```
cafe.setFloat("",  f)
cafe.getLong ("", &l)
cafe.setShort("",  s)
cafe.getEnum ("",&us)
cafe.setChar ("", uc)
```

Operations on *Waveforms* with *Implicit* Data Types

```
#include <cafe.h>

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

   CAFE cafe;
   char st[2][40]={"1.1","2.2"};
   double d[2]={1.0,2.0};

  cafe.init(); // Initialize CA

   try{
       // set/get String
       cafe.setStringN("F-WF1", 2, st);
       cafe.getStringN("F-WF1", 2, st);
       // set/get Double
       cafe.setDoubleN("F-WF1", 2, d);
       cafe.getDoubleN("F-WF1", 2, d);
   }
   catch(CAFEException &e){
       cafe.printException(e);
   }

  //Release CA resources
   cafe.terminate();
}
```

Epics channel name

Data type is Implicit

No. elements

Data

```
long n=512
cafe.setFloatN("",n,f)
cafe.getLongN ("",n,l)
cafe.setShortN("",n,s)
cafe.getEnumN ("",n,us)
cafe.setCharN ("",n,uc)
```

Aggregation of Channels into Groups

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

struct PVDatum {
   char pv[40];
   char attrib[20];
   CA_DATATYPE dbrType;
   unsigned long nelem;
   long status;
   long 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;
}
```

Several request delivered with one method invocation, hence greater efficiency

Useful for
-> snapshot of selected machine data
-> group related devices (e.g. Magnets, BPMs, rf cavities, etc.) into collections

All data pertaining to a group are encapsulated within the CAFE defined data type PVGroup

```
PVGroup pvgroup;
//Fill pvgroup dynamically
...
//Take snapshot
cafe.getGroup(&pvgroup);
```

The XML Configuration File for Groups

```
<cafe::config xmlns:cafe=”http://xfel.web.psi.ch/ns”>

  <cafe::group id=gVarious1>

     <cafe::description> snapshot        </cafe::description>
     <cafe::statusGroup> ECA_NORMAL      </cafe::statusGroup>

     // Complete form
     <cafe::member>
        <cafe::name>     F-CV-01:I-SET </cafe::name>
        <cafe::nelem>    1             </cafe::nelem>
        <cafe::status>   ECA_NORMAL    </cafe::status>
        <cafe::rule>     True          </cafe::rule>
        <cafe::dbrType>  CA_DOUBLE     </cafe::dbrType>
     </cafe::member>

     // Minimal form
     <cafe::member>
        <cafe::name>     F-CV-02:I-SET </cafe::name>
        <cafe::dbrType>  CA_DOUBLE     </cafe::dbrType>
     </cafe::member>

   </group>

</cafe::config>
```

Synchronous Operations on Groups

```
#include <cafe.h>

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

   CAFE cafe;
   PVGroup pvgroup;

   cafe.init(); // Initialize CA
   cafe.loadGroups(“cafeGroups.xml”);

   pvgroup =cafe.getPVGroup(“gVarious1”);

   try{
        // Snapshot of group members
        cafe.getGroup(&pvgroup);
   }
    catch(CAFEGroupException &ge){
        cafe.printExceptionSeq(
                    ge.pvgroup.npv, ge.es);
        pvgroup=ge.pvgroup; //recover data
        delete []ge.es;     //release memory
    }

   //Release CA resources
   cafe.terminate();
}
```

Parses
XML configuration file

Retrieves members
for group “gVarious1”

getGroup (or setGroup)

PVGroup struct single
input/output argument

Failed operation
Status for each individual
group member is returned

Collections of related devices

- Group: an aggregation of *unrelated* channels
- Collection: an aggregation of *devices of the same type,* e.g. magnets, BPMs, rf cavities...
- CAFE collection “internally” collapses to a group and uses the “group” methods with the PVGroup struct as single input/output argument
- However, user interface to collections is much simpler!

Device:Attribute Paradigm

MAG-Q1:I-SET ← EPICS Records
MAG-Q1:I-READ
MAG-Q2:I-SET
MAG-Q2:I-READ
MAG-Q3:I-SET
MAG-Q3:I-READ

Related devices can be grouped to form a collection, e.g. "cMAG-Q" from members MAG-Q1, MAG-Q2, MAQ-Q3

Attributes are named parameters belonging to a device

```
cafe.operation("cMag-Q", "I-SET", CA_DOUBLE, d[3]);
```

pointer to data

Collection: a single logical software entity

- Collections first introduced by CDEV (1995)
- However if an operation of just one member of a collection failed, no diagnostics on the individual members would be returned!
- CAFE collection differs in two ways:
  - operations are sent and a status returned for all members of the collection
  - introduction of a rule flag, allows member of the collection to be withdrawn at any time -> client able to respond dynamically to changes in the operating conditions

The XML Configuration File for Collections

```
<cafe::config xmlns:cafe=”http://xfel.web.psi.ch/ns”>

  <cafe::group id=cCV>

     <cafe::description> Vert. Correctors </cafe::description>
     <cafe::statusGroup> ECA_NORMAL       </cafe::statusGroup>

     // Only “device” portion of epics record name entered
     <cafe::member>
        <cafe::name>     F-CV-01:     </cafe::name>
     </cafe::member>
     <cafe::member>
        <cafe::name>     F-CV-02:     </cafe::name>
     </cafe::member>
     <cafe::member>
        <cafe::name>     F-CV-03:     </cafe::name>
     </cafe::member>

   </group>

</cafe::config>
```

Synchronous Operations on Collections

```
#include <cafe.h>

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

   CAFE cafe; double dVal[3];
   bool rule[3]={true,true,true};

   cafe.init(); // Initialize CA
   cafe.loadGroups("cafeGroups.xml");

   try{
      cafe.getCollection("cCV","I-SET"
                    CA_DOUBLE, dVal, rule);
      rule[2]=false;
      for (int i=0; i<3; ++i) dVal[i]+=0.1;
      cafe.setCollection("cCV","I-SET"
                    CA_DOUBLE, dVal, rule);
   }
   catch(CAFEGroupException &ge){
      pvgroup=ge.pvgroup; //recover data
      delete []ge.es;     //release memory
   }

  cafe.terminate(); //Release CA resources
}
```

Parses
XML configuration file

Collection name

Remove 3rd element
from collection and
increment I-SET

Failed operation
Status for each individual
collection member is
returned

Asynchronous Operations

Monitors
- preferred when data changes infrequently
- can be established/removed on individual channels or groups/collections of channels

Requires client to provide a callback function that is invoked whenever the monitored value changes; callback fn can then update local variable or redraw a GUI component

CAFE provides a “generic” callback fn that when triggered inserts the updated value into multimap containers; an independent thread initiates an action in response to a given triggered event

Asynchronous Operations

```
#include <cafe.h>
#include “cafeCallback.h”

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

   CAFE cafe; evid * pEv1, pEv2; evid pEvid[6];
   Pcallback cb;    pCallback usc[6];

   cafe.init(); // Initialize CA
   Try{
      //start monitors
      cafe.startMonitor(“CALC1”, CA_STRING, cb, pEv1);
      cafe.startMonitor(“FIND1-SCA:data”, CA_CHAR, 786432, cb, pEv2);
      cafe.startCollectionMonitor
         (“cCV”, ”I-SET”,CA_DOUBLE, rule, usc, (evid &) pEvid);

      //stop monitors
      cafe.stopMonitor(“FIND1-SCA:data”);
      cafe.stopMonitor( pEvid[2] );
      cafe.stopCollectionMonitor(“cCV”,”I-SET”,rule);

   }
   catch(..){} //CAFEException; CAFEGroupException;
                                             // CAFECollectionException
   cafe.terminate(); //Release CA resources
}
```

Applications with CAFE

- For use in C++ frameworks, e.g. ROOT
- Basis for *event processing agents* (*)

  e.g. capture machine data for

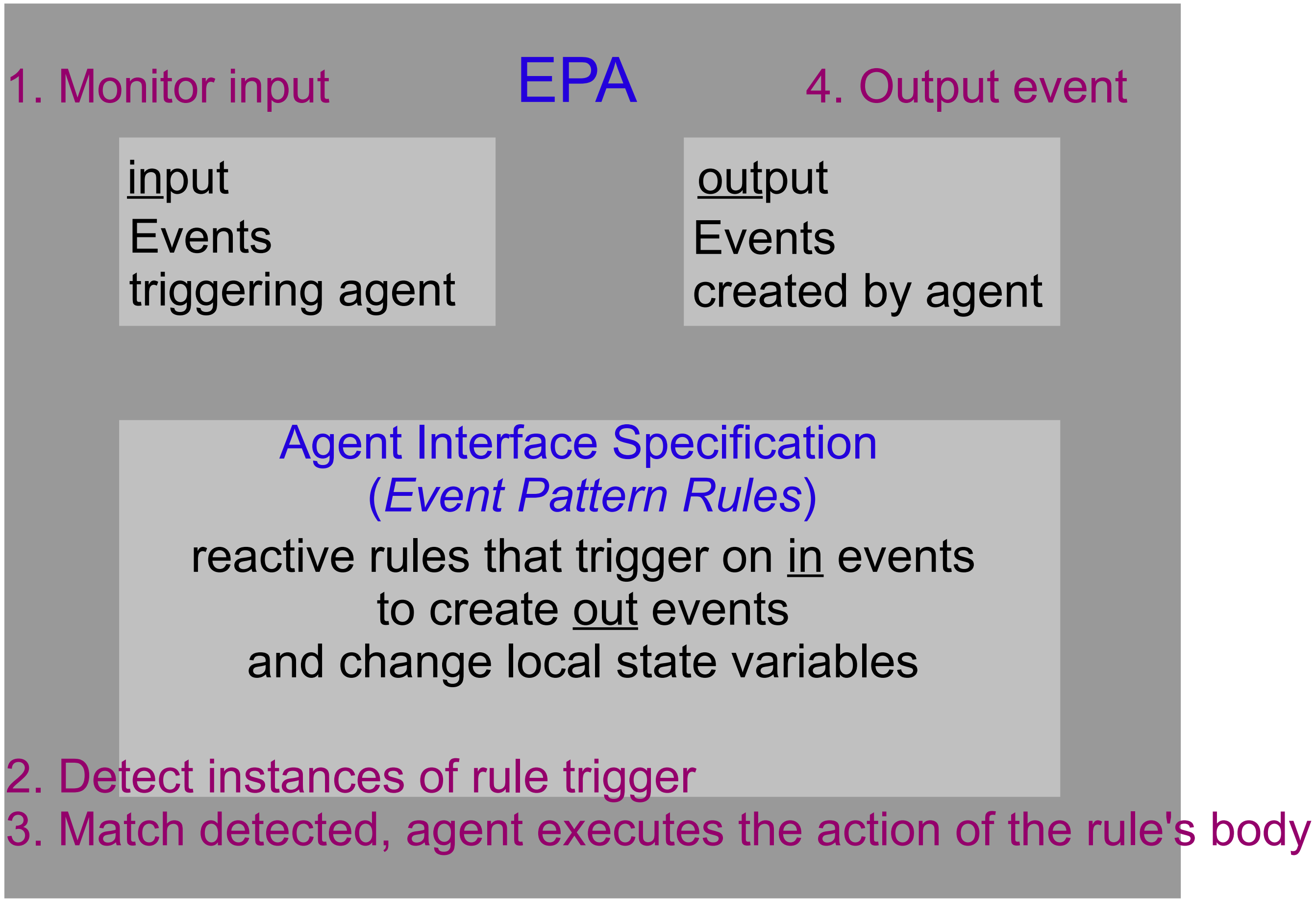
  (i) storage e.g. RDB or HDF

  (ii) inter-shot analysis: 10ms

(*) *Complex Event Processing:* Defined set of tools/techniques for analyzing and controlling events

Event Processing Agents (EPA)

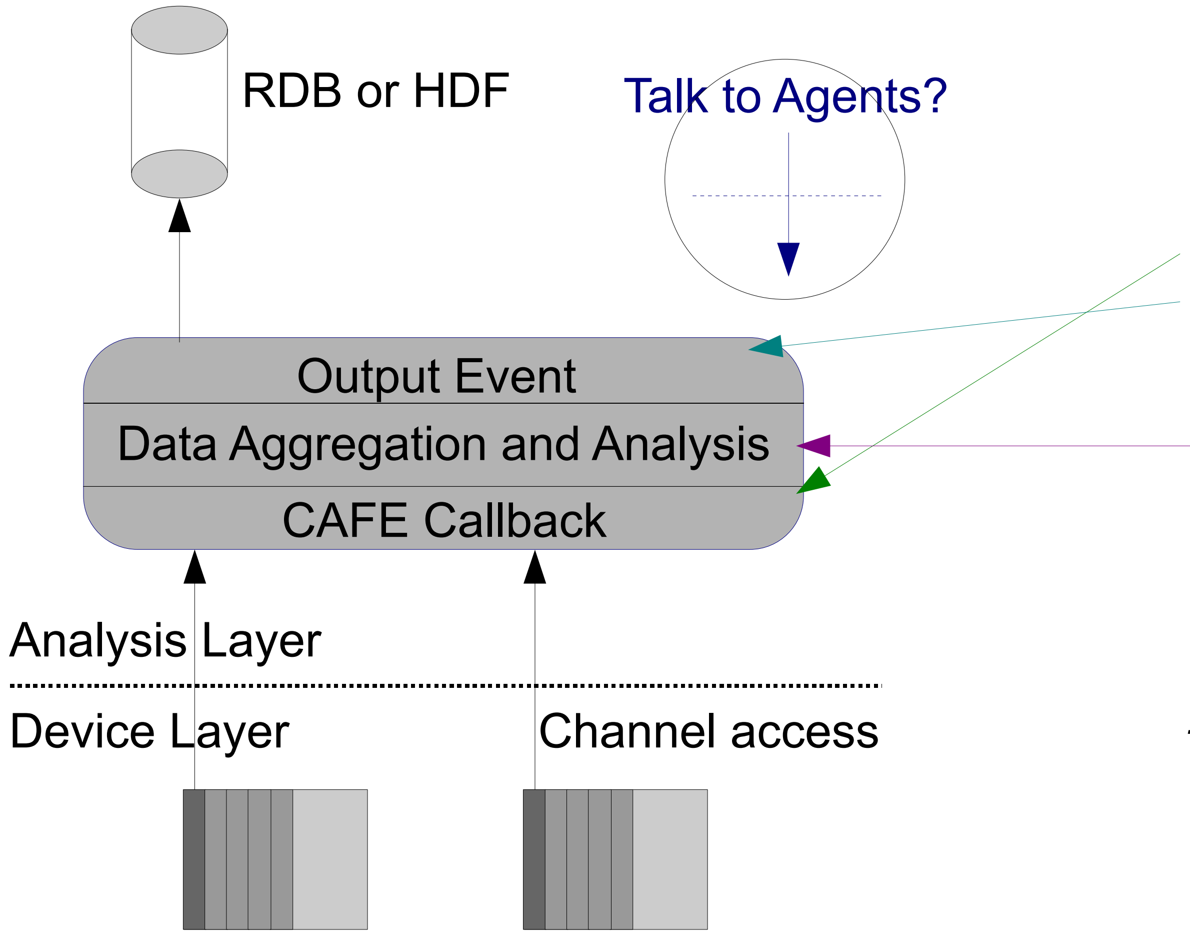
EPA: *event pattern rules,* comprised of a *trigger* and *body of actions* and *local variables* whose values form its state


1. Monitor input
EPA
4. Output event
input
Events
triggering agent
output
Events
created by agent
Agent Interface Specification
(*Event Pattern Rules*)
reactive rules that trigger on in events
to create out events
and change local state variables
2. Detect instances of rule trigger
3. Match detected, agent executes the action of the rule's body

Generic interface to an EPA class

Agent for Data Aggregation and Analysis

run continually on server rather than on demand



```
class  EPA_Class_Name
in input_execution_type
out  output_execution_type {

    Event Pattern Rules
    (Trigger, body of action)

    Local Variables
    ....

}      EPA_Class
       Filter
       Map
       Constraints
```

can be effective in understanding what is happening within a system, and enhance operation and performance (e.g. feedback systems), identify (and solve) problems

Further Developments

- Improve current interface, add new functionality in response to user requests
- XML schema to validate both the syntax and content of the XML configuration file
- Implement *event processing agents* at OBLA 4

  e.g. to capture machine data for storage or inter-shot analysis

  --> test CAFE bandwidth for monitors: shared memory access for maximal performance

Summary

CAFE: a new “in-house” C++ library for remote access to EPICS data

Build on latest CA functions (multithreaded) providing both simple interfaces and structured interfaces for more demanding clients

New software techniques
for
New particle accelerators!

Acknowledgements

- Thomas Schietinger

  Coining CAFE and testing with ROOT

  cafns -> CAFE (Channel Access interFacE)

- Benedikt Oswald

  Automake: generates makefiles to compile and install CAFE

- Mirek Dach

  Soft channels for code development

- Dirk Zimoch

  Matters EPICS

A Taste of CAFE?

v. 1.0.0Beta available

/fel_home/felop/cafe/lib   (SL5)

"A Taste of CAFE"
internal note (draft)

includes Makefiles linking
CAFE and EPICS libraries