PVXS

A(nother) PV Access Client/Server library in C++

Michael Davidsaver
mdavidsaver@ospreydcs.com

Work Sponsored by SNS

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Contents:

There is documentation!

https://mdavidsaver.github.io/pvxs/

Osprey DCS

EPICS

What is PVXS?

PV access in eXcesS ?

- Library providing PV Access network client and server APIs
  - Like pvDataCPP+pvAccessCPP
  - **Inter-operates** with ...
    - And other PVA clients/servers
  - Does ***not*** depend on or use ...
  - Does not conflict with ...
- CLI tools

https://github.com/mdavidsaver/pvxs/tree/master/example

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Why rewrite?

- To Users
  - Existing API is not user friendly
  - Difficult to use and error prone
  - Lots of boilerplate!
- To Maintainer
  - Confusing internals
  - Lingering threading and synchronization issues
- Kay Kasemir (SNS) wrote a new PVA client in Java

History of “simple” PVA

- Wrappers
  - pvaClientCPP / pvDatabaseCPP
  - pva/client.h and pva/server.h (in pvAccessCPP)
    - cf. API usage Examples http://epics-base.github.io/pvAccessCPP/
- Focus on hiding network API complexity
- Data container API as is

Improving Data Containers

- Use libstdc++ definitions where possible
  - *epics::pvData::uint32* → *uint32_t*
- Reduce number of C++ classes
  - *class **Field*** hierarchy (12 classes) avoided
    - Container type definitions represented by “prototype” Value objects
  - *class **PVField*** hierarchy (30 classes) collapsed into single *class **Value***
- FieldBuilder → TypeDef
- Avoid singleton factories (eg. no *getFieldCreate()*)
- Reduce user visible *shared_ptr<>*

Ex: Extract a value from a container

```
PVStructurePtr top = ...;   // maybe result of a Get operation (assume !NULL)
PVIntPtr value = top->getSubField<PVInt>("value");
if(!value)
    throw …;
int32_t val = value->get();
```

Original pvDataCPP (no type conversion)

pvDataCPP w/ type conversion

```
PVStructurePtr top = ...;
int32_t val = top->getSubFieldT<PVScalar>("value")->getAs<pvInt>();
```

PVXS

```
Value top = ...; // Can be “NULL”
int32_t val = top["value"].as<int32_t>();
```

Omitting C++ namespaces
epics::pvAccess::
pvxs::

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Ex: Defining a structure

```
PVStructurePtr top = getFieldCreate()->createFieldBuilder()
                ->add("value", pvInt)
                ->addNestedStructure("alarm")
                   ->add("severity", pvInt)
                ->endNested()
                ->createStructure()
                ->build();
```

C++11 std::initializer_list<>

```
using M = pvxs::members;
Value top = TypeDef(TypeCode::Struct, {
            M::Int32("value"),
            M::Struct("alarm", {
                M::Int32("severity"),
            }),
        }).create();
```

```
Value top = nt::NTScalar{Int32}.create();
```

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

API Design goals

- End user API is first class citizen
- Safety
  - Avoid possible *NULL
  - API enforce required ordering
  - Clear lifetime wrt. cancellation
    - Reference loops still possible w/ callbacks
- Synchronous (blocking) and Asynchronous (callbacks)
- No global ctor/dtor

Ex: Sync. Client GET

```
#include <iostream>
#include <pvxs/client.h>

using namespace pvxs;
client::Context ctxt(client::Config::fromEnv().build());

Value result(ctxt.get("pv:name")
                  .exec()
                  ->wait(5.0);
// wait() throws on timeout

std::cout<<result["value"];
```

*Uses $EPICS_PVA_**

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Ex: Async. Client GET

```
#include <iostream>
#include <pvxs/client.h>

using namespace pvxs;
auto ctxt(client::Config::fromEnv().build());

auto oper(ctxt.get("pv:name")
                .result([](Result&& result) {
                    // on PVA worker thread
                    std::cout<<result()["value"];  // result() throws for remote error
                })
                .exec();

// wait somehow ...
```

C++11 lambda function

Please avoid I/O in callbacks!

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Server API

- PVXS API similar to pva/server.h API in pvAccessCPP
  - class SharedPV
    - *SharedPV is not a record*
- Differences
  - *class ChannelProvider → class Source*
  - No singleton ChannelProviderRegistry
    - pvxs/iochooks.h

Ex: Server API

*Uses $EPICS_PVAS_* or $EPICS_PVA_**

```
auto serv = server::Config::fromEnv().build();

auto initial = nt::NTScalar{TypeCode::Float64}.create();
initial["value"] = 42.0;

auto pv(server::SharedPV::buildMailbox());    // “mailbox” PUT handler accepts anything
pv.open(initial);                             // vs. ::buildReadonly()
                                              // or replace with custom: pv.onPut(...)

serv.addPV(“pv:name”, pv);

serv.run(); // returns on Ctrl+C or serv.interrupt();
// alternative non-blocking   serv.start()
```

Ordering is flexible

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Ex: Unit test w/ Isolation

```
// auto serv = server::Config::fromEnv().build();
auto serv = server::Config::isolated().build();

serv.addPV("some:pv", …);

auto cli = serv.clientConfig().build();

serv.start();

cli.get("some:pv").exec()->wait(5.0);
```

Setup Server listening on localhost w/ random port

Create client Context which connects only to this Server.

Caveats

- Requires toolchain w/ C++11
  - GCC >= 4.8
  - MSVC >= 2015 / 12.0
- Depends on libevent >=2.0
  - https://libevent.org/
  - Bundling + cross build via git submodule
- Heavy use of C++ features
  - eg. python ctypes out of luck
- Concurrency
  - >libca, <pvAccessCPP

- PVA only
  - No wrapper for CA
  - Focus on doing PVA “right”
- New development
  - API stabilizing
  - Test results wanted
    - negative and positive!

cf. Building from Source

https://mdavidsaver.github.io/pvxs/

Osprey DCS

EPICS

Going Forward (1)

- Initial Development
  - Q4 2019
- Public Beta
  - Q1 2020 → Announce on Tech-talk list
- Pre-production 0.X
  - Q4 2020? → Begin release notes. eg. on API changes
- Stable 1.X
  - 2021? → 1.X API “freeze”

aka. Incompatible change → 2.X

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

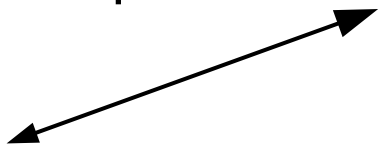
Going Forward (2)

pvDataCPP/pvAccessCPP

- Staged deprecation
  - End of feature development
    - Q1 2020
  - Critical fixes only
    - 2021?
  - Removal (from Base releases)
    - 20222?

PVXS

- v1.0 ?
- Expand use of PVXS
  - Gateway
  - IOC integration
    - QSRV
    - PVA Links
  - CLI tools
  - Language bindings



Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Metrics (Source Lines of Code)

- PVXS
  - 13.5 k LoC++
- libevent_core
  - ~20 k LoC

- pvDataCPP
  - 14 k LoC++
- pvAccessCPP
  - 23 k LoC++
- normativeTypesCPP
  - 6 k LoC++
- pvaClientCPP
  - 6 k LoC++
- pvDatabaseCPP
  - 6 k LoC++

~55 k LoC++

Base
libCom 60 k
libca 25 k

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS

Metrics (Exec. Text Size)

Base

- Com – 0.4MB
- ca - 0.3MB
- dbCore – 0.5MB
- dbRecStd – 0.3MB

PVXS

- pvxs – 1MB
- event_core – 0.2MB

pv*CPP

- pvData – 1.2MB
- pvAccess – 1.4MB
- nt – 0.5MB
- pvaClient – 0.6MB
- pvDatabase – 0.4MB

gcc 8.3 w/ -O3 on Linux/amd64

https://mdavidsaver.github.io/pvxs/

Metrics (gcov)

GCC Code Coverage Report

By PVXS unittest suite

			Exec	Total	Coverage
Directory:	../../				
Date:	2020-10-20 12:42:34	**Lines:**	5187	7351	70.6 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %	**Branches:**	4351	11605	37.5 %

File	Lines			Branches	
base-git/include/epicsEvent.h		100.0 %	1 / 1	- %	0 / 0
base-git/include/epicsGuard.h		100.0 %	16 / 16	- %	0 / 0
src/O.linux-x86_64/bitmask.cpp		93.7 %	89 / 95	73.3 %	66 / 90
src/O.linux-x86_64/bitmask.h		96.4 %	27 / 28	50.0 %	2 / 4
src/O.linux-x86_64/client.cpp		80.6 %	366 / 454	36.0 %	309 / 858
src/O.linux-x86_64/clientconn.cpp		70.7 %	147 / 208	30.9 %	144 / 466

C++ Exceptions?

Osprey DCS

https://mdavidsaver.github.io/pvxs/

EPICS