Channel Access

Kay Kasemir

kasemirk@ornl.gov

With material copied from

- Bob Dalesio (LANL)
- Ned Arnold (APS)
- Ken Evans (APS)

Feb 2021
Channel Access: The EPICS Network Protocol

- Read and write Process Variables over the network.
- To many, CA is EPICS.
  - Especially to users of systems that have no IOC database.
  - "Integrate into EPICS" can mean: Talk CA on the network.

Since ca. 1990.
Alternative: PV Access
Consider a ‘news’ website...

- People create web pages with news
- http:// serves them

- Doesn’t mean you can list all the people, or get the color of their socks
- People might change
- Some pages are created by programs, they don’t wear socks!

- Records on IOCs provide data
- Channel Access serves them

- Doesn’t mean you can list all records
- IOCs might change
- Some channels are provided by python, LabVIEW, …, there are no records!
Keep in mind

- The protocol http:// is different from the people who create web sites
- The Channel Access (and PV Access) protocol is different from the IOCs and records

This ‘decoupling’ has proven essential but is often forgotten!
What is a Process Variable?

Good question!

"A named piece of data with attributes"

Consider this record:

```plaintext
record(calc, "t1:calcExample")
{
    field(DESC, "Sawtooth Ramp")
    field(SCAN, "1 second")
    field(CALC, "(A<10)?(A+1):0")
    field(INPA, "t1:calcExample.VAL")
}
```
What is a PV, given that record?

- "t1:calcExample"
  - PV for the current value of the record.
  - Number 0…10, changes each second.

- "t1:calcExample.DESC"
  - PV for the DESC (description) field of the record.
  - String "Sawtooth Ramp", static.

- "t1:calcExample.VAL"
  - Same as "t1:calcExample".

- “t1:calcExample.SCAN”
  - “1 second”, type enumerated, static.

Pretty much every field of a record can be a PV:
- "{record name}.{field name}"
- ".VAL" is implied when omitting field
'caget', 'caput'

'caget' command-line tool:

> caget tl:calcExample
tl:calcExample                 6

> caget tl:calcExample.VAL
tl:calcExample.VAL             9

> caget tl:calcExample.DESC
tl:calcExample.DESC  Sawtooth Ramp

'caput' allows writing:

> caput tl:calcExample.DESC  "Howdy"
Old : tl:calcExample.DESC  Sawtooth Ramp
New : tl:calcExample.DESC  Howdy
'camonitor'

'camonitor' monitors value changes:

> camonitor t1:calcExample

```
  t1:calcExample 2006-10-06 13:26:03.332756 6
  t1:calcExample 2006-10-06 13:26:04.332809 7
  t1:calcExample 2006-10-06 13:26:05.332866 8
  t1:calcExample 2006-10-06 13:26:06.332928 9
  t1:calcExample 2006-10-06 13:26:07.332981 10
  t1:calcExample 2006-10-06 13:26:08.333034 0
  t1:calcExample 2006-10-06 13:26:09.333097 1
  t1:calcExample 2006-10-06 13:26:10.333143 2
```

... plus one more each second...
... press Ctrl-C to stop ...

> camonitor t1:calcExample.DESC

```
  t1:calcExample.DESC 2006-10-06 13:29:12.442257 Howdy
```

... and then nothing ...

AKA publish and subscribe.
How Clients find Channels

Wants to read "PS1:Voltage"

Channel Access

IOC

IOC

IOC

IOC

Meter

Power Supply

Camera

Has PV "PS1:Voltage"
Internet 101

- The Internet Protocol (IP) consists of UDP and TCP
  - We ignore the very low-level Internet Control & Message Protocol (ICMP).

- User Datagram Protocol (UDP)
  - Sends a network packet
    - from one port on one computer
    - to one or more ports on one or more other computers.
    - ..with one or more listeners on the target port
  - Fast!
  - Checksum: If the packet arrives, it's OK.
  - Not reliable: Packets get lost, arrive out-of-order, arrive more than once.

- Transmission Control Protocol (TCP)
  - Sends a stream of bytes from one port on one computer to another port on another computer, with exactly one listener on the target port
  - Reliable: Bytes arrive at the receiver in the correct order.
    - Basically adds serial numbers to UDP packets, requesting repeats for missing packages.
  - Slower, and message boundaries get lost:
    - "Hello Fred!" might arrive as "Hel" <pause> "lo F" <pause> "red!"
Search and Connect Procedure

1. UDP Broadcast Sequence
   Who has it?

2. UDP Reply
   I have it!

3. TCP Connection
   Let’s talk!

Client → Client → Client → Client → Client → Client

Check → Check → IOC → Check

Meter → Power Supply → Camera
Search Request

- A search request consists of a sequence of UDP packets
  - Per default: Broadcast to the local subnet.
    - Basically plug-and-play when you get started.
  - Or to IP addresses listed in EPICS_CA_ADDR_LIST
    - Routers do not forward broadcasts!
    - You have to add 'other' subnets or specific IOCs off the local subnet to that environment variable!
  - Starts with a small interval (30 ms)
    - Doubles each time, until reaching 5 second intervals.
    - Stops after 100 packets (~8 minutes) or when it gets a response
    - Wakes again on "beacon anomaly" (details follow later)

- CA Servers check each search packet
- Usually connects on the first packet or the first few
  - But non-existent PVs cause a lot of traffic
  - Try to eliminate them
Important Environment Variables

- **EPICS_CA_ADDR_LIST**
  - Determines where to search
  - Is a list (separated by spaces)
    - “123.45.1.255 123.45.2.14 123.45.2.108”
  - Default is broadcast addresses of all interfaces on the host
    - Works when servers are on same subnet as Clients
  - Broadcast address
    - Goes to all servers on a subnet
    - Example: 123.45.1.255
    - Use `ifconfig -a` on UNIX to find it

- **EPICS_CA_AUTO_ADDR_LIST**
  - YES: Include default addresses above in searches
  - NO: Do not search on default addresses
  - If you set EPICS_CA_ADDR_LIST, usually set this to NO
EPICS_CA_ADDR_LIST

EDM → EDM → Client → Client → Client → EDM

Broadcast:
123.45.1.255

Specific:
123.45.2.2

Subnet:
123.45.1.x

CA Server:
123.45.1.1

 IOC:
123.45.1.2

Subnet:
123.45.2.x

 IOC:
123.45.2.1

 IOC:
123.45.2.2

Not Included:
Channel Access in One Slide

Process Variables:
- S1A:H1:CurrentAO
- S1:P1:x
- S1:P1:y
- S1:G1:vacuum

CA Client
- Who has a PV named “S1A:H1:CurrentAO”?
  - I do.
  - What is its value?
    - 25.5 AMPS
  - OK, it is now 30.5
  - It is now 20.5 AMPS
  - It is now 10.5 AMPS
  - It is now -0.0023 AMPS

CA Server
- Change its value to 30.5
- Notify me when the value changes

- “set a monitor”
- “connection request” or “search request”
  - “get” or “caGet”
  - “put” or “caPut”
  - “put complete”
  - 30.5 is too high. It is now set to the maximum value of 27.5.
  - You are not authorized to change this value
- “post an event” or “post a monitor”
Multiple IOCs on Host

- IOCs on IP 1.1.1.1, net 1.1.1.0
  1. UDP 5064, TCP 5064
  2. UDP 5064, TCP ???

- Try to reach from other subnet
  - EPICS_CA_ADDR_LIST=1.1.1.1
    - Won’t work!
    - Quirk in network kernels:
      Only the IOC started LAST will get anything on UDP 5064
  - EPICS_CA_ADDR_LIST=1.1.1.255
    - OK. When using broadcast into subnet, all IOCs on UDP 5064 will see search requests.
Firewall?!

- IOCs on IP 1.1.1.1, subnet 1.1.1.0
  - UDP 5064, TCP 5064
  - UDP 5064, TCP ???
  - EPICS_CA_ADDR_LIST=1.1.1.255

- Firewall cannot open unpredictable TCP ???

- Likely to block broadcasts

- Need to run CA Gateway:
  - Firewall allows access to CAGateway
  - CAGateway uses broadcast inside subnet
Handling of Network Interruptions

- No Network is up 100%, so CA was designed to handle this:
  - TCP connection closed by server?
    - Notify client code about problem
      - Operator displays tend to indicate this.
    - Client sends new search requests.
  - No data nor beacon from server for 30 sec.?
    - Client sends “Are you there?” query
    - If no response for 5 sec, also notify client code, but TCP connection is kept open to avoid network storms.
    - If server eventually sends data: OK. Otherwise we're waiting until the OS cuts the TCP connection (~hours).
Beacons

• Assume all is fine, we are connected, but the data simply doesn't change.
  – How do we know the server is still OK?

• Assume we searched for a PV, didn't get any response for ~8 minutes.
  – How do we learn about a new CA server starting up which might have the missing PV? What triggers renewed search requests?
Beacons

- UDP broadcast packet sent by a CA Server
- When it is healthy, each Server broadcasts a UDP beacon at regular intervals (like a heartbeat)
  - EPICS_CA_BEACON_PERIOD, 15 s by default
- When it is coming up, each Server broadcasts a startup sequence of UDP beacons
  - Starts with a small interval (~30 ms)
  - Interval doubles each time until reaching 15 sec
- Clients monitor the beacons
  - Receive beacons: Server is OK.
  - Receive new beacons at changing intervals: Beacon anomaly, new CA server, restart searches.
caRepeater?

- Older OSs didn't allow multiple programs to listen to the same UDP port
  - They didn't see the beacons (UDP broadcasts)!

- caRepeater solves this problem
  - There is one caRepeater process per workstation
  - Clients make a TCP connection to it when they start up
  - caRepeater receives the beacons
    - EPICS_CA_REPEATER_PORT [usually 5065]
    - .. and forwards them to clients.

![Diagram showing the relationship between IOC1, IOC2, IOC3, CaRepeater, and Clients](image-url)
Issues

- CA Client does not connect
  - Check basic network connectivity.
    - Can server and client machines 'ping' each other?
    - Check EPICS_CA_ADDR_LIST if server is on different subnet.

- CA Client does not re-connect after network issue or IOC reboot
  - Use casw, wireshark: Does the client computer receive the (anomal) beacons of the rebooting IOC?
  - Check EPICS_CAS_BEACON_ADDR_LIST, since routers will not forward beacons across subnets.
  - Check if 'caRepeater' is running on the client.
What is a PV (Channel)?

- Whenever there's a CA server out there which decides to respond to a search request, that's a PV!
- iocCore responds to "{record}.{field}" searches if
  - the {record} is for a record on this IOC,
  - and the {field} is an accessible field of that record,
  - or it's the pseudo-field "RTYP" (record type).
- So every field of every record is a PV.
- But you can implement your own CA server based on the CAS library (for C++), or the pcaspy wrapper for Python, and then you decide when to respond!
Channel Properties

- Each channel comes with properties:
  - Value
    - of type string or double or int or …
    - Scalar or array
  - Time stamp
    - Up to nanosecond precision
  - Severity code
    - OK, MINOR, MAJOR, or INVALID
  - Status code to qualify the severity
    - OK, READ error, WRITE error, at HIGH limit, …
  - units, suggested display range, control limits, alarm limits.
Client interface to properties

- The available properties are fixed.
  - One cannot add a new 'color' property.

- The request types are fixed.
  - "DBR_…" types.
  - Available:
    - Just value.
    - Value with status and severity.
    - Value with status, severity and time stamp.
    - "Everything:" value, units, time, status, limits, ...
  - Not available:
    - Custom combinations like value with units.

- See `caget –h`
Records & Fields vs. Channels & Properties

- A CA client asks for the properties of a channel.
- The implementer of the CA server decides how to answer.
- The iocCore implementation maps the fields of a record to the properties of a channel.
  - Details are in the source code for the respective record type. Not always predictable or meaningful!
Example: AI record "fred"

- PV "fred" or "fred.VAL"
  - value property of channel = VAL field of record.
    - Type double, one element (scalar).
  - time property = TIME field
  - status = STAT
  - Severity = SEVR
  - units = EGU
  - Precision = PREC
  - display limit low, high = LOPR, HOPR
  - control limit low, high = LOPR, HOPR
  - alarm limits = LOLO, LOW, HIGH, HIHI

- Makes a lot of sense.
  - GUI can display the value together with units, formatted according to the precision, as e.g. "12.37 volts".
Example: AI record "fred"

- PV "fred.SCAN"
  - value property of channel = SCAN field of record.
    - Type enumerated, values: "Passive", "1 second", ...
  - time property = TIME field?
  - status = STAT?
  - Severity = SEVR?
  - control limit low, high = 0, ??
When will ‘camonitor’ receive new value?

- When the CA server (IOC) sends a new value!
  - Analog records: VAL change $\geq$ MDEL
  - Binary records: Every change

- Assuming Client uses ‘DBE.VALUE’ subscription
  - DBE_LOG
    - Meant for archive systems. Analog record change $\geq$ ADEL
  - DBE_ALARM
    - Meant for alarm systems
Database Channel Access Link Flags

- **CA**: Force CA link, even though target in same IOC
- **CP**: For INP link, process on received CA monitor
- **CPP**: CP, but only if SCAN=Passive

Allows for “process record if inputs change”
Points to remember

- In 99% of the cases, CA "just works"
  - If not, check EPICS_CASCADE_ADDR_LIST
  - If that's not it, there could be a subnet/router issue with UDP search broadcasts and beacons.

- Channel/property and Record/field are different things!
  - This decouples the CA clients from the IOC database and its record types, allowing EPICS collaborators to share CA client tools for vastly different records and databases.
  - But also means that CA clients have no idea about records nor fields.
    - Client can't know that there might be a "readback" AI that goes with a "setpoint" AO record.
    - The archiver stores channels and their properties, not a whole AI or motor record.
    - Important properties for dealing with waveform data is definitely missing (sample rate, type of data).