Channel Access
Server Tool

Developers Training

Jeff Hill, Kay-Uwe Kasemir,
LANL
CA Servers & Clients

IOC

Server

Host

Client: EDM

Host

Client: Archiver

IO

Client: CA Link

CA

Server

Source
CA Server Library: CAS

- C++ library for WIN32, Solaris, Linux, ...
- Part of EPICS/base/...:
  - Include: include/casdef.h
  - Library: lib/<arch>/cas.a, cas.lib, ...
  - Sources: src/cas/...
  - Examples: src/cas/example
Export Data to EPICS

CA Protocol

CAS Library

Server Side tool

Data Source/Store

Your Task
CAS Library API

- Four Classes
  - Server - “caServer”
  - Process variable - “casPV”
  - Channel (optional) - “casChannel”
  - Asynchronous IO (optional) - “casAsyncXxxIO”

- Override virtual methods
- Uses GDD class (Gen. Data Descriptor) for portable data handling
- Driven by EPICS fdManager
Server Tool Responsibilities

- Respond to PV existence test requests: override `caServer::pvExistsTest`
- Attach client to named PV: override `caServer::createPV`
- Process PV read requests: override `casPV::read`
- Process PV write requests: override `casPV::write`
- Notify server library when PV changes: call `casPV::postEvent`
GDD

- Reference counted
  - Allocate dynamically
  - Add/delete reference, removes itself when no longer referenced

- Three types of GDDs
  - Scalar
  - Vector (Atomic)
  - Container (e.g. value + time stamp + limits)

- Characterized by
  - primitive type: integer, float., ...
  - application: value, time, limits, units ...

- gddAppFuncTable.h
  Helper class to dispatch read requests by application, also for containers
Example:

- Extremely Simple CA Server

- `<EPICS base>/src/cas/example/simple`
- more in `<EPICS base>/src/cas/example`
Caveats

- There is no EPICS database at work! Your server tools decides what channels to serve.

- CAS helps by handling not only DBR_DOUBLE but also e.g. DBR_CTRL_DOUBLE requests. If you fill those container requests, clients can see the control limits, units, etc.

- BUT: If you serve “fred”, there is no “fred.VAL” nor “fred.HIHI” unless you serve that, too, as separate PVs.
Advanced “caServer”

- Optional virtual member functions
  - show server tool state: watch clients attach..

- Ordinary member functions
  - register new event type
Advanced “casPV”

- Optional virtual member function
  - maximum matrix dimension and bounds
  - client interest (event subscription) notification
  - begin / end transaction notification
  - no clients attached to PV “destroy” hint
  - create channel (for access security)
  - show
Asynchronous IO

- The server tool should *not* block when completing a client initiated request
- Currently four IO operations can be completed asynchronously
  - PV read
  - PV write
  - PV exist test
  - PV attach
Completing IO Asynchronously

- Create appropriate asynchronous IO object
- Return S_casApp_asyncCompletion
- When the IO completes
  - call asynchronous IO object’s “postIOCompletion()”