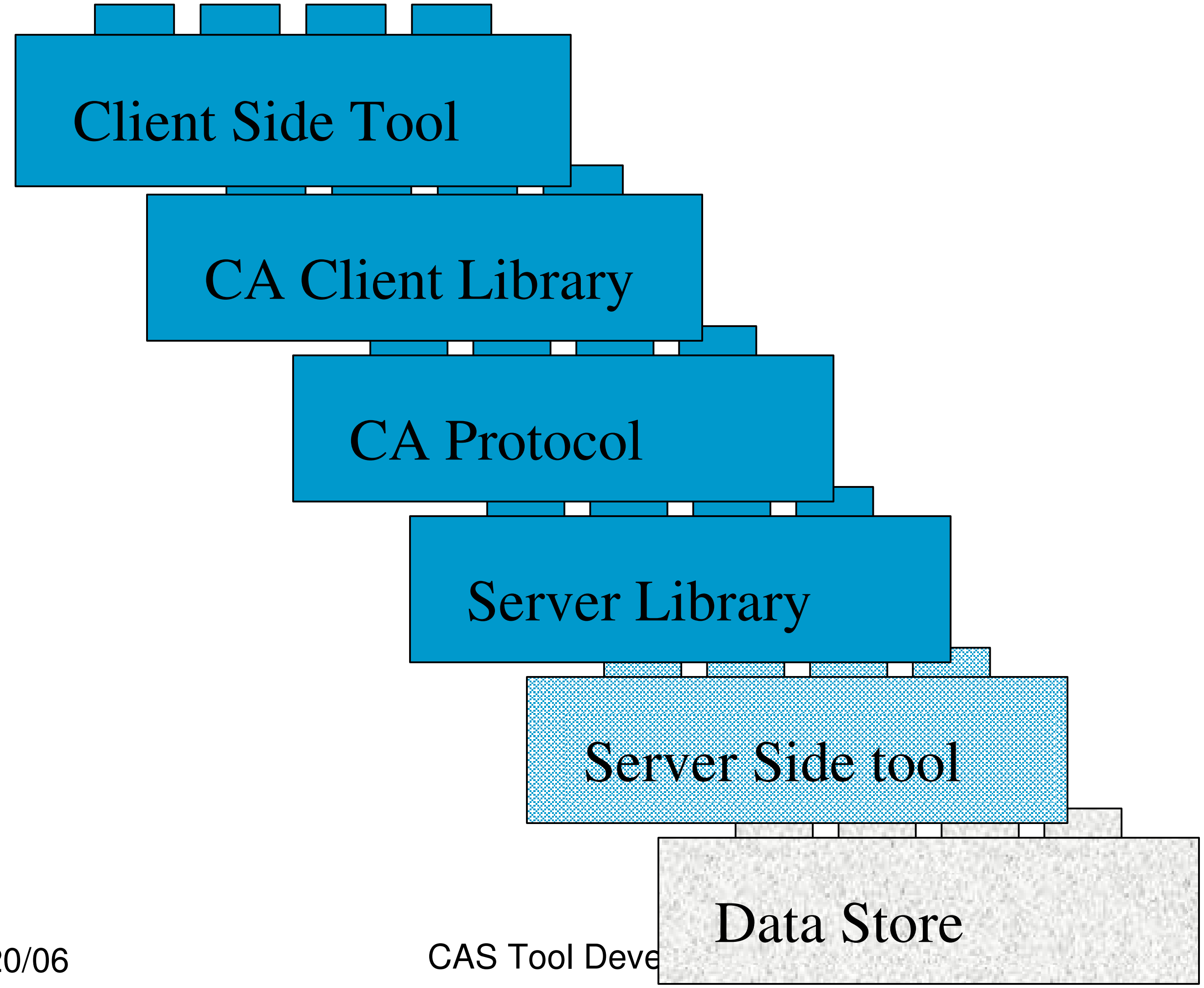
Channel Access Server Tool Developers Training

Jeff Hill

Export Data to EPICS



Leverage EPICS Tool Set

Client Side Tools

- Operator Interfaces
- Alarm Manager
- Data Archives
- Data Analysis
- 4th Generation Languages
- Active X / DDE

Server Side Tools

- Function Blocks
- Gateways to other systems
- Data Analysis
- CA Proxy (Gateway)
- 4th Generation Languages
- Active X

Client Side Tool Capabilities

- Locate process variable (PV)
- Read process variable (PV)
- Write process variable (PV)
- Subscribe for event notification
  - – process variable (PV) value change
  - – alarm state change
  - – connection state change
  - – access right change

Server Tool Responsibilities

- Respond to PV existence test requests
- Attach client to named PV
- Process PV read requests
- Process PV write requests
- Notify server library when PV state change events occur

Server Application Programmers Interface (API)

- C++ based
  - server tool derives from base classes
- Ordinary class member functions
  - server tool requests to the server library
  - supplied by library
- Virtual class member functions (VF)
  - client requests to server tool
  - supplied by server tool

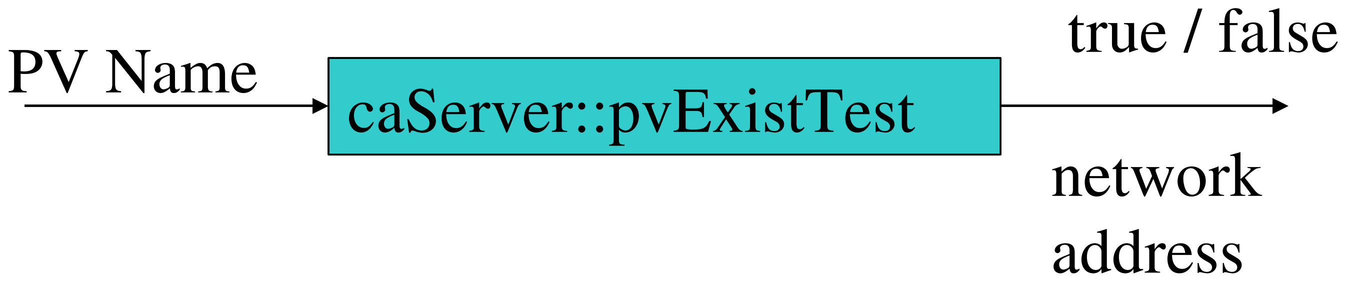
Four Classes in the API

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

Server Class - “caServer”

- Required virtual member functions
  - – named PV existence test
  - – attach to named PV
- Optional virtual member functions
  - – show server tool state
- Ordinary member functions
  - – register new event type

Server Tool Supplied PV Name Directory (VF)

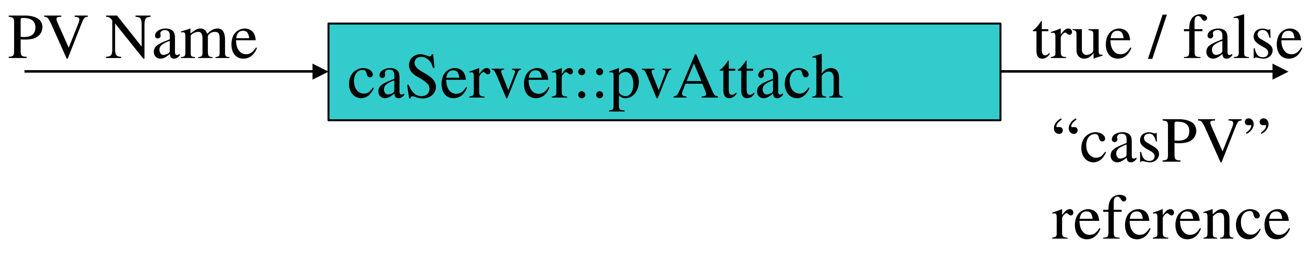


Note:

PV name could be an alias.

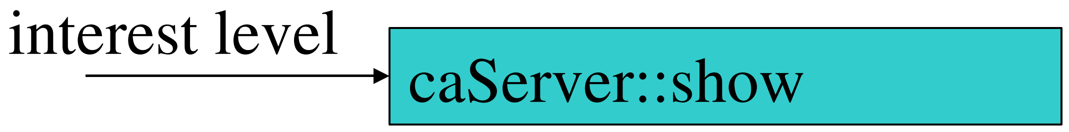
String hashing support libraries are available in EPICS base. See example server tool.

Server Tool Supplied PV Object Factory (VF)



Note:
A C++ "reference" is a special form of pointer which cant be NULL. While the reference here is to a "casPV" the object returned is actually some server tool invented class deriving from "casPV".

Server Tool Supplied Diagnostics Dump (VF)



Note:
Server tool provides increasing diagnostics information to “stdout” with increasing “interest level”. The default action in the base class is to dump the internal state of the server library.

Server Tool Registers New Event Type Name With Library

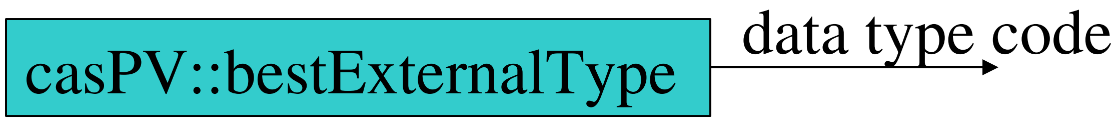
$$\begin{tikzcd}
\text{Event Name} \arrow[r] & \boxed{\text{caServer::registerEvent}} \arrow[r] & \text{Event Mask}
\end{tikzcd}$$

Note:
Currently, the protocol supports only 3 “built in” event types: value change, archive value change, and alarm state change events.

Process Variable Class - “casPV”

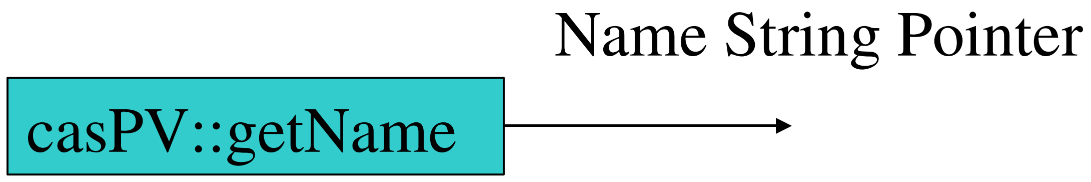
- required virtual member function
  - – best external primitive data type
  - – process variable name
  - – read / write

Server Tool Supplied Best External Data Type (VF)



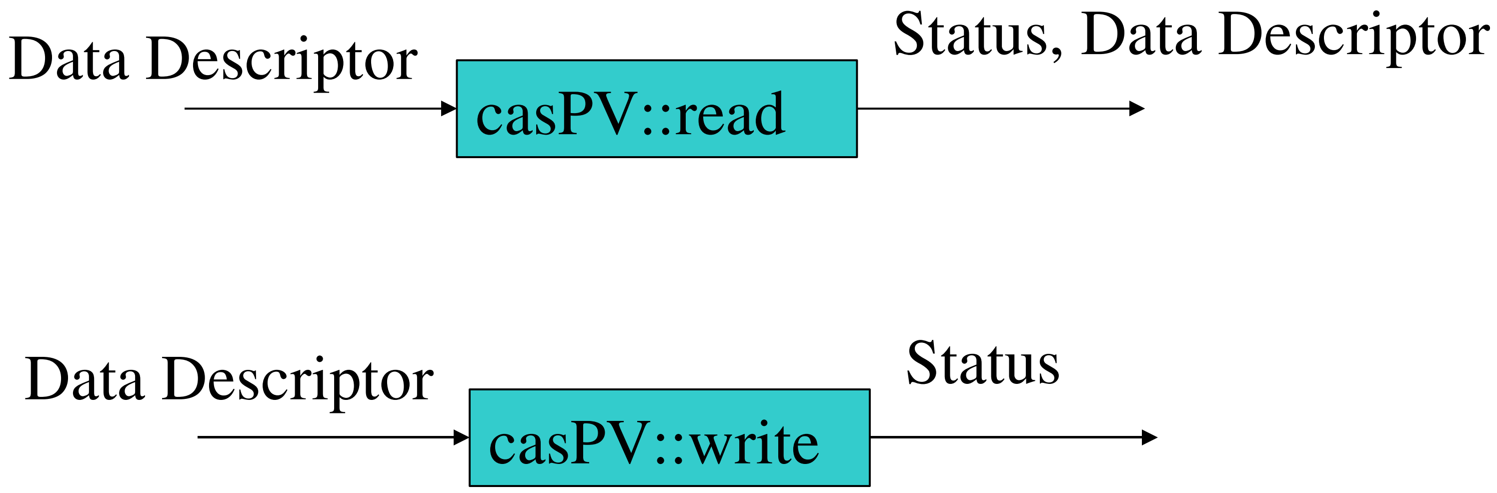
Note:
client tools frequently use this “primitive” data type code to infer if the “value” attribute of the process variable is analog, discrete, or enumerated. The default primitive type is a character string.

Server Tool Supplied Process Variable Name (VF)



Notes:
Returns the canonical name of the PV and not an alias.
Name string pointer must remain valid during the life span of the PV.

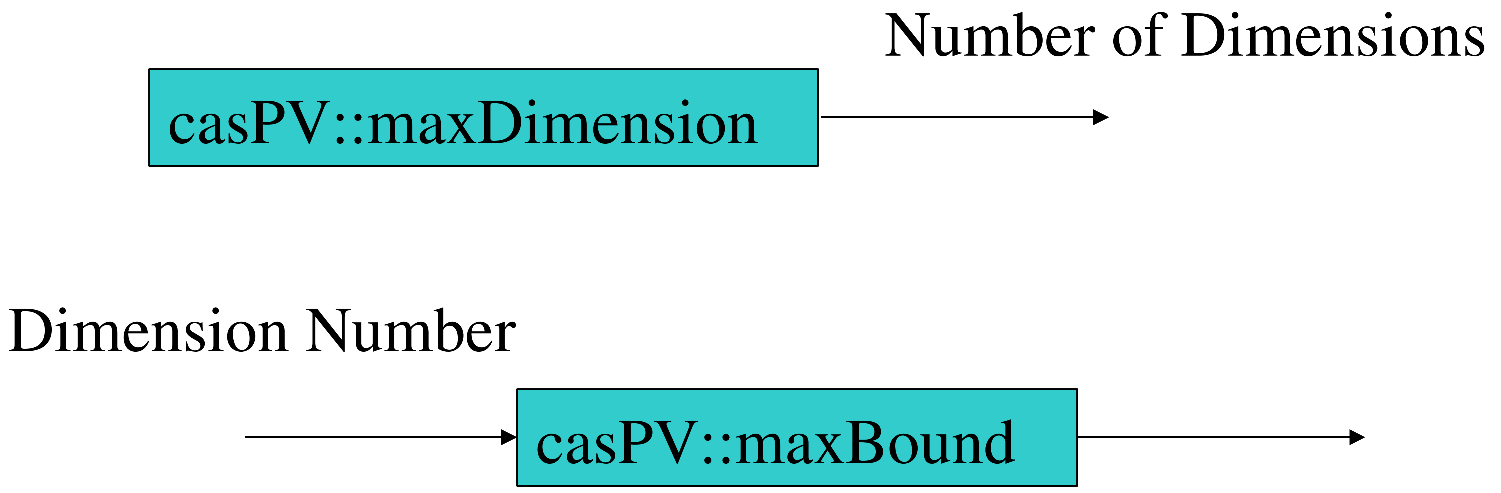
Read / Write PV (VF)



Process Variable Class - “casPV”

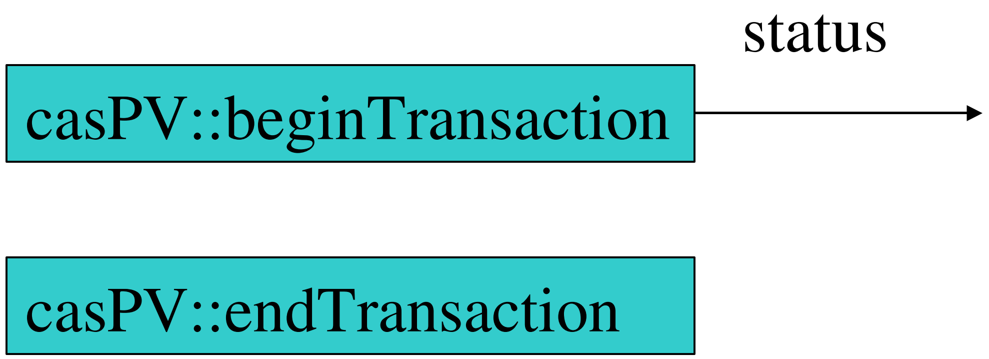
- optional virtual member function
  - maximum matrix dimension and bounds
  - client interest (event subscription) notification
  - begin / end transaction notification
  - no clients attached to PV “destroy” hint
  - create channel
  - show

Server Tool Supplied Maximum Matrix Bounds (VF)



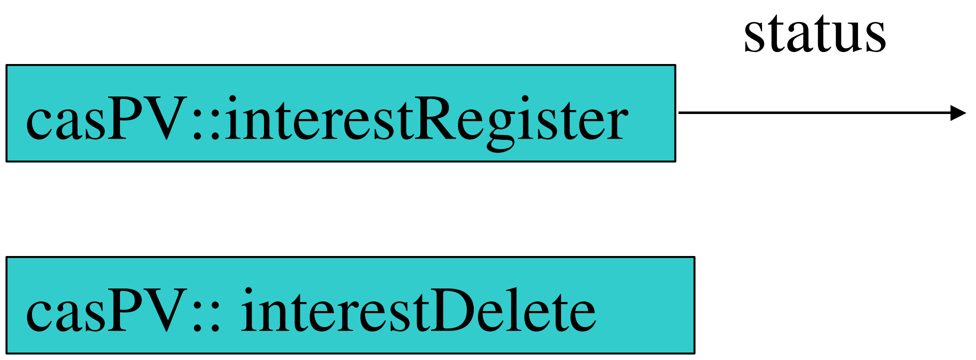
Note:
The default is scalar bounds

Server Tool Supplied Begin / End Transaction (VF)



Note:
These functions are called immediately before and immediately after each read or write operation respectively.

Server Tool Supplied Client Interest Notification (VF)



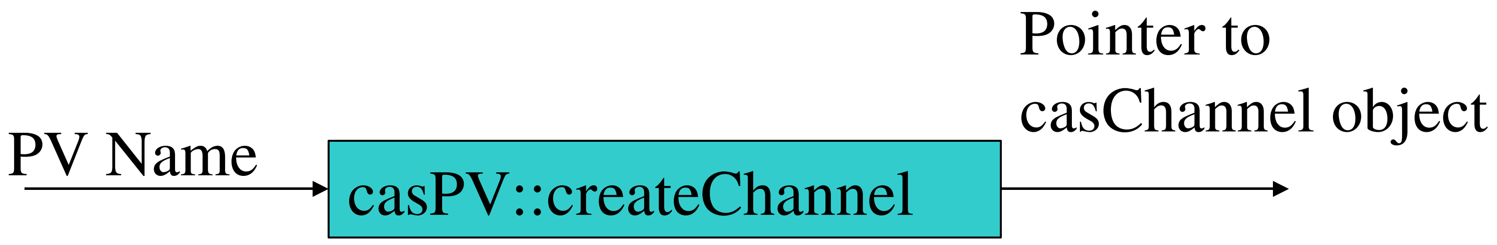
Note:
These functions are called when the first client’s event subscription is added and the last client’s event subscription is removed respectively.

Server Tool Supplied No Clients Attached Hint(VF)

casPV::destroy

Note:
This function is called when the last client disconnects from the PV. The default action in the base class is to C++ “delete” the PV. It is acceptable to ignore this “destroy” hint.

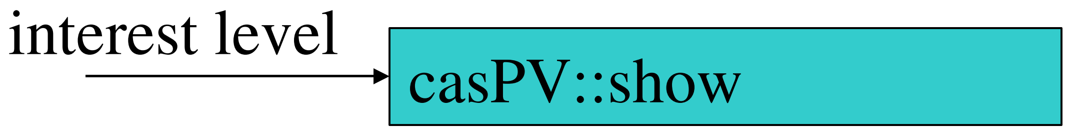
Server Tool Supplied Channel Object Factory (VF)



Note:
The channel object is currently only used for:
o Access control
o Determining the host and user that is attached to the PV

The default action is to create the casChannel base class.

Server Tool Supplied Diagnostics Dump (VF)

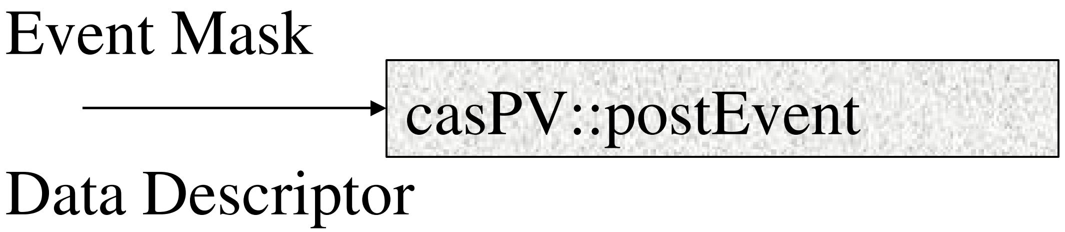


Note:
Server tool provides increasing diagnostics information to “stdout” with increasing “interest level”. The default action in the base class is to dump the internal state of the server libraries casPV base class.

Process Variable Class - “casPV”

- ordinary member functions
  - – post process variable state change event
  - – return pointer to the server object

Server Tool Posts Process Variable State Change Events



Notes:
Currently, the protocol supports only 3 “built in” event types: value change, archive value change, and alarm state change events.
Data descriptor reference counting guarantees that the data descriptor will not be released by the server library until event delivery to each client

Server Tool Requests Pointer to the PV's Server Object

$$\begin{tikzcd}\boxed{\text{casPV::getCAS}} \arrow[r, "\text{server pointer}"] & {}\end{tikzcd}$$

Notes:
It is possible for a PV to exist, but not be attached to a server, and in this situation the function returns NULL

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()”

Data Descriptors

- GDD C++ based class library is used
- Three types of GDDs
  – Scalar
  – Vector (Atomic)
  – Container

GDD Data Types

- primitive type
  - 32 bit floating point, 16 bit integer, string ...
- application type
  - value, limits, units …
- gddAppFuncTable.h
  - links to server tool's function for each application type

GDD Reference Counting

- GDD created with a reference count of one
- When reference count goes to zero
  - GDD's C++ destructor is called
- Resulting limitation
  - GDD can *only* be created in pool with the new operator

GDD Reference Counting

- Store a new pointer to the GDD
  - Increment reference count
- Throw away a pointer to GDD
  - Decrement reference count
- GDD smart pointer class painlessly manages all of this for you

Documents of Interest

- CA Server Library Tutorial
- CA Server Library Reference
- GDD Reference Manual
- All on the Web

Example Server Tool Source Code

- <EPICS>/base/src/cas/examples/simple