PV Access Java API

EPICS Collaboration Meeting

Slovenia, Sept. 2022

Kay Kasemir

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

History: Two compatible Implementations

Initial Implementation (Since ~2014)

C++: pvDataCPP, pvAccessCPP, …

Java: pvDataJava, pvAccessJava, …

Python: pvaPy

Gateway: pva2pva

- ✓ Included in EPICS 7: softIocPVA, 'QSRV', pvget/put/info/monitor
- ✓ Used in successful operation
- Same API for C++ & Java: Lowest common denominator, missing language advantages.
- Bugfixes, but no additions.

Updated Implementation (~2020)

C++: PVXS

Java: core-pva

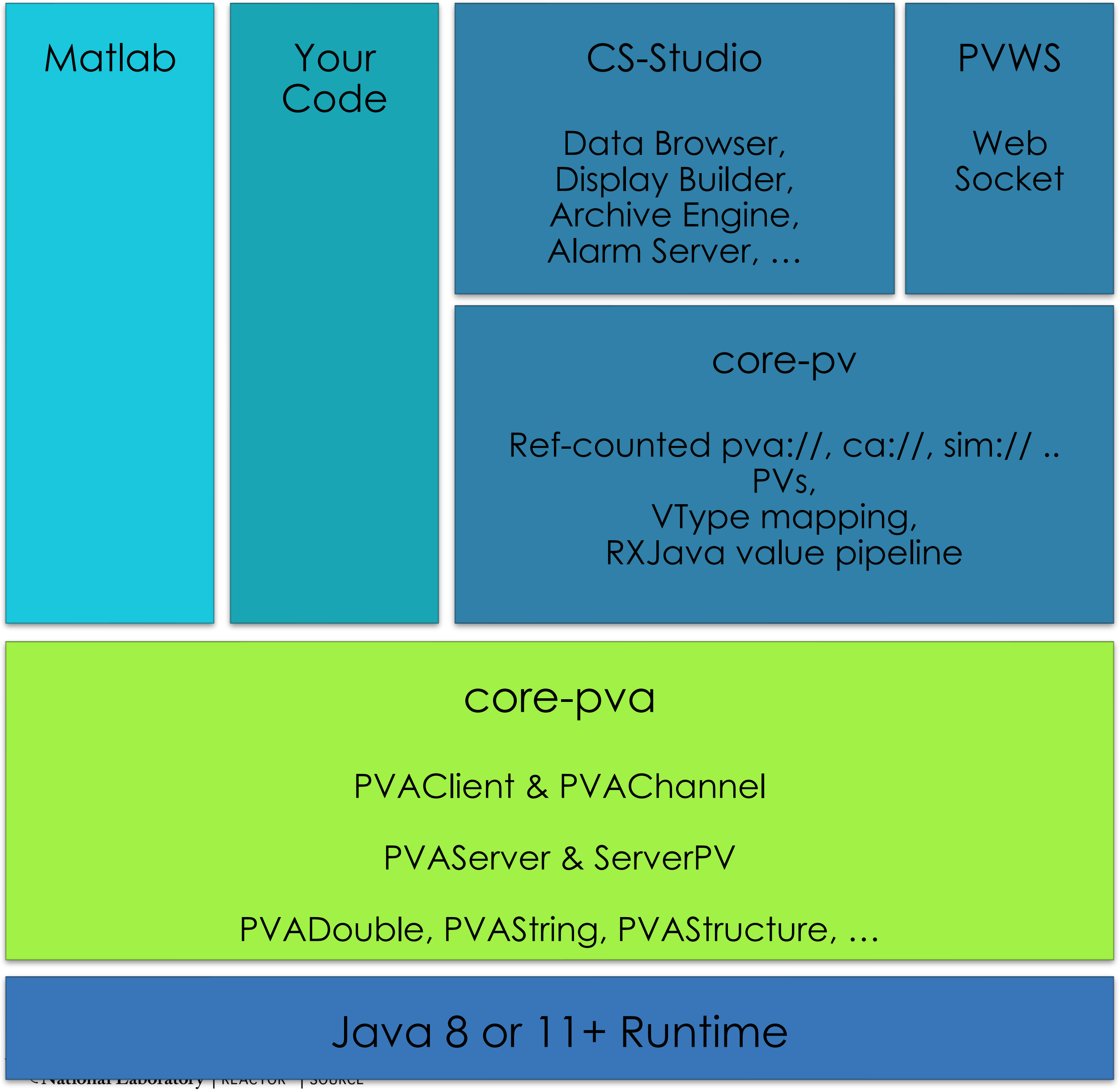
Python: p4p

Gateway: p4p gateway

- ✓ APIs take advantage of each language
- ✓ Gateway's "fair" scheduling helps with arrays; known UDP port allows use via firewalls
- ✓ Active Development
  - ✓ IPv6 support
  - ✓ EPICS_PVA_NAME_SERVERS for TCP-only usage
- Not in EPICS base, yet.

Same Protocol!

`core-pva` Java Library



Matlab

Your Code

CS-Studio

Data Browser, Display Builder, Archive Engine, Alarm Server, …

PVWS

Web Socket

core-pv

Ref-counted pva://, ca://, sim:// .. PVs, VType mapping, RXJava value pipeline

core-pva

PVAClient & PVAChannel

PVAServer & ServerPV

PVADouble, PVAString, PVAStructure, …

Java 8 or 11+ Runtime

- No dependencies beyond JRE
- Source code:
  - https://github.com/ControlSystemStudio/phoebus/tree/master/core/pva
  - Builds with “mvn clean install” or “ant clean core-pva”
- Releases:
  - https://search.maven.org/search?q=core-pva
- Sources are compatible with Java 8 for usage with Matlab, but you’ll have to compile with Java 8 yourself. Released binaries are for Java 11 or higher.

Examples

- Source code: See src/test/…

- Here we'll use https://github.com/kasemir/JavaPVADemo

- In VM terminal:

```
cd /ics/examples/JavaPVADemo
code .
```

.. and follow the README.MD

Summary

- `core-pva` is the current Java Client library
  - CS-Studio
  - Matlab (java8)
  - Any Java code (java 11+ binaries are released)
- Also includes simple PVA Server
  - Mostly for tests at this time
- Basic access to all PVA functionality

pvDataJava & pvAccessJava   vs.   core-pva

- Replaced large interfaces with “@FunctionalInterface”
- Future<..the data..> instead of custom callback interfaces
- Plain ‘instanceof’ instead of custom data API
- PVADouble, PVAString, PVAStructure, … define a type and hold the data.
  Original implementation had twice the API because defining type was separate from holding data.
  In practice, all but the ‘pvinfo’ command needs type & data.
  Now ’info’ simply returns PVA… with values all zero/null/empty.