Paper ID: TUPHA146

Interface Between EPICS and ADO

Andrei Sukhanov, James Jamilkowski
Brookhaven National Laboratory, Upton, NY, USA

ABSTRACT

EPICS is widely used software infrastructure to control Particle Accelerators, its Channel Access (CA) network protocol for communication with Input/Output Controllers (IOCs) is easy to implement in hardware. Many vendors provide CA support for their devices. The control systems of the Collider-Accelerator Department (C-AD) at Brookhaven National Laboratory (BNL) is a complex system consisting of approximately 1.5 million control points. The control of all devices is unified using an Accelerator Device Objects (ADO) software abstraction layer. In this paper we present software solutions for cross-communication between two different platforms. They were implemented for the integration of a NSLS II Power Supply Controller hardware into the RHIC Controls System.

EPICS

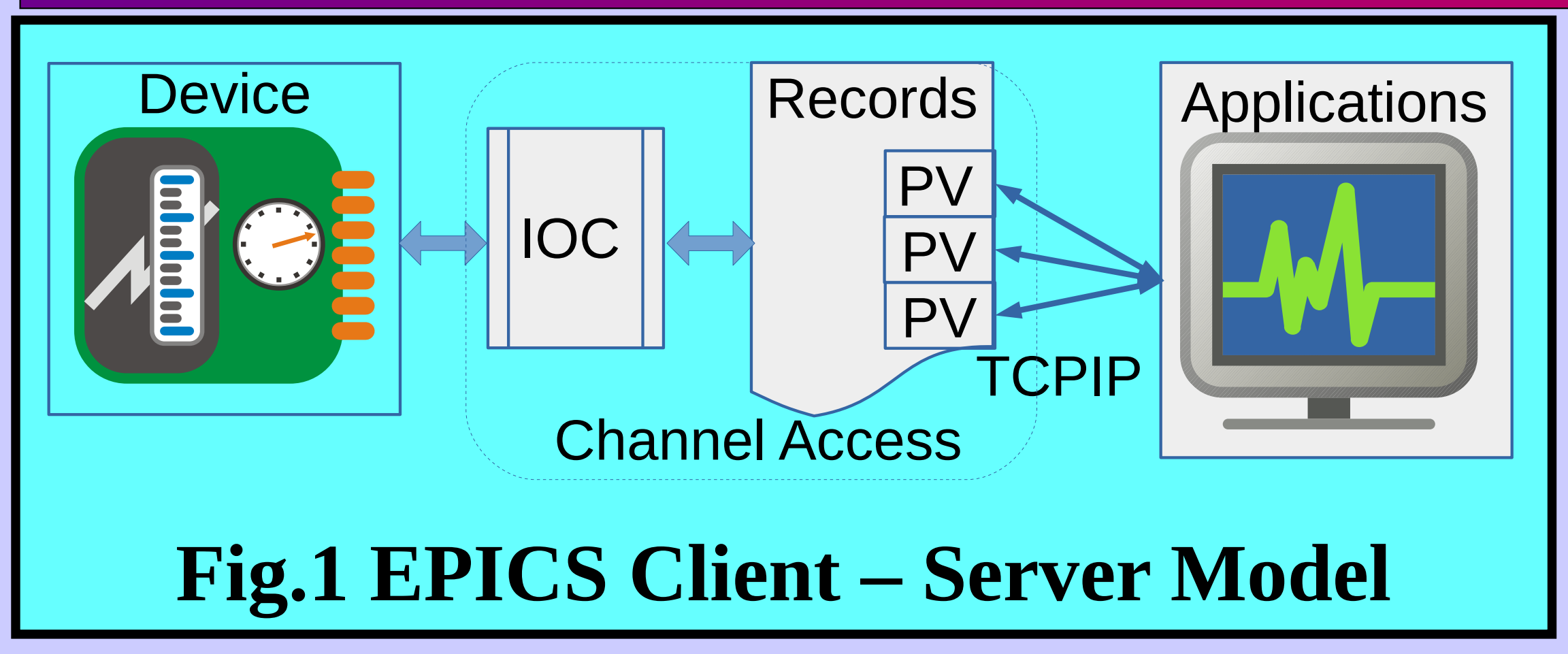


Fig.1 EPICS Client – Server Model

- **Sites**: Over 100 independent projects around the globe.
- **Number of controlled parameters**: from 1K to 300K.
- **Base Source Code**: Open Source. Language: C since 1989.
- **Client-Server Model**: Device is controlled by IOC which provide support for 'records'. Records provide access and control to process variables (PV).
- **Client-sever protocol**: EPICS Channel Access protocol.
- **Transport layer**: TCPIP.
- **Name service**: No central name service.
- **Number of lines of the Base Code**: 200K of C code.

RHIC Controls

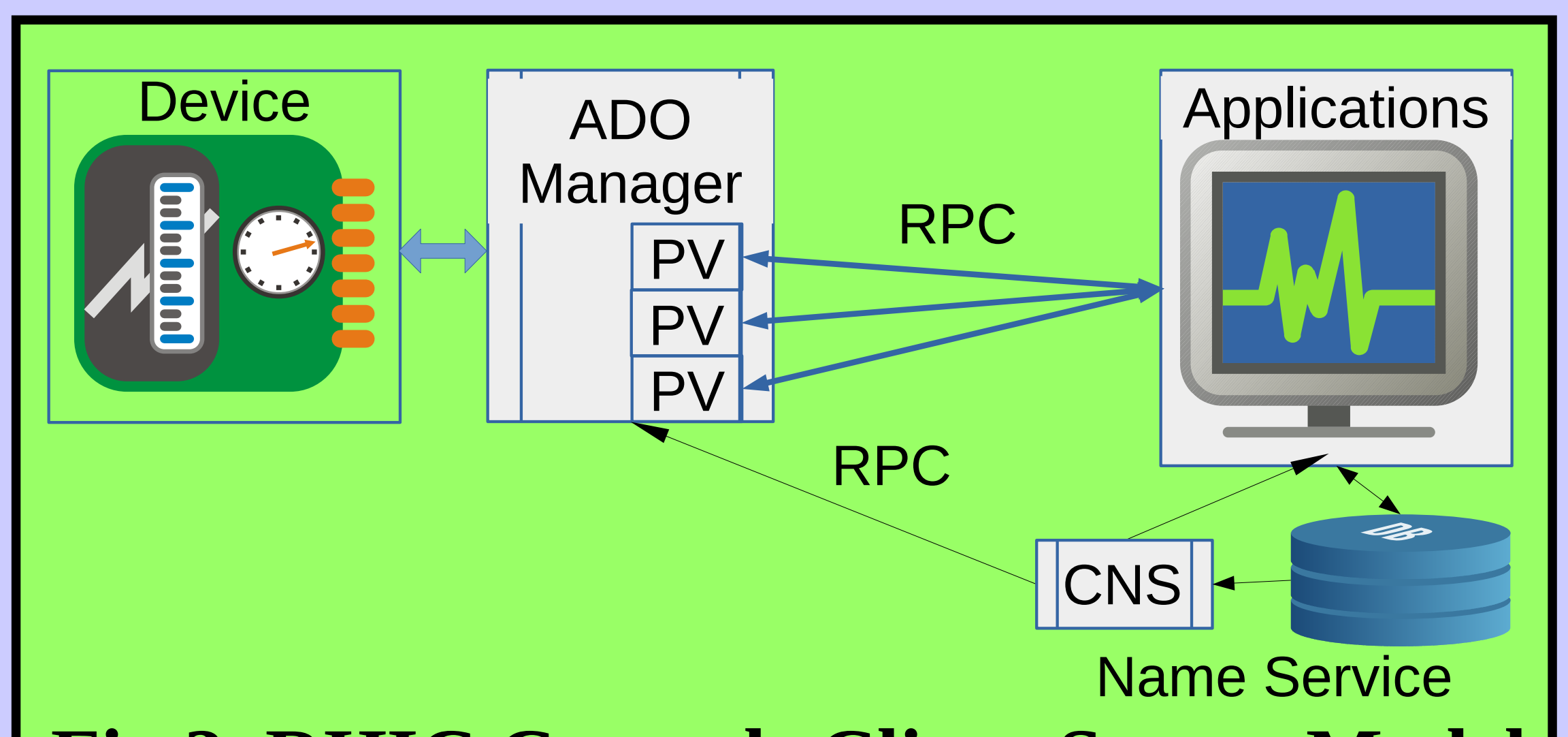


Fig 2. RHIC Controls Client-Server Model

- **Site**: RHIC Collider complex at BNL, Upton, NY, USA.
- **Number of controlled parameters**: ~1.5 million.
- **Base Source Code**: Proprietary. Language: C++ since 1995, Python since 2016.
- **Client-Server Model**: Device is controlled by ADO manager, which provides the process variables.
- **Client-sever protocol**: RPC.
- **Transport layer**: TCPIP.
- **Name service**: RPC server program, connected to DB (Sybase).
- **The character set of PV naming** is more limited than in EPICS, name translation is necessary.
- **Number of lines of the Base Code**: 5K for Python implementation.

EPICS DEVICE in RHIC ENVIRONMENT

Option 1: Two-way translation

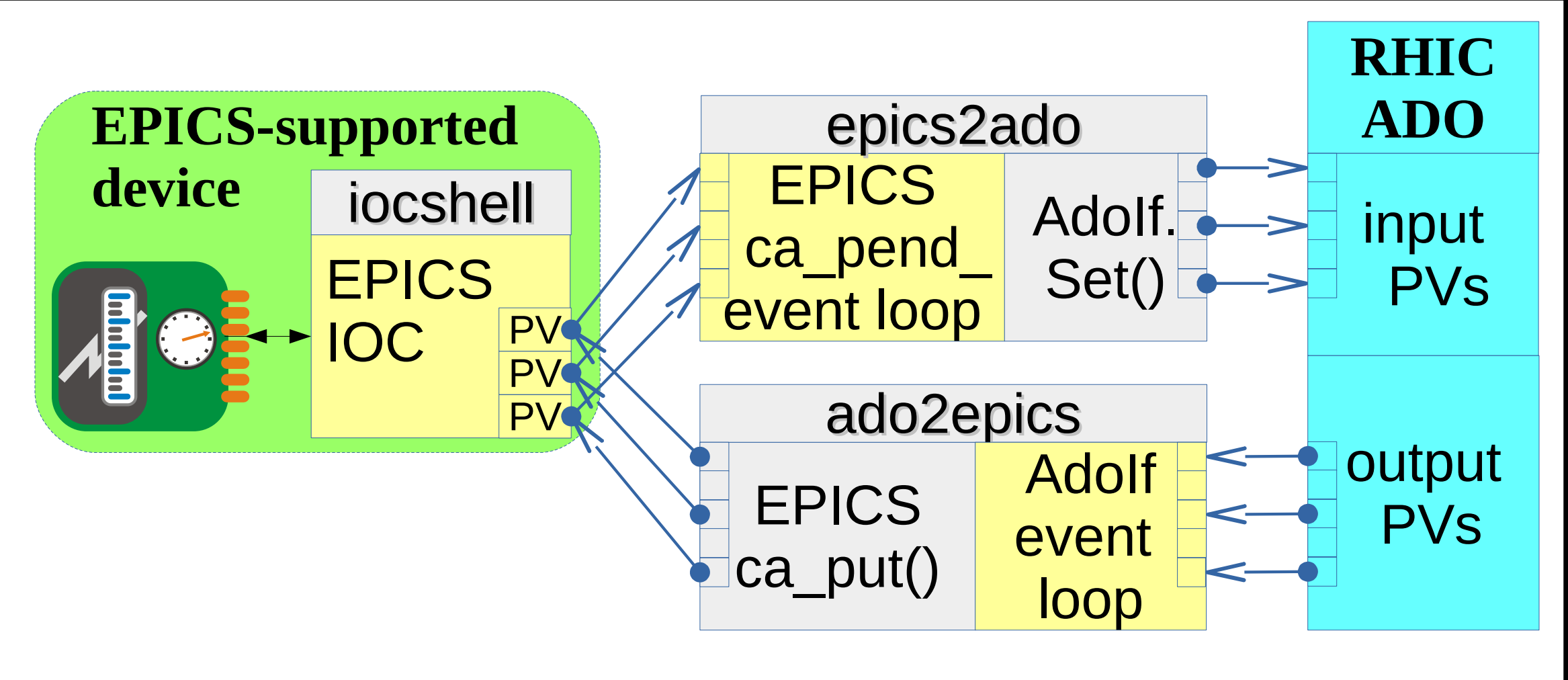
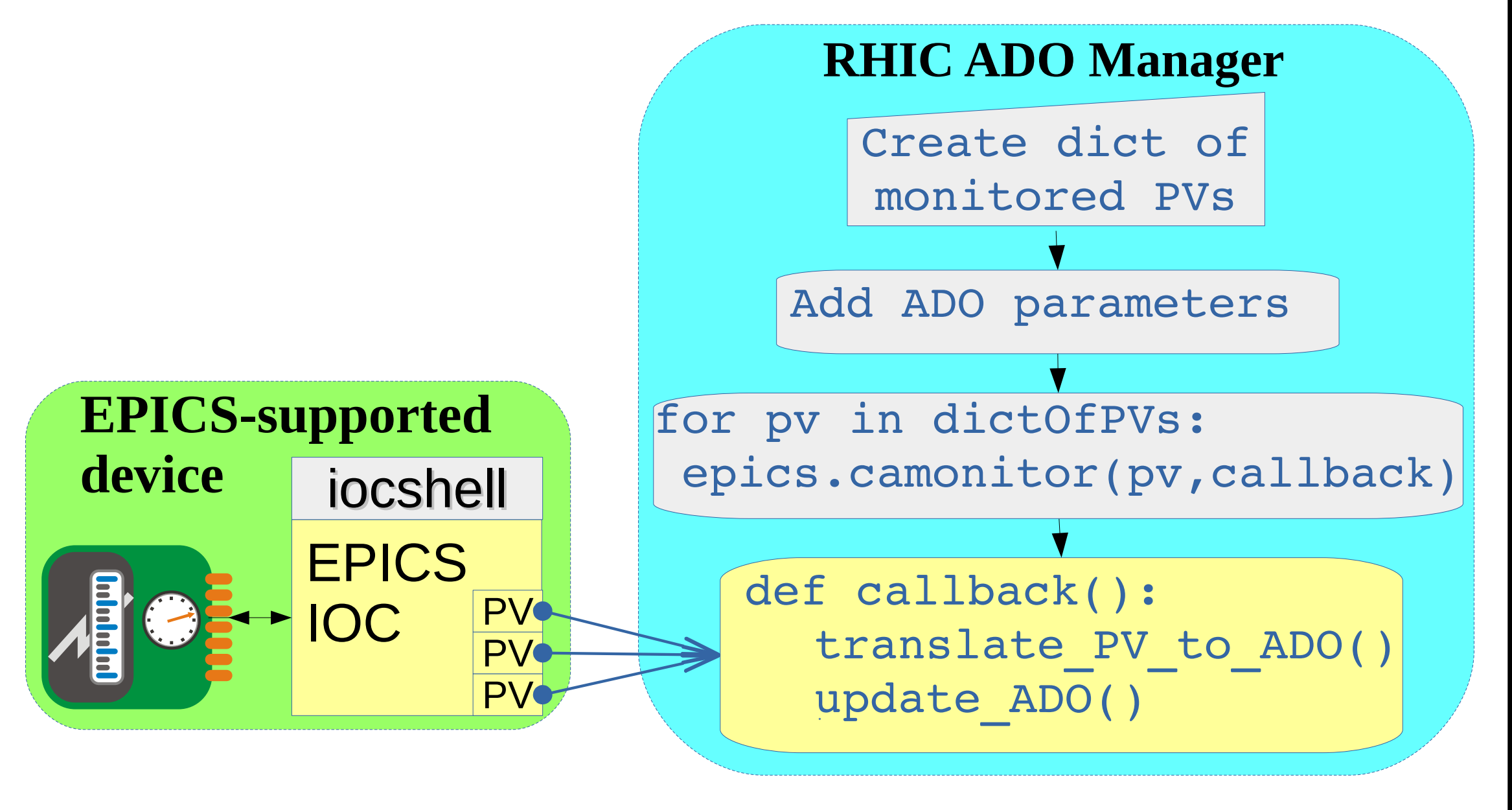


Fig 3. Two-way software bridge

Prerequisites:
Device is provided with EPICS support.
The target ADO already exists at RHIC Controls.
The host computer has all EPICS Base libraries installed.
Two processes on the host are compiled and linked with ADO and EPICS libraries:
(1) epic2ado: EPICS to ADO translator, runs ca_pend_event loop to monitor changes in all EPICS PVs, the ADO parameters are updated using adoif.Set()
(2) ado2epics: ADO to EPICS translator, runs adoIf event loop to monitor changes in ADO parameters, the EPICS PVs updated using caput().
Both programs are using the same translation table of PV names.

Disadvantages:
- **Complicated linking**
- **Number of code lines: 1200 in (1) + 1000 in (2)**

Option 2: Python-based ADO manager & pyepics



```
for pv in dictOfPVs:
 epics.camonitor(pv,callback)
```

```
def callback():
   translate_PV_to_ADO()
   update_ADO()
```

Fig 4. Python-based ADO manager & pyepics

Prerequisites:
Device is provided with EPICS support.
Device is not behind the firewall.
Python package PyEpics is installed on the host.
The EPICS shared library libca.so exists on the host.

Advantages:
- **Only libca.so is needed.**
- **Number of source code lines: 300**
- **PV access time: ~ 1ms**

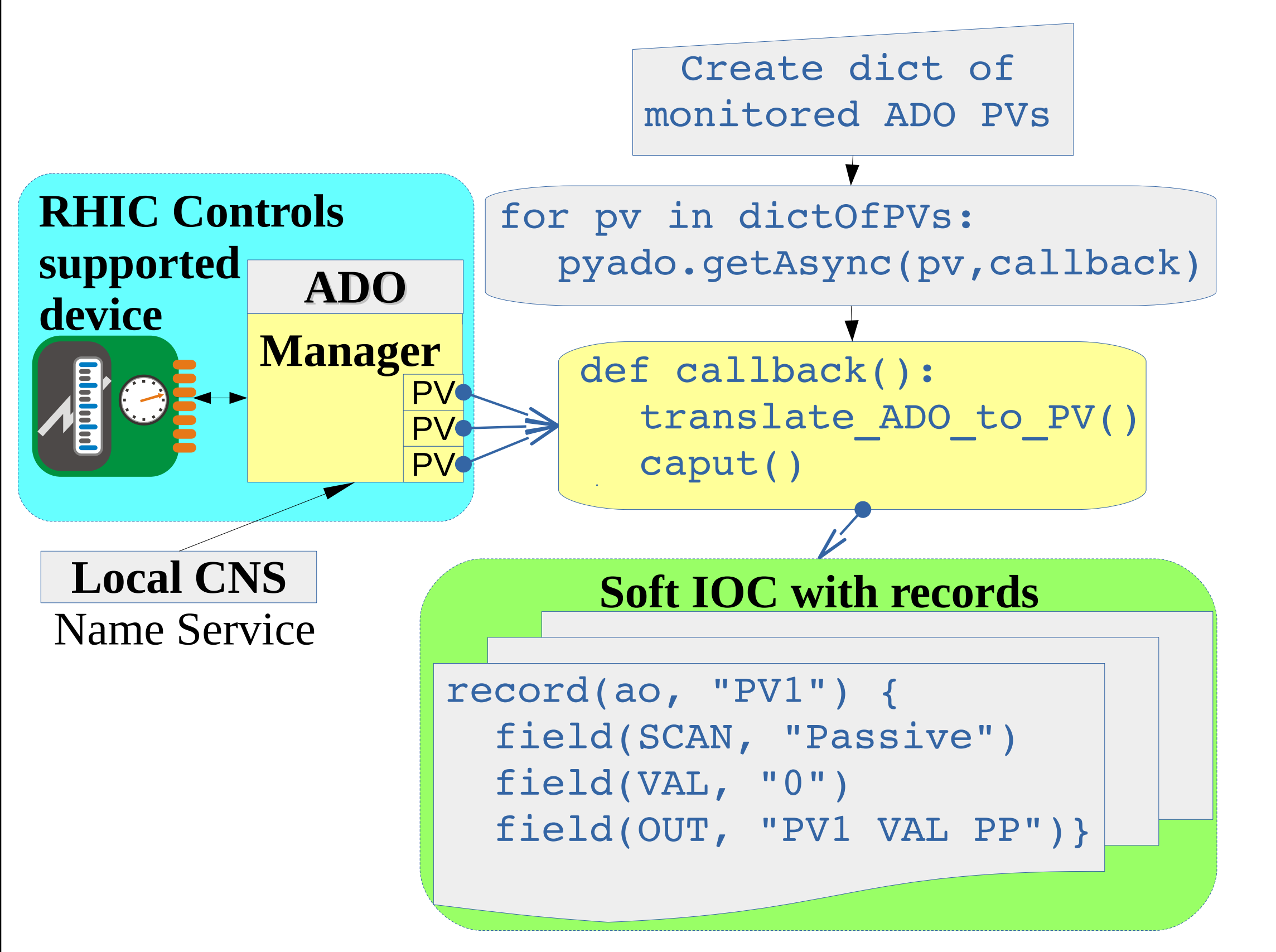
Option 3: Python-based ADO manager ssh access to EPICS PVs

Prerequisites:
Same as option 2 but using subprocess.Popen() to execute camonitor and caget program on the server.

Advantages:
- **Device can be behind the firewall.**
- **No EPICS components installed on the host.**
- **Number of source code lines: 300**

Disadvantage:
- **PV access time: ~ 400 ms**

ADO-managed Device in EPICS Environment


Create dict of monitored ADO PVs
RHIC Controls supported device
ADO
Manager
PV
PV
PV
Local CNS
Name Service
Soft IOC with records

```
for pv in dictOfPVs:
   pyado.getAsync(pv,callback)
```

```
def callback():
   translate_ADO_to_PV()
   caput()
```

```
record(ao, "PV1") {
  field(SCAN, "Passive")
  field(VAL, "0")
  field(OUT, "PV1 VAL PP")}
```

Fig 5. Control of an ADO-managed device in EPICS environment

Prerequisites:
Device is provided with the python-based ADO manager.
Python package pyado is installed on the host.
The cns.py module is modified to use a local table of name to RPC-ID translation.

Methods:
The soft IOC program is using pyado.getAsync(), which is the ADO equivalent of epics.camonitor().
The soft IOC is hosting database of records. The 'ai' records are handled in the callback function of the pyado.getAsync() and they are set using caput(), The 'ao' records are handled in the callback of the epics.camonitor() and they are passed to pyepics.set().

SUMMARY

If a device, provided with the EPICS support needs to be used in the RHIC Control environment then the most optimal solution is Option 2: Python-based ADO manager, using camonitor method from pyepics Python library.

This approach was implemented to control a 180 degree bend magnet for Low Energy RHIC electron Cooling Project[1], controlled by an EPICS-based NSLS-II power supply control system[2].

The inverted task, when the device is delivered with the RHIC Control support, needs to be used in EPICS environment, that solution is described on fig.5.

REFERENCES

1)http://slideplayer.com/slide/6537483/
2)http://www.c-ad.bnl.gov/esfd/LE_RHICeCooling_Project/LEReC.htm#Reviews

ACKNOWLEDGMENTS

Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.