Channel Finder

A Directory Service

Kunal Shroff, Kay Kasemir, Tynan Ford, Michael Davidsaver, Georg Weiss, Ralph Lange and many other

- A flat name space restricts seriously:
  - Clients need to know all channel names beforehand,
  - Portable generic clients must be simple,
  - Apps need full configuration or framework supplied service,
- Develop a Directory Service
  - Generic
    - No dependency on installation and local conventions;
  - Simple and fast (enough)
    - Use standards wherever possible;
  - Provides "query-by-functionality".

Channel Definition Example


```
    "name" : "SR:C02-MG:G04A{HFCor:FM1}Fld-I",
   "owner" : "train",
"properties": [
                "handle":"Setpoint",
                "elemType":"HFCOR",
                "devName":"FM1G4C02A",
                "iocName":"motorsim",
                "time":"2016-03-21"
              ]
    "tags": ["x", "sys.SR"]
```

Example Data Set

Device FM1G4C02A

Channel Name	SR:C02-MG:G04A{HFCor:FM1}		SR:C02-MG:G04A{VFCor:FM1}	
	Fld-I	Fld-SP	Fld-I	Fld-SP
handle	READBACK	SETPOINT	READBACK	SETPOINT
elemName	FXM1G4C02A		FYM1G4C02A	
elemType	HFCOR		VFCOR	
elemField	x		y	
devName	FM1G4C02A			
sEnd	65.5222			
cell	C02			
girder	G4			
symmetry	A			
length	0.44			
ordinal	263		264	
tags	eget	eput	eget	eput
	x		y	
	sys.SR			

Query for PV’s

SR:C02*&elemType=HFCOR*&handle=setpoint&~tag=sys.SR

- All PVs from with names starting with **SR:C02**
- Represent the setpoint of the horizontal corrector
- Part of the storage ring

Example*

corrector = Corrector(deviceName=`FM1G4C02A`)

>> channels = cf.find(deviceName='FM1G4CO2A')

xPos, yPos = corrector.getPositions()

*>> xPosPv = cf.find([(deviceName,'FM1G4CO2A'), (*elemField, 'x'*)])*

*>> yPosPv = cf.find([(deviceName,'FM1G4CO2A'), (*elemField, 'y'*)])*

>> caget(xPosPv) and caget(yPosPv)

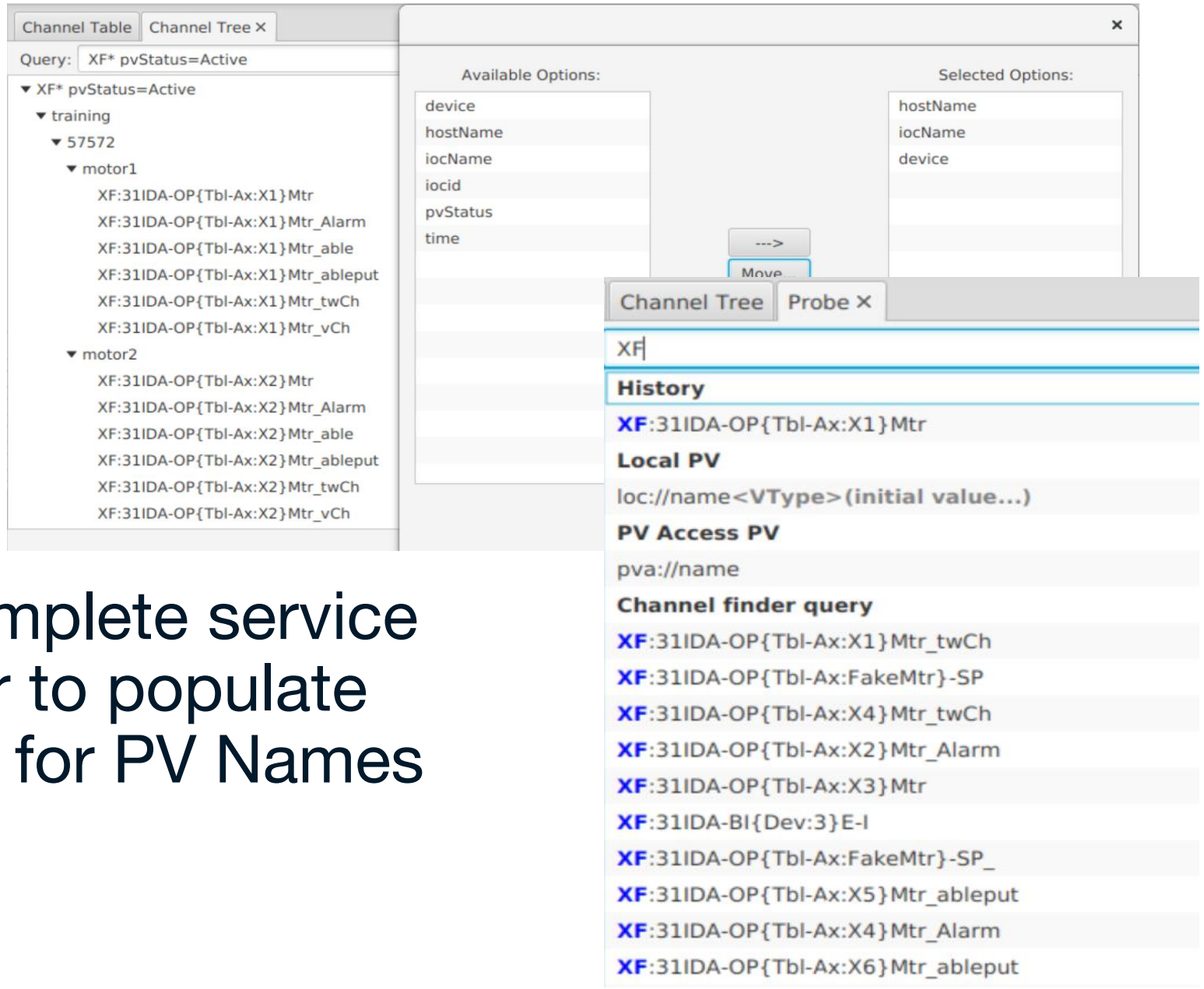
corrector.setPositions(xPos=1.23, yPos=3.21)

>> caput('SR:C02-MG:G04A{HFCor:FM1} Fld-SP', 1.23)

>> caput('SR:C02-MG:G04A{VFCor:FM1} Fld-SP', 3.21)

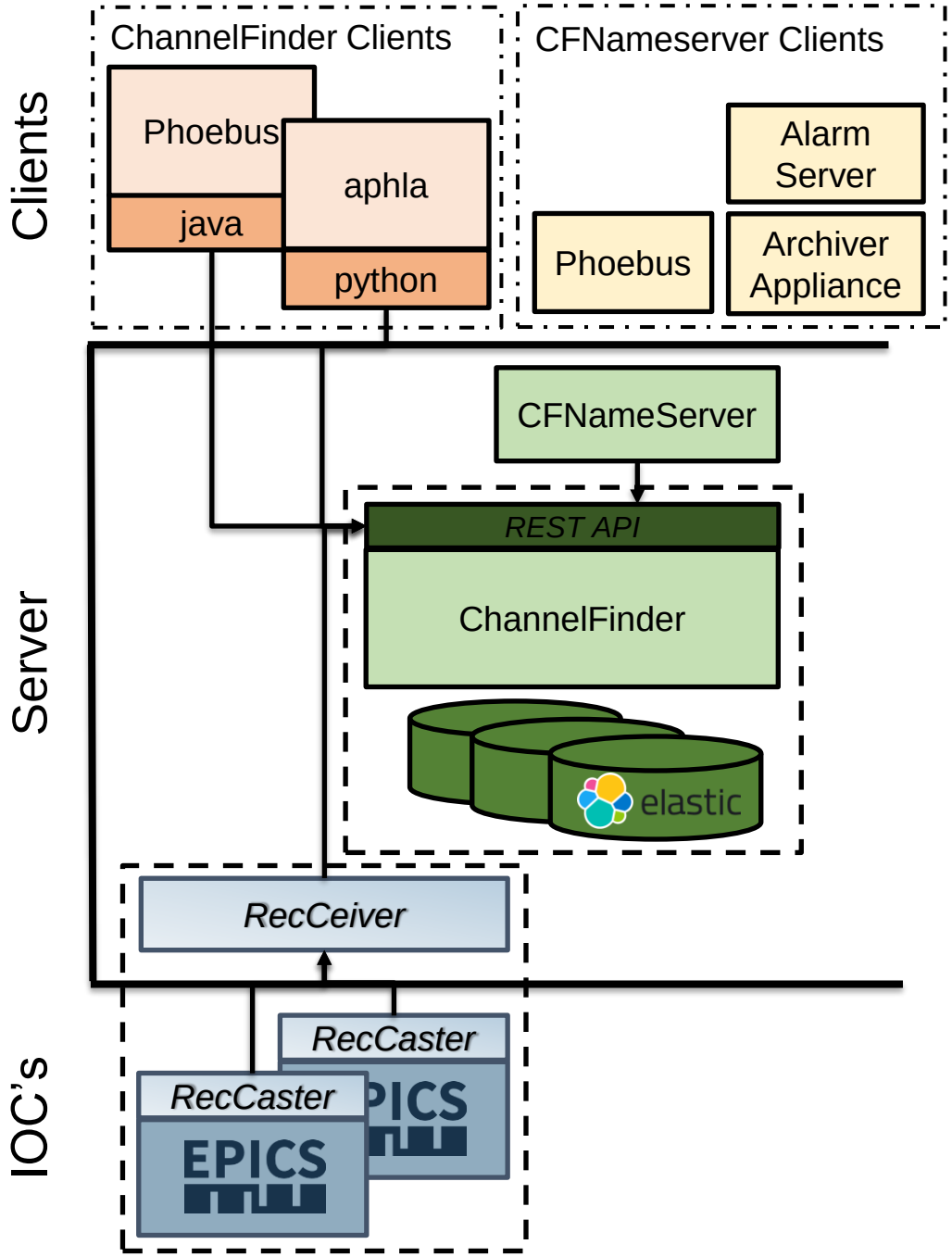
ChannelFinder Applications and Use Cases

- Channel Tree
  Create hierarchical views of EPICS PVs



- The phoebus autocomplete service queries channelfinder to populate the autocomplete list for PV Names

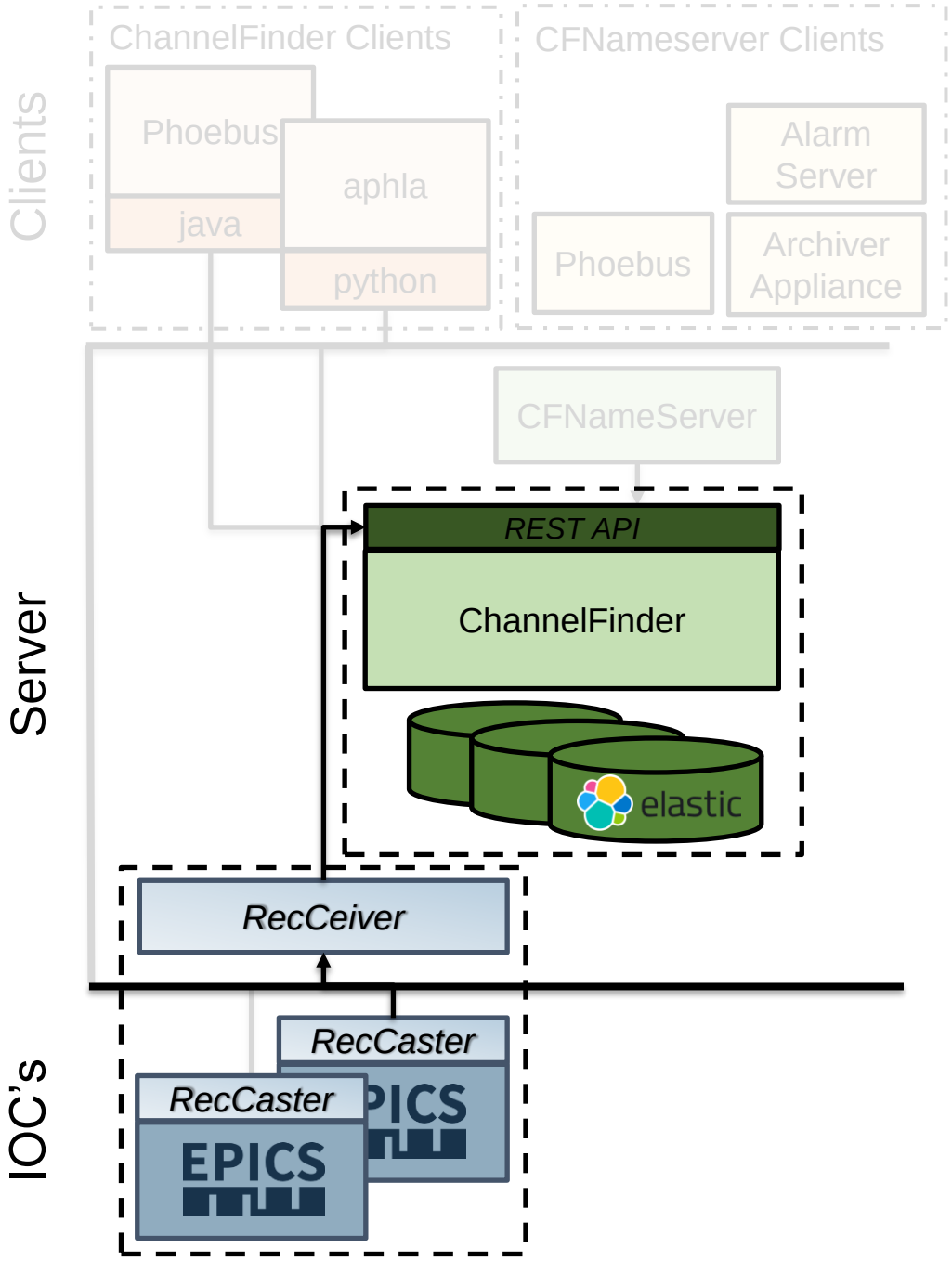
Populating Channel Finder *with* Rescyn



Managing ChannelFinder data

recSync

- How to populate new channels
  - Without having to learn the client api's
- How to manage existing channels
  - Orphaned channels
  - Moved channels



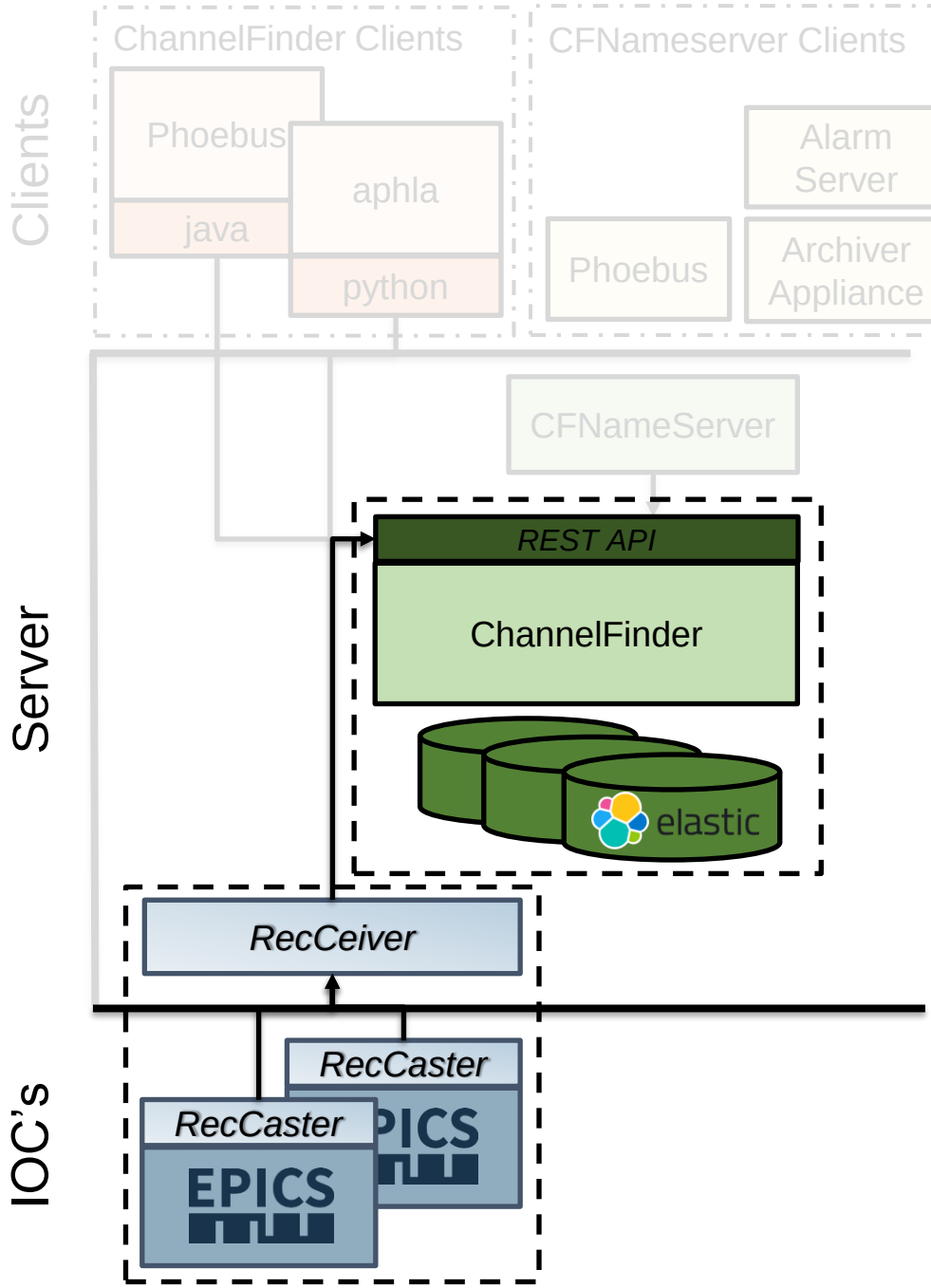
Managing ChannelFinder data

recSync

- The record synchronizer
  - A client (RecCaster) which running as part of an EPICS IOC (EPICS support module)
  - A server (RecCeiver) which is a stand alone daemon.
- Ensure ChannelFinder have a complete list of all records currently provided by the client IOCs.
- Recsync Information
  - The EPICS Base Version
  - A predefined set of environment variables
  - The name and type of all records
  - Any info() tags associated with these records
  - A set of user defined environment variables
- *https://github.com/ChannelFinder/recsync*

CFNameServer

Populating ChannelFinder with Connection data



CFNameServer

https://github.com/epics-base/epics-base/pull/269

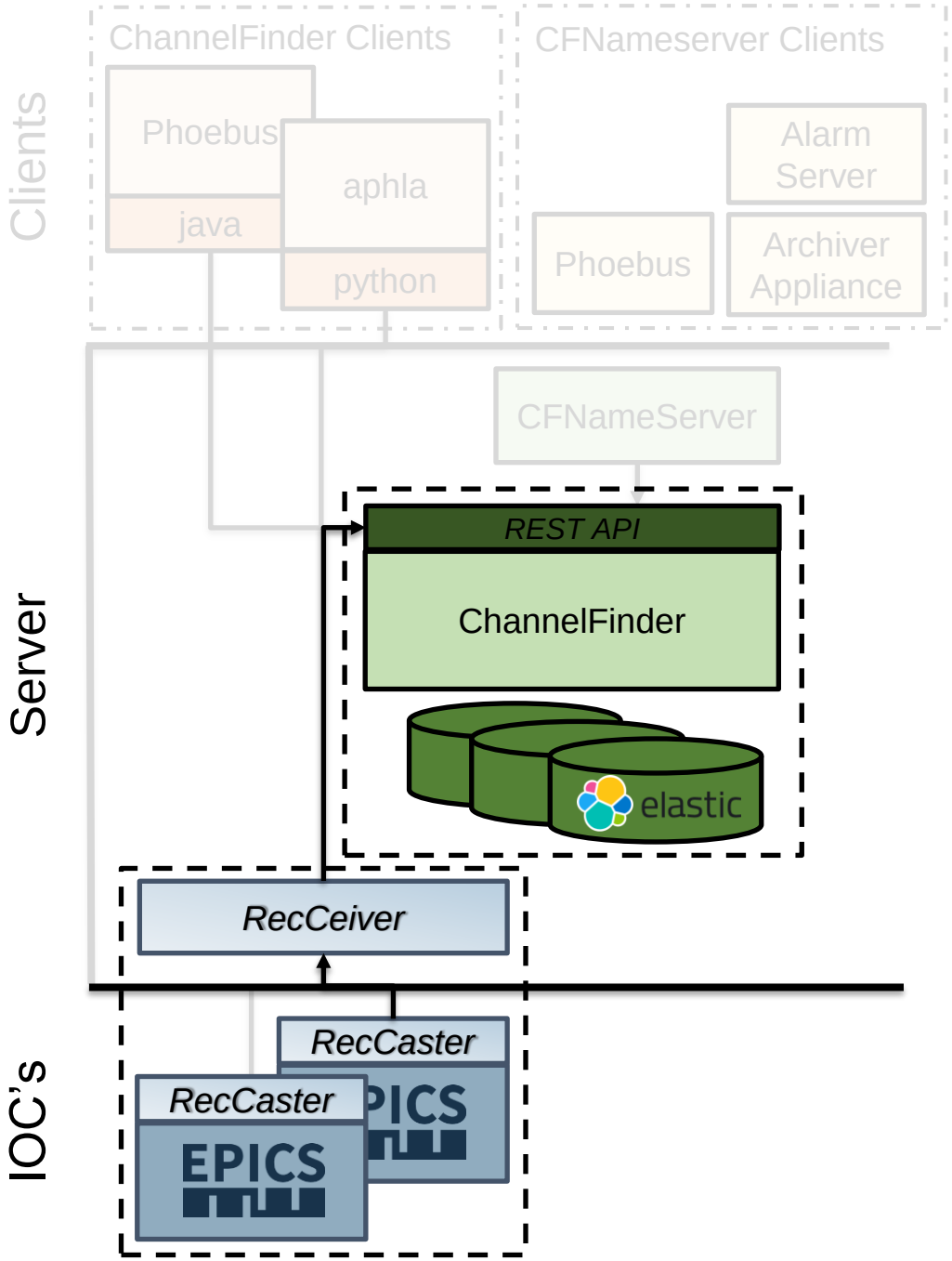
- EPICS 7.0.7 the TCP port used by RSRV was exposed as an environment variable

  `RSRV_SERVER_PORT`

- RecCaster provides both host IP & RSRV port

- RecCeiver populates ChannelFinder with the PV name and properties containing the IP and port info

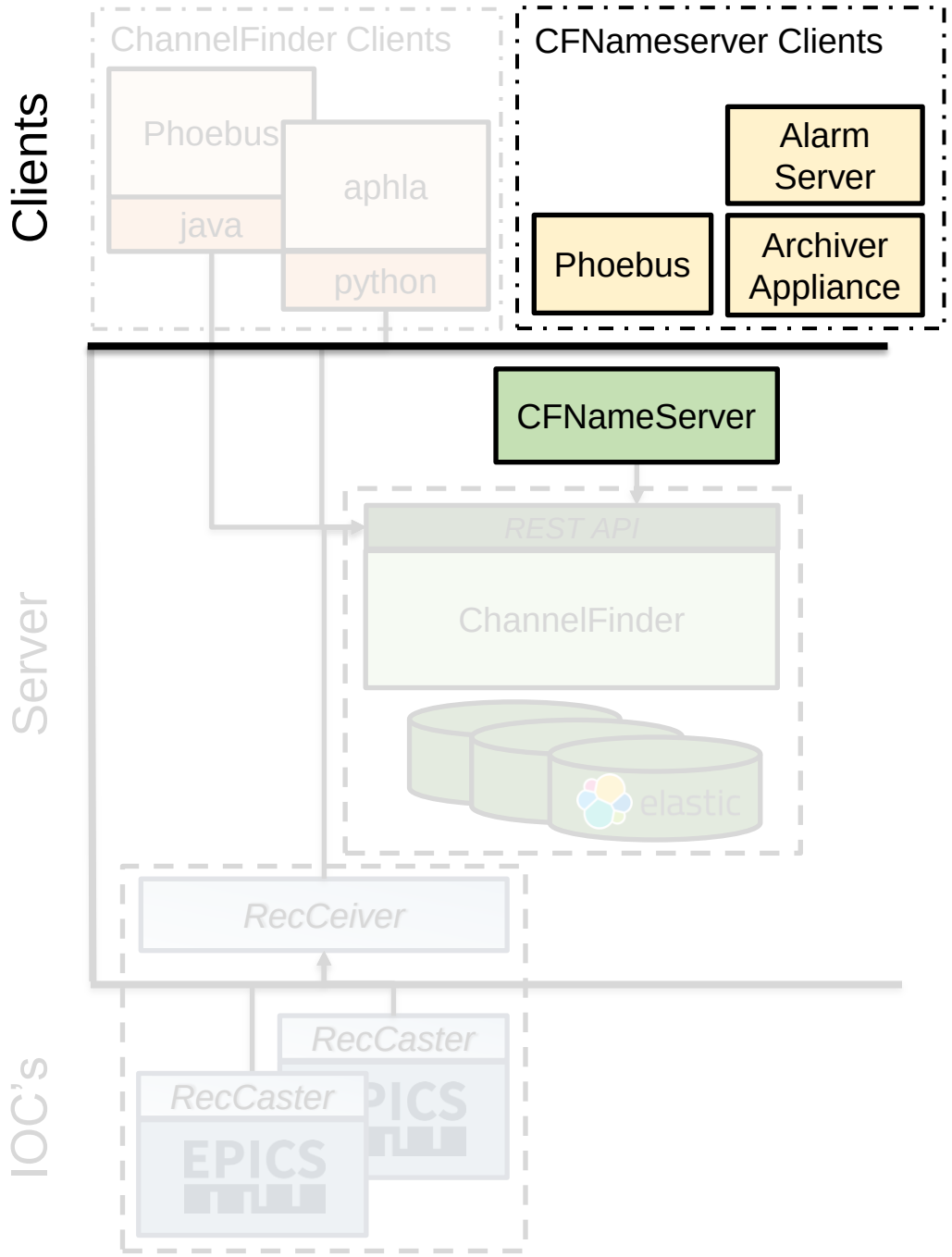
Populating ChannelFinder with Connection data



CFNameServer

Example data populated by the receiver in ChannelFinder

```
{
    "name": "tst:counter",
    "properties": [
        {
            "name": "socket_address",
           "value": "130.199.219.181:5078"
        }
    ]
}
```



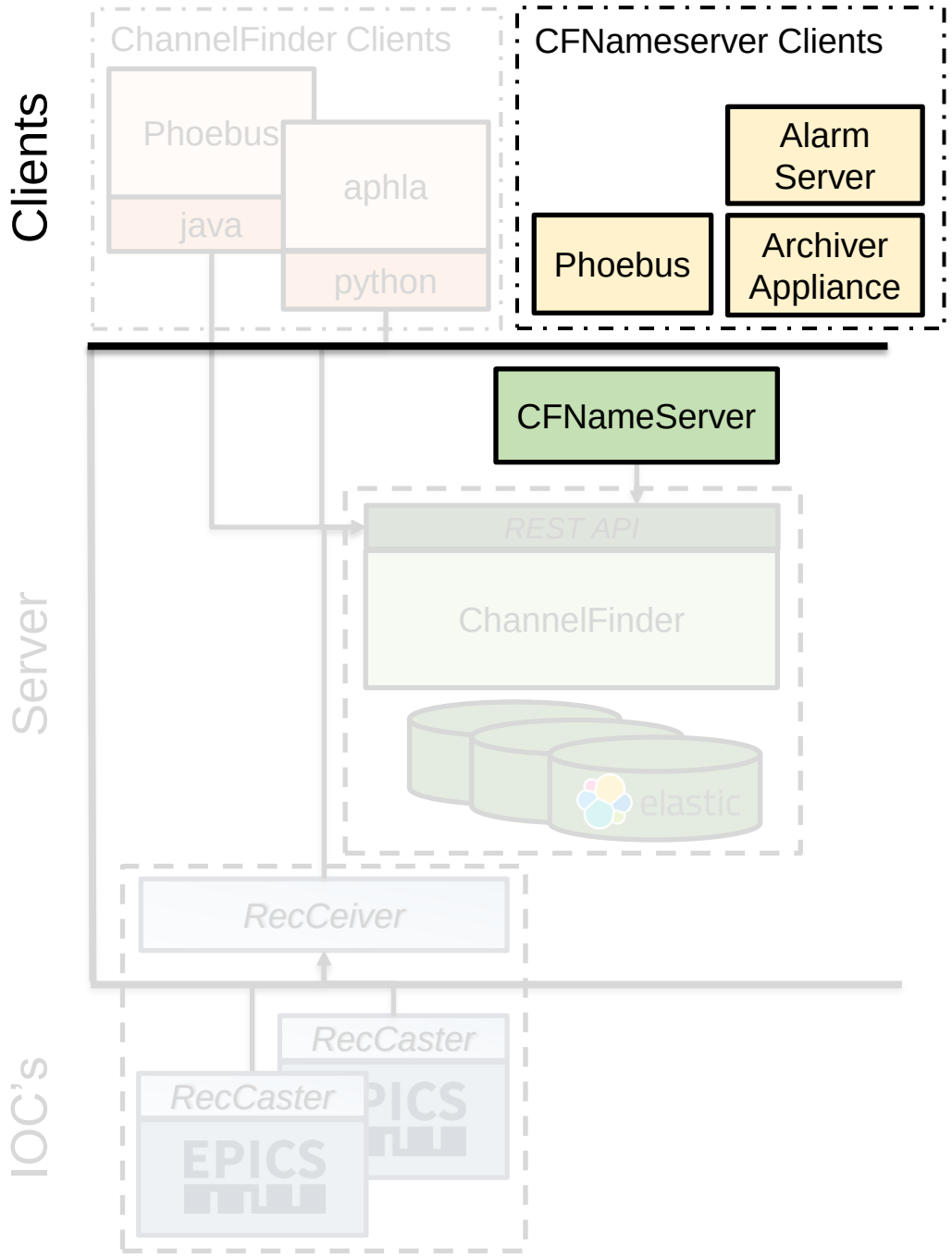
Name Resolution using ChannelFinder data

CFNameServer

- A PVAccess Nameserver using ChannelFinder service
  - Upon receipt of a name resolution request, CFNameServer queries the ChannelFinder service for information about the host IP and port associated with the PV

    GET http://ChannelFinder/../channels/tst:cnt

  - Using the "socket_address" property value, CFNameserver responds to the PV name resolution requests with information about the IOC IP port



Client to CFNameServer

CFNameServer

- core-pva
- pvxs

Client using the above libraries can be configure via an environment variable to use the CFNameServer

EPICS_PVA_NAME_SERVERS=CFNameServer.nsls2.bnl.gov:5076

pvget/pvmonitor/pvput tst:cnt

Channel Processors

control system studio

- Whenever a new Channel is created or an existing Channel is modified, CFProcessors allow sites to configure a set of actions to be performed

  Examples:
  - *Parsing the PV name to populate additional properties like device name*
  - *Configure other middle layer services like adding new PV to the archiver*

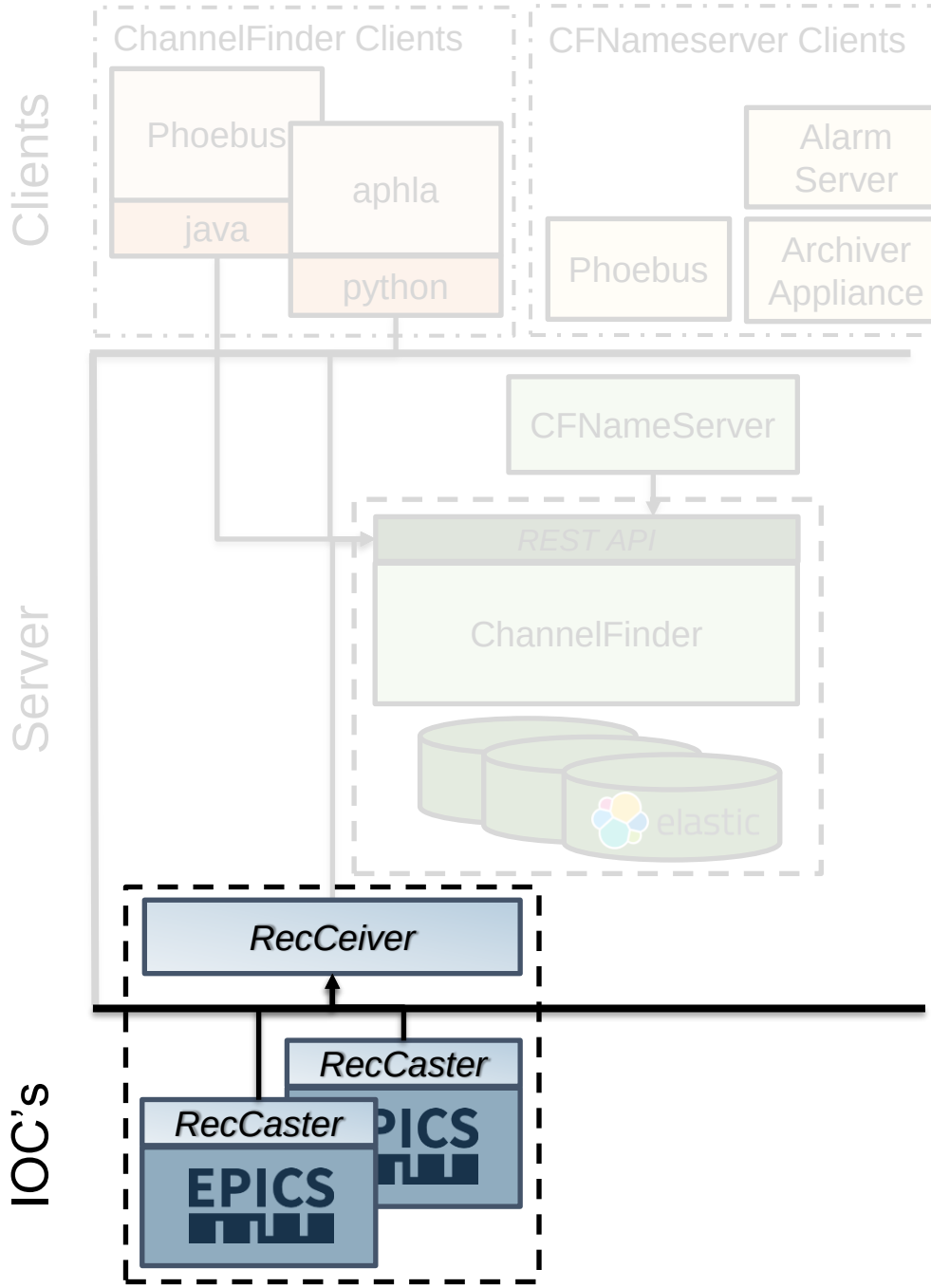
Implementing a ChannelProcessor

```
public interface ChannelProcessor
{
        void process(List<XmlChannel> channels);
}
```

- A processor can be added to ChannelFinder by implementing the "***ChannelProcessor***" interface

- Implementation of the processor are discovered and loaded using the Service Provider Interface (SPI)
- Each instance of ChannelFinder can be configured with 0-n different processors
- The processors are run asynchronously on a dedicated thread pool

- AAChannelProcessor
  - The AA processor, checks if the newly created channel includes a property called ***archive***, if present it uses the value of this property and adds this channel to the archiver appliance to begin archiving.



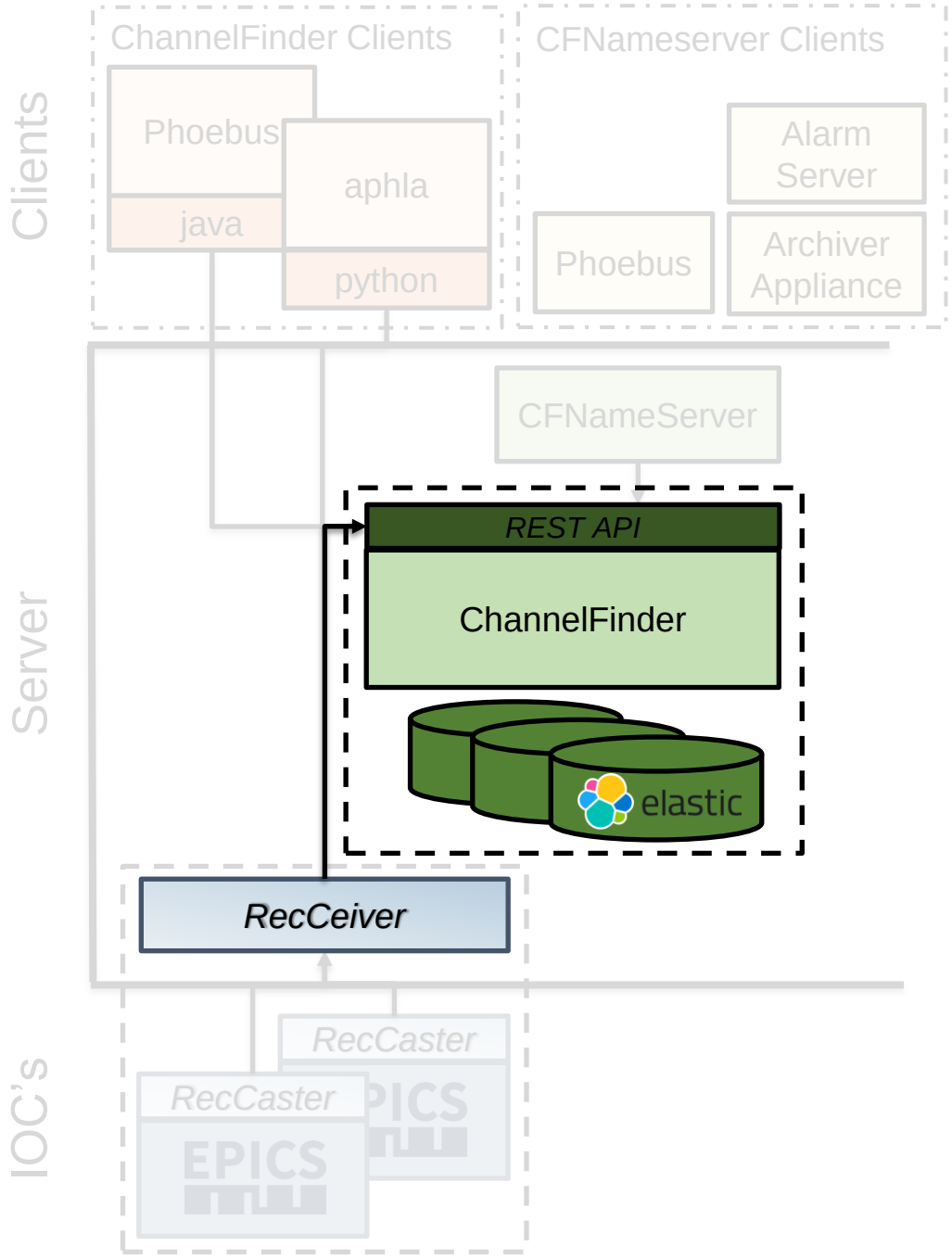
AAChannelProcessor

The IOC developer adds an info tag to records they wish to be added to the Archiver

example.db

```
record(longin, "tst:counter") {
    field(VAL, "0")
    …
    info("archive","MONITOR@0.1")
}
```

AAChannelProcessor

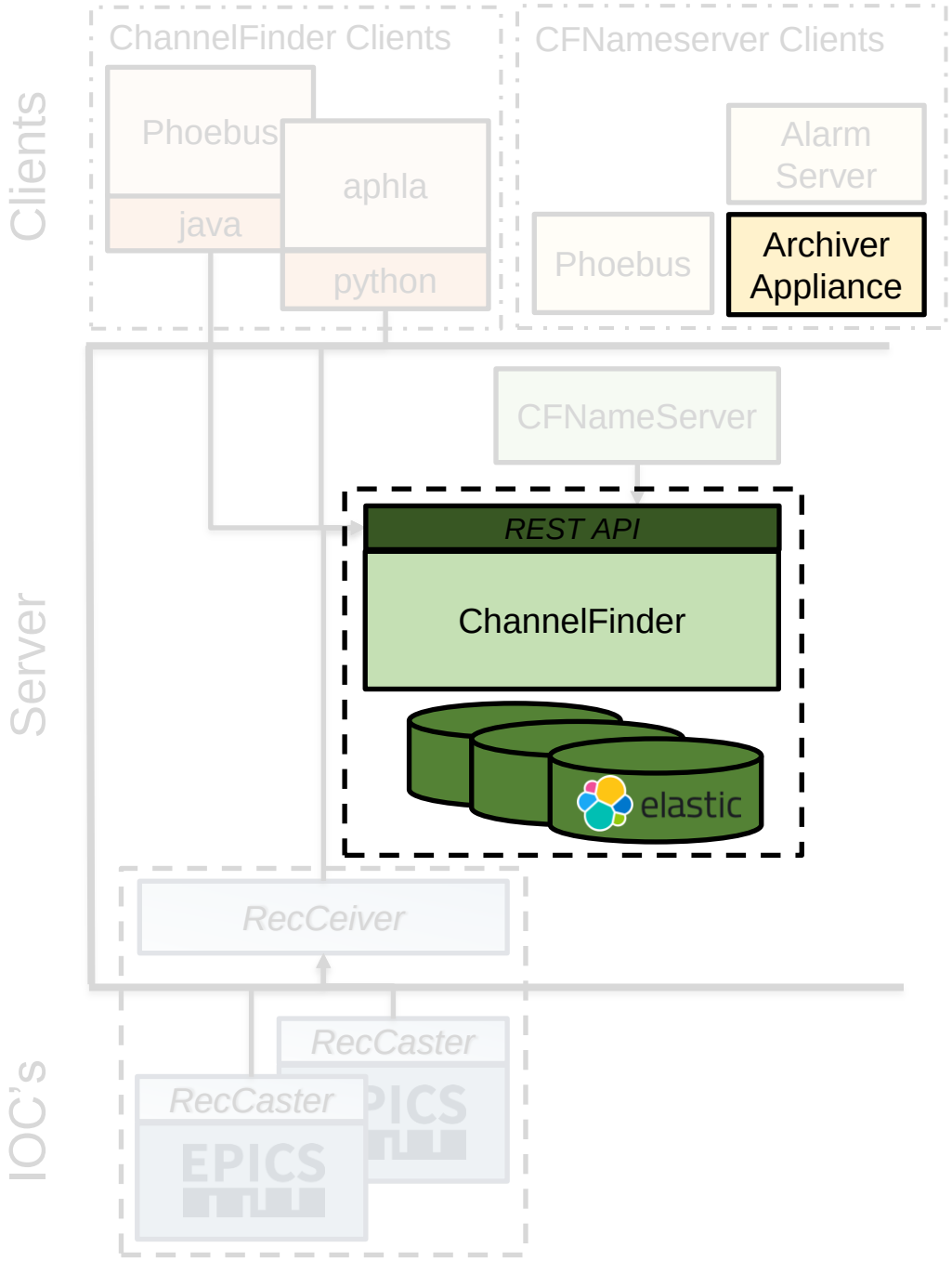


The RecCeiver converts the ***archive*** info tag into a property

GET http://Channelfinder/.../tst:counter

```
{
    "name": "tst:counter",
    "properties": [
        {
          "name": "archive",
           "value": " MONITOR@0.1 "
        }
    ]
}
```

AAChannelProcessor




```
public interface AAChannelProcessor
{
    void process(List<XmlChannel> channels) {

      #check if the channels have *archive* property

      # make a http request to the

     http://archiver.bnl.gov/mgmt/bpl/archivePV?
                pv=tst:counter&
                samplingperiod=.1&
                samplingmethod=MONITOR

    }
}
```

- Releases
  - Version 1.x & 2.x Ironing out the REST API
  - Version 3.x Migration to elastic scalability and performance
  - Version 4.x Simplify build and deployment with switch to springboot, code cleanup, extensible and pluggable architecture
- Next Releases
  - Standardize some of the common properties like ***archive*** & ***socket_address***
  - Improvements to RecCeiver
  - Stress tests and scalability tests for CFNameserver
  - New Phoebus applications and tools