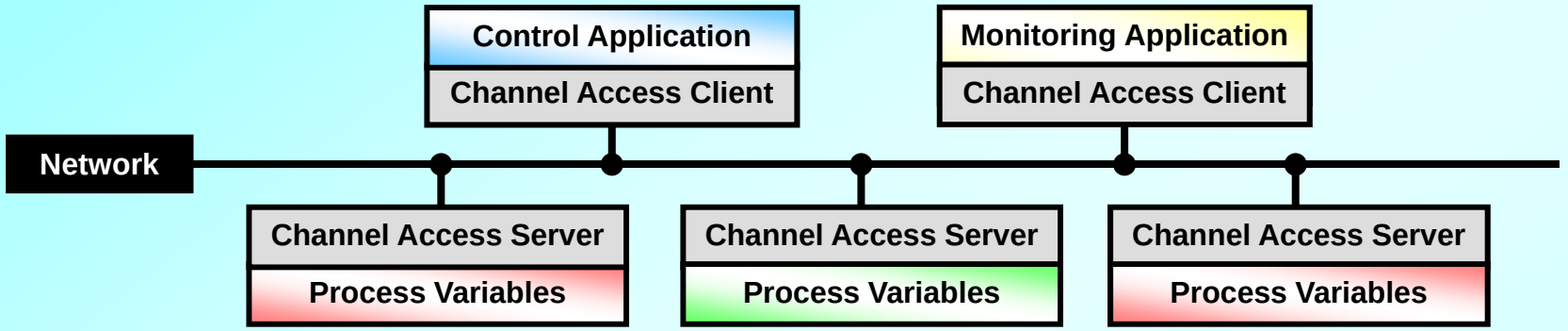
# Tracker Slow Control & Monitoring

Imperial College London

# Introduction

- Overview of EPICS
- Channel Access servers
- AFEIIt software
- AFEIIt Channel Access server
- AFEIIt Channel Access clients
- Current status
- Summary

# EPICS Overview

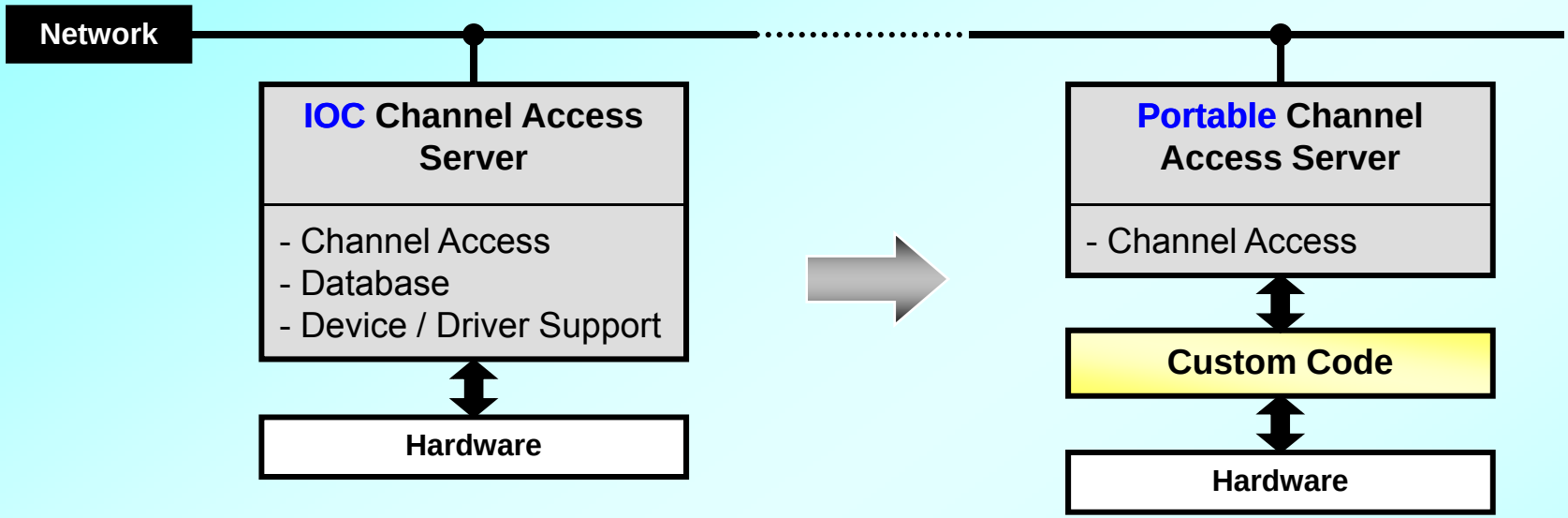



- Each control / status parameter represented by a 'Process Variable' (PV)
  - PV: Named piece of data (e.g. temperature) with a set of attributes (e.g. safe operating limits)
- Channel Access (CA) servers provide access to PVs
- CA clients read / write PVs to perform control & monitoring tasks
- Network-based, distributed system
  - PVs can be spread over multiple servers, accessed transparently over network

# Channel Access Servers

- Typically, an EPICS CA server consists of an Input Output Controller (IOC)
  - An IOC loads one or more databases of PV records, which define system configuration & hardware connections
- Fine for many applications, **but**:
  - Structure of driver support in IOC framework best suited for relatively simple hardware access code
  - Need to control / monitor AFEIIts – not simple
  - Definition of hardware configuration in IOCs not well established
- AFEIIt code should be well-structured, well-abstracted, modular & scaleable
  - 'IOC driver support' not really appropriate
- Fortunately, there is an alternative

# Portable Channel Access Server




- EPICS provides an additional *Portable Channel Access Server* C++ library
- Contains all the CA network protocol & PV framework, without EPICS database / IO routines
- Can wrap arbitrary hardware access code to produce fully EPICS-compliant ‘custom’ CA servers

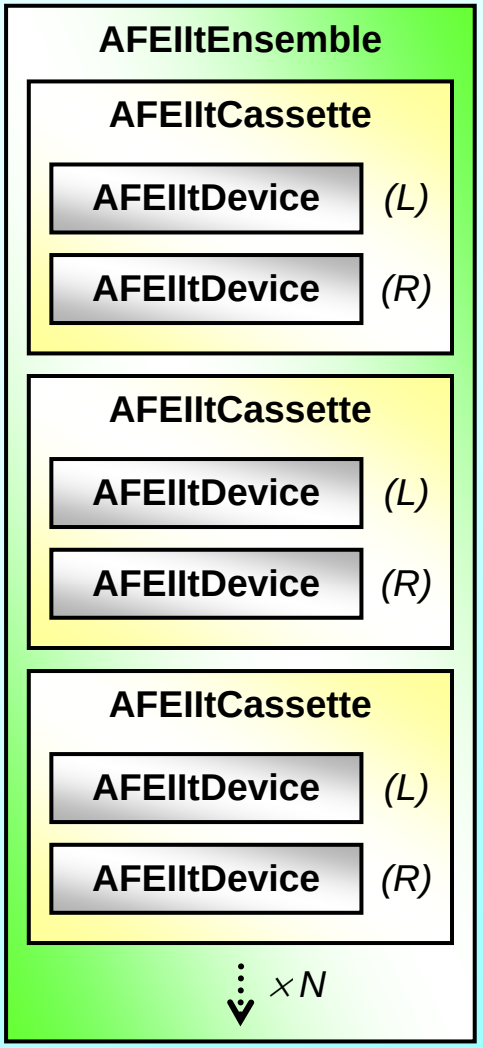
# A New Portable CA Server Interface

- Portable CA server API said to be simple – not *entirely* true…
  - User has to implement & inherit from a number of classes
  - A significant number of subtleties (annoyances?)
- Consequently, a new wrapper framework has been written:
  - Hides (almost) all EPICS complexities
  - Greatly simplifies creation of new servers
- User essentially just has to inherit from 2 classes and implement 4 virtual functions – effectively no EPICS knowledge required

# AFEIIt Software

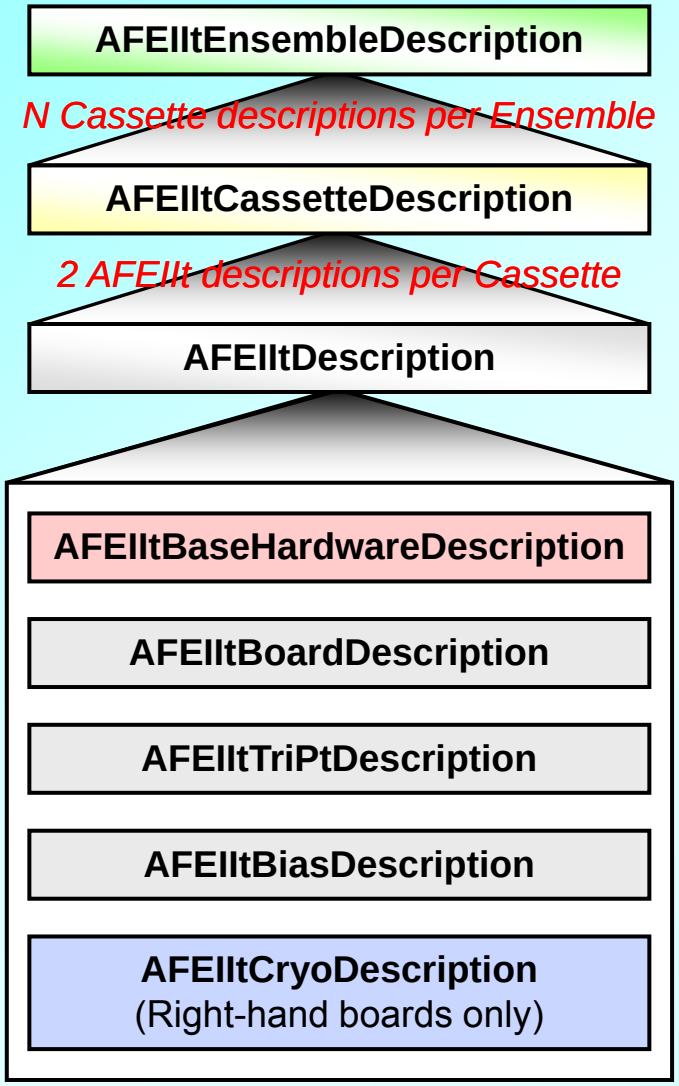
**MICEBaseHardwareInterfaces *class list:***
- SbsVmeBusAdaptor
- VmeAddressTable
- VmeBusAdaptor
- VmeDevice

**MICETrackerHardwareInterfaces *class list:***
- AFEIItBaseHardwareDescription
- AFEIItBiasDescription
- AFEIItBoardDescription
- AFEIItCassette
- AFEIItCassetteDescription
- AFEIItCassetteDescriptionXmlReader
- AFEIItCassetteDescriptionXmlWriter
- AFEIItCore
- AFEIItCryoDescription
- AFEIItDefines
- AFEIItDescription
- AFEIItDescriptionXmlReader
- AFEIItDescriptionXmlWriter
- AFEIItDevice
- AFEIItEnsemble
- AFEIItEnsembleDescription
- AFEIItEnsembleDescriptionXmlReader
- AFEIItEnsembleDescriptionXmlWriter
- AFEIItInterface
- AFEIItStringConverter
- AFEIItTriPtDescription
- MIL1553Interface



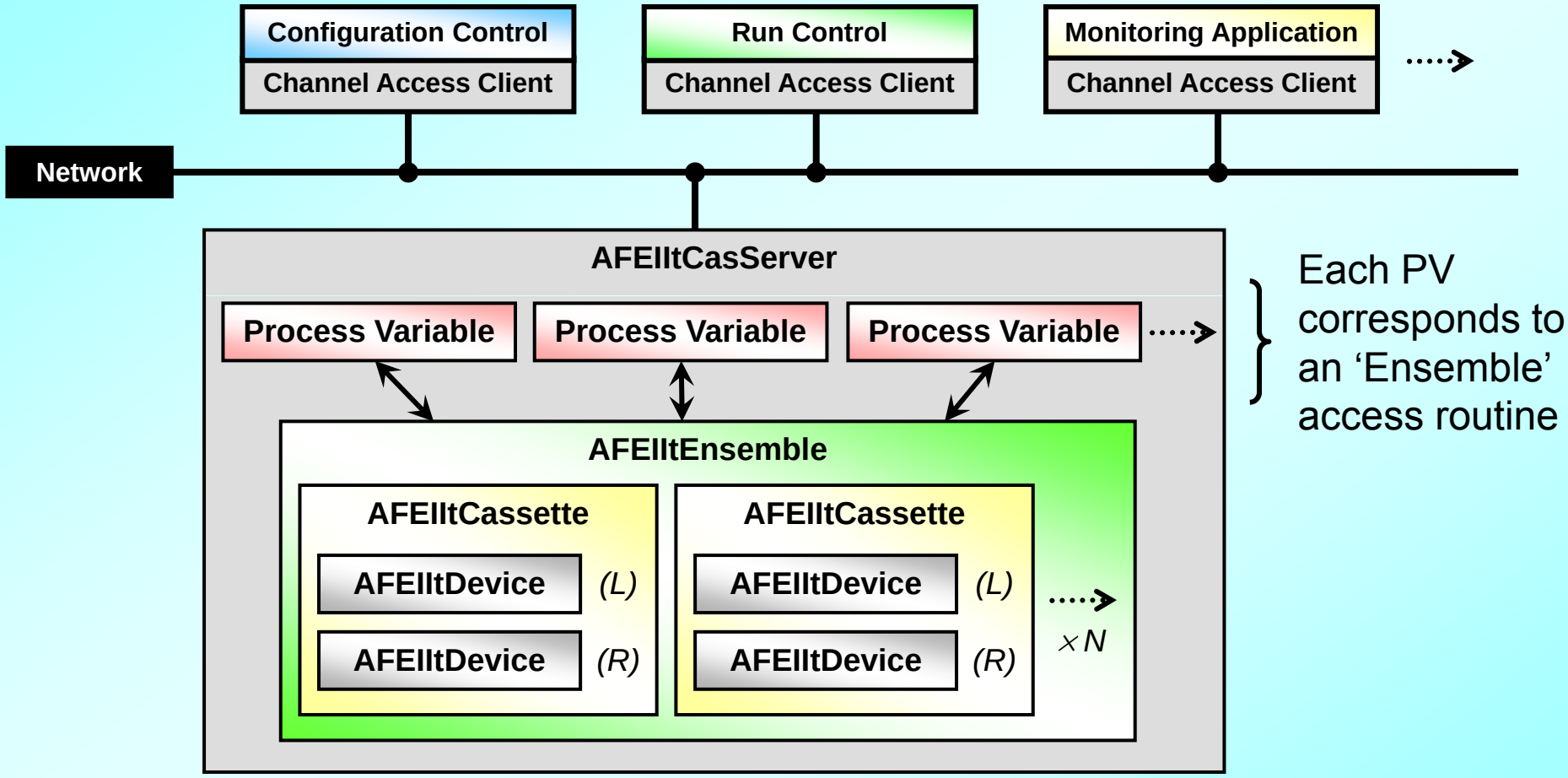

- AFEIIt libraries provide structured interface at board & cassette levels
- An ‘Ensemble’ contains an arbitrary number of cassettes, enabling control of entire system via a single object

# AFEIIt Configuration




- Multi-tiered hierarchy of 'description' objects
- Divides system configuration into logical units
  - ~230 parameters per board → organisation important
- Description files stored in XML format
- Only 'relevant' board descriptions can be modified while software running
  - Base hardware settings, VME information, numbers of cassettes etc. are protected
  - i.e. Clients cannot 'break' representation of physical hardware

# AFEIIt CA Server




- Boards may be split between multiple servers, running on different PCs

# AFEIIt Process Variables

Cassette Index (1→N)

**Cassette Level**

| | |
|---|---|
| MICE-TK-CAS-01:CONFIG-ALL | (R/W, restricted access) |
| MICE-TK-CAS-01:CONFIG-BOARD | (R/W, restricted access) |
| MICE-TK-CAS-01:CONFIG-TRIPT | (R/W, restricted access) |
| MICE-TK-CAS-01:CONFIG-BIAS | (R/W, restricted access) |
| MICE-TK-CAS-01:CONFIG-CRYO | (R/W, restricted access) |
| MICE-TK-CAS-01:TRIG-ENABLE | (R/W, restricted access) |
| MICE-TK-CAS-01:TEMP | (R, unrestricted access) |
| MICE-TK-CAS-01:HEATER | (R, unrestricted access) |
| MICE-TK-CAS-01:CRYO-DESC | (R/W, restricted access) |
| MICE-TK-CAS-01:STATUS-READ-ENABLE | (R/W, restricted access) |
| MICE-TK-CAS-01:STATUS-READ-ENABLED | (R, unrestricted access) |

**Board Level**

| | |
|---|---|
| MICE-TK-CAS-01:AFE-01:CONFIG-ALL | (R/W, restricted access) |
| MICE-TK-CAS-01:AFE-01:CONFIG-BOARD | (R/W, restricted access) |
| MICE-TK-CAS-01:AFE-01:CONFIG-TRIPTS | (R/W, restricted access) |
| MICE-TK-CAS-01:AFE-01:CONFIG-BIAS | (R/W, restricted access) |
| MICE-TK-CAS-01:AFE-01:AFE-DESC | (R/W, restricted access) |
| MICE-TK-CAS-01:AFE-01:BOARD-DESC | (R/W, restricted access) |
| MICE-TK-CAS-01:AFE-01:TRIPT-DESC | (R/W, restricted access) |
| MICE-TK-CAS-01:AFE-01:BIAS-DESC | (R/W, restricted access) |

Board Index (1→2)

(NB: Can add more, if required)

# AFEIIt PV 'Types'

| PV Type(s) | PV Function | Access* |
|---|---|---|
| - AFE-DESC<br>- BOARD-DESC<br>- TRIPT-DESC<br>- BIAS-DESC<br>- CRYO-DESC | XML strings containing cassette / board / subsystem description parameters | Configuration Controller |
| - CONFIG-ALL<br>- CONFIG-BOARD<br>- CONFIG-TRIPT<br>- CONFIG-BIAS<br>- CONFIG-CRYO | Configure hardware with current cassette / board / subsystem description settings | Configuration Controller |
| - TRIG-ENABLE | Enable / disable triggers | Run Controller |
| - STATUS-READ-ENABLE | Enable / disable status monitoring | Run Controller |
| - STATUS-READ-ENABLED | Current monitoring enable state | Monitor (public) |
| - TEMP<br>- HEATER | Arrays of status information (currently temperatures & heater values) | Monitor (public) |

* User access rights for each PV fully configurable – can modify as required

# Expected Mode of Operation

**Server initialised with default AFEIlt configuration files**

↓

**Configuration Client**

- Update description PVs (if required)
- Access configuration PVs to set up hardware

↓

**Run Control Client**
(Implemented within DATE?)

- Start of run: disable status readout & enable triggers
- End of run: enable status readout & disable triggers

**Monitoring Client**

- Runs periodically, whenever status readout enabled
- Read ‘temperature’ & ‘heater’ PVs
- Generate plots & alarms

# AFEIIt Client Development

- EPICS provides client library for accessing servers
  - Written in C, not particularly well suited for object orientated C++
  - A significant number of subtleties (annoyances?)
- As with CA server interface, a wrapper has been produced to hide all the EPICS complexities
  - Simplifies creation of new clients, no EPICS knowledge required
- An AFEIIt client library has been implemented, with interface identical to that of cassette / ensemble hardware access code
  - Allows full AFEIIt control without user having to consider underlying CA client / server mechanism

# Current Status

## Essentially Complete:

- CA server framework
- CA client framework
- AFEIIt hardware access code
  - But may require added (as yet unforeseen) functionality
- AFEIIt CA server
- AFEIIt CA client library

```
Just for fun (including simple beam monitor written
using the same CA client / server framework):
- Total Physical Source Lines
  of Code (SLOC)_____________________ 36,443
- Development Effort Estimate,
  Person-Years (Person-Months)_______ 8.72 (104.69)
- Schedule Estimate, Years (Months)__ 1.22 (14.64)
- Estimated Average Number of
  Developers (Effort/Schedule)_______ 7.15
- Total Estimated Cost to Develop____ $ 1,178,519
Generated using David A. Wheeler's 'SLOCCount'
```

## Remaining Tasks:

- Configuration GUIs
- AFEIIt client applications
  - Configuration
  - Run control (DATE?)
  - Monitoring
- Extensive testing
  - Very little access to ‘real’ hardware so far…
    - Setup at Imperial either in use for QA testing, or non-operational
  - Hope to schedule work around final station QA run

# Summary

- General purpose framework for generating CA servers / clients has been established
  - Initially used to write application for temporary beam monitors – framework functions correctly
- C++ library for AFEIIt hardware access has been written
  - Requires additional testing
- AFEIIt Server and client library essentially complete
- Remaining work primarily involves configuration GUIs & specific client applications (& more testing)