EPICS and choosing a framework

Steve Hunt
Alceli

Contents

- Overview of EPICS
- Development Roadmap for Framework
- Pros/Cons
- Examples
- Comments

Overview of EPICS

- A framework for building distributed control
- Publish – subscribe architecture
- Location transparency (get data by name)
- Version transparency
- Optional Security
- State Machines
- Alarms of every object
- Timing of every object
- Archiving
- APIs for everything

Development Roadmap

- Mature Technology – limited new features needed
- Latest version (3.15+V4) includes complex data types, and zero copy of arrays
- Collaborative development of core features + anyone can add a component – if it is good it will be used. No support costs or commitment needed.

Pros/Cons

- PRO- Very stable
- PRO – Very Reliable
- PRO – Very High Performance
- PRO – Very simple to use
- PRO – Support for SDD
- CON – Too simple for computer scientists
- PRO/CON – You don't program EPICS you configure it

Examples -Sites

- APS, Australian Synchrotron, BESSY, BSRF, CLS, DLS, Duke-FEL, ESS, Gemini, ITER, J-PARC, KECK-II, KEK, K-STAR, LANL, LIGO, LNLS, NSCL, NSLS, SLAC, SNS, TJNAF, TRIUMF, Diamond, KEK, NSLS, SDSS,SLS, VECC, ...

Comments (1)

- Its good you are selecting a standard framework early
- Either EPICS or Tango will meet your needs

Comments 2

- More important are :
- Timing architecture and integration with framework
- Consider using a standard for development life-cycle (better than inventing your own) – we use ISO12207

Comments 3

- Choice of platform (Linux, Windows, Real-Time) ?
- Naming convention NOW – appoint naming Tzar

Comments 4

- Why not have a 1 week hands on development workshop to build rapid prototypes of each sub-system – implement the interface definition. You could use this as training and test of scalability.
- TM group could distribute framework very quickly with basic functionality – Alarms, Archiving, Trending, OPI, APIs, ... this would help standardisation. These are TM functions needed now for prototyping