PVAccess Server Interface

Ralph Lange – HZB / BESSY II

EPICS Collaboration Meeting, San Francisco, 05 Oct 2013

PVs and Services

PVs – access to pvData structures
- PvProvider – container of Pvs
- Pv implementation – pvData structure plus methods
  init() get() put() putget() process() monitor() destroy()

Services – RPC type interface (data in / data out)
- User space class that provides
  RPCService implementation – method request()

Both interfaces implemented in Java and C++

Real-Life Service (Java)

```
public class DSService {
    private static class DSServiceImpl implements RPCService {
        private static CFConnector dsConnector = new CFConnector();

        @Override
        public PVStructure request(PVStructure args) throws RPCRequestException {
            PVStructure query;
            try {
                if (args.getStructure().getID().equals("uri:ev4:nt/2012/pwd:NTURI")) {
                    query = args.getStructureField("query");
                } else {
                    query = args;
                }
                return dsConnector.getData(query);
            } catch (Exception e) {
                throw new RPCRequestException(Status.StatusType.FATAL, e.getMessage());
            }
        }
    }
    public static void main(String[] args) throws PVAException {
        RPCServer server = new RPCServer();
        server.registerService("ds", new DSServiceImpl());
    }
}
```

PvDatabase Framework

- Under development (C++)
- Simplifies connecting a database of pvData structures to the PVA server
- Will handle different monitor algorithms, put/get/putget methods, trapping value changes, locking etc.
- Greatly reduces number of methods in API
  init() process() destroy()

Summary

- Server interface is still a moving target
  Refactoring, API changes
- Parts (RPCService) are nice and easy to use
- For the rest, convenience layers are under development

Questions?

Thank you.