Writing a Device Server

Tutorial

Tutorial

How to write a device server in 1 hour

is a bit like

How to dance the tango in 5 minutes !

http://www.bbc.co.uk/broadband/mediawrapper/consoles/strictlycomedancing/nb_rm_console.shtml?pack4-tango_16x9

Tutorial Goals

- *Understand :*
  - the concept of Device Servers
  - the Ultimate Question
  - abstract and concrete classes
  - commands and attributes

- *Know how to :*
  - start POGO and generate a device server
  - generate an abstract class
  - implement a concrete class
  - define a device in the database
  - add a command to a device server
  - add an attribute to a device server
  - compile a device server on Linux

Device Server concepts

- Device servers implement services which are accessed in a standardised way
- For developers : a device server is code which is linked into a process which implements functions which are accessed via a standard protocol over the network
- Device servers are not totally new - remember *subroutines*, *remote procedure calls*, *client-server*, *SOA*, ...

Device Server concepts

- Device Servers are part of TANGO framework i.e. they are part of libraries, there are support programs, they can be scripted, there are tools for developing them and glueing them together

- Some of the advantages of devices servers are :
  - *software is loosely coupled*
  - *they are very flexible i.e. they can offer any service*
  - *the framework takes care of all the system details*
  - *they can be easily implemented, extended and maintained (thanks to the framework)*

Our Example

Apple iSight ieee 1394 camera

Device Server Preparation

- *Requirements*
  - find out what the user wants as service
- *Hardware*
  - identify the hardware you have
- *Software*
  - find out what software libraries you have and whether they provide the necessary functionality

Apple iSight Camera

- *Requirements*
  - read an image
  - get/set features
- *Hardware*
  - ccd camera
  - ieee 1394 interface
  - iidc compatible
- *Software*
  - libdc1394 for Linux can get/set camera images and parameters :

Device Server DESIGN

- *THE ULTIMATE QUESTIONS*
  - my device ***“IS A ...”*** ?
  - my device ***“HAS A ...”*** ?

Device Server Design
Abstract vs. Concrete

“THINK ABSTRACT”

- Try to identify the family of the device i.e. the device type e.g. DigitalInputOutput, AnalogInput, Powersupply, Vacuum Pump, Temperature Controller, Linac
- This will define the AbstractClass of your device
- If someone has already implemented an abstract class adopt or adapt it
- If no AbstractClass exists then define one

- Determine if your device **“IS A xyz”**

Device Server Design
Thinker's Dilemma

POWERSUPPLY ?

CCD ?

VACUUM PUMP ?

BPM ?

RGA ?

SIGNAL GENERATOR ?

STEPPER MOTOR ?

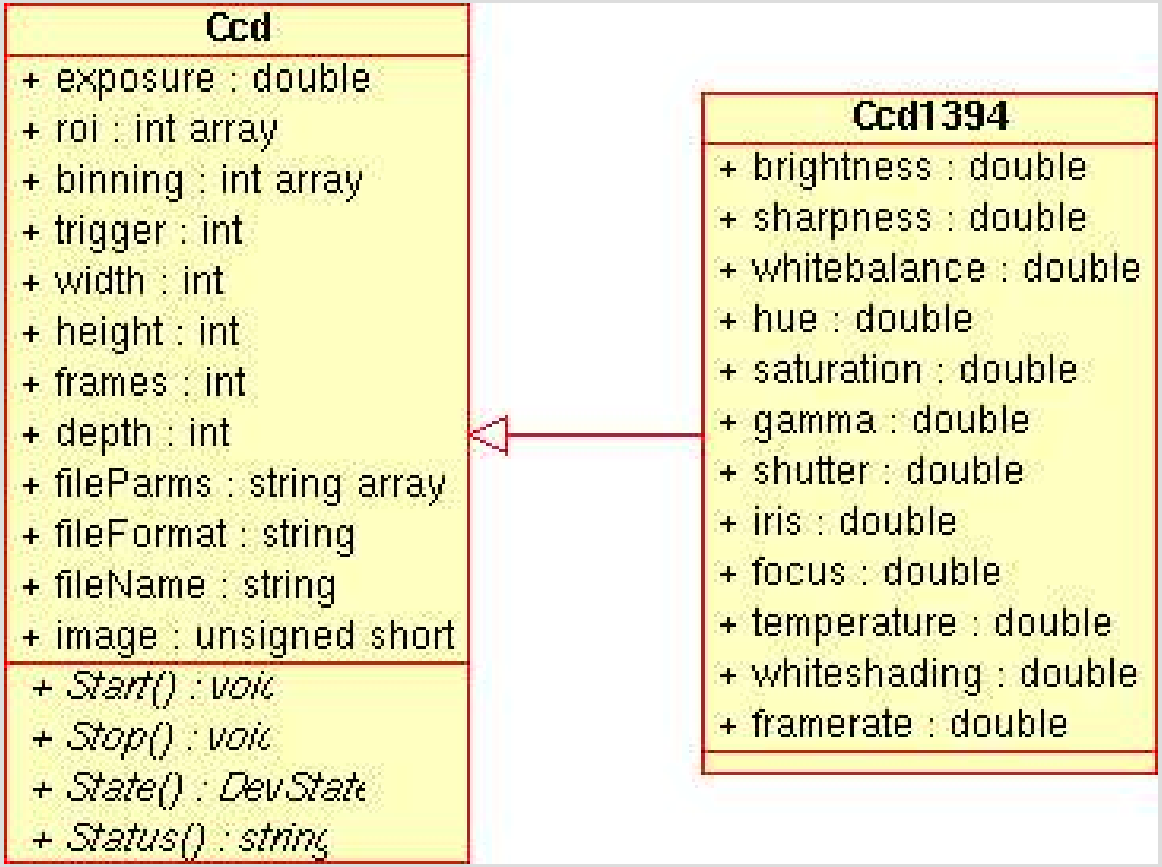
Device Server Designers all too often ...

Apple iSight Camera

- ***It is*** a ***Camera*** with an Ieee 1394 interface
- An AbstractClass exists for cameras called ***Ccd***

- Design decision :
  - we will inherit from the **Ccd** abstract class
  - we will call our concrete class **Ccd1394**

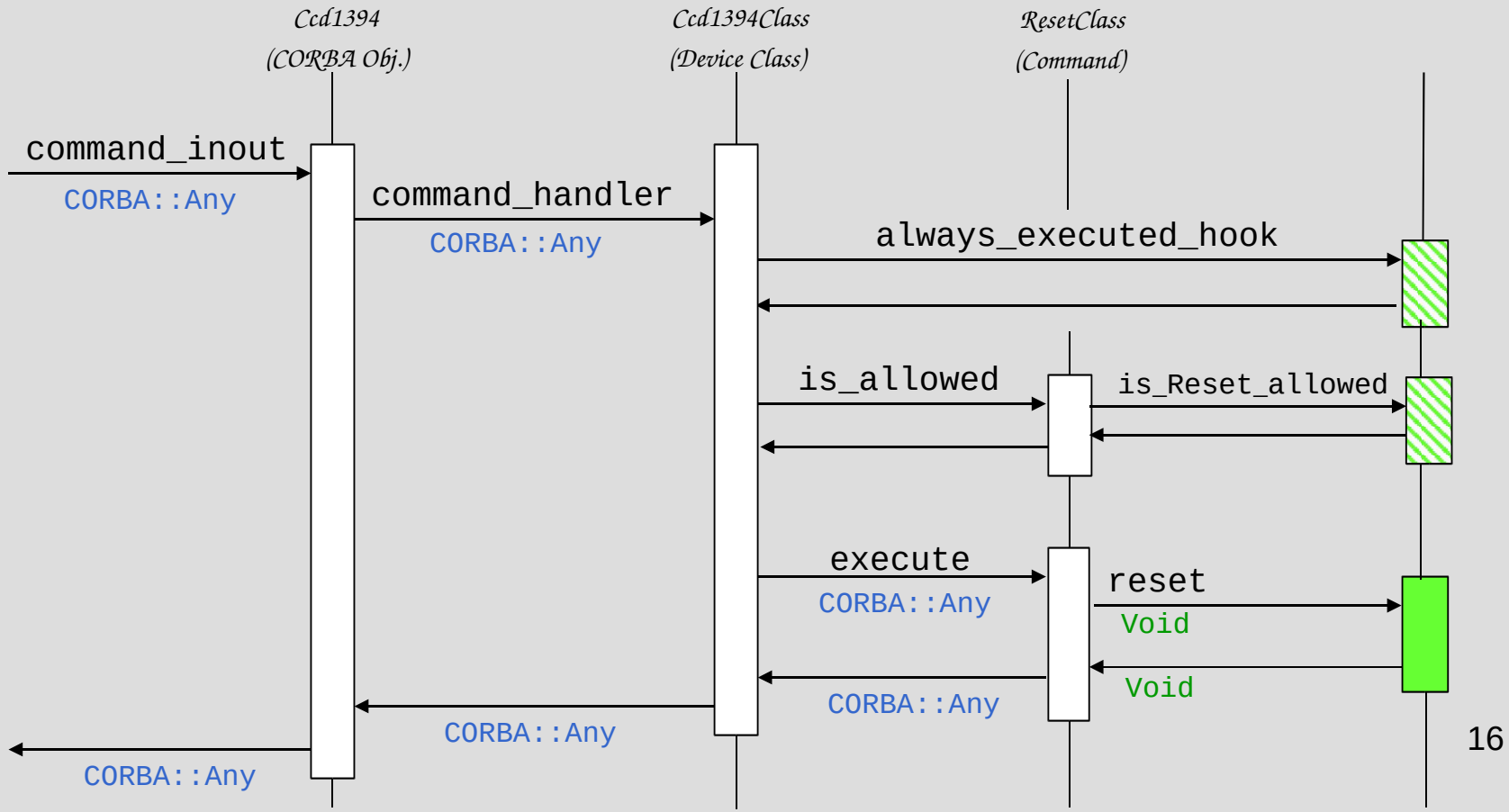
Apple iSight Class Diagram



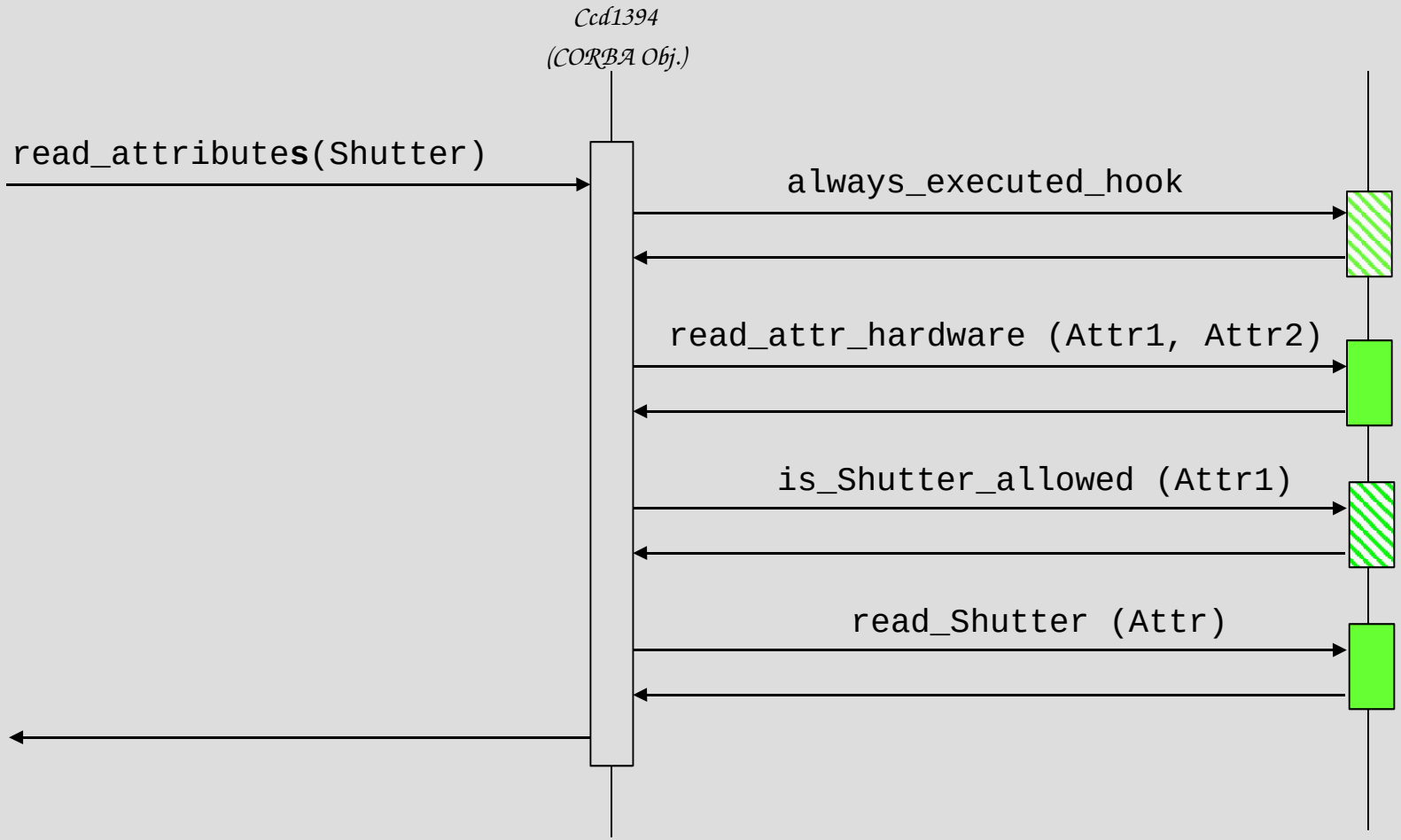
Commands, Attributes and Properties

- ***Commands***
  - are actions e.g. On, Off, Start, Stop
- ***Attributes***
  - are data e.g. current, voltage, position
  - can be READ, WRITE, READ_WRITE or READ_WITH_WRITE
  - have standard properties e.g. min, max, ...
- ***Properties***
  - are descriptive values stored in the database e.g. gpib address, serial line descriptor, ...

Command sequence diagram



Read attribute sequence diagram



Data types

- ***Commands***
  - boolean, short, long, float, double, string, unsigned short, unsigned long, array of these, string+long, string+double and State data type
- ***Attributes***
  - boolean, unsigned char, short, unsigned short, long, float, double, string, State
  - scalar (one value), spectrum (an array of one dimension), image (an array of 2 dimensions)
- ***Properties***
  - boolean, short, long, float, double, unsigned short, unsigned long, string, arrays of short, long, float, double, string

State Machine

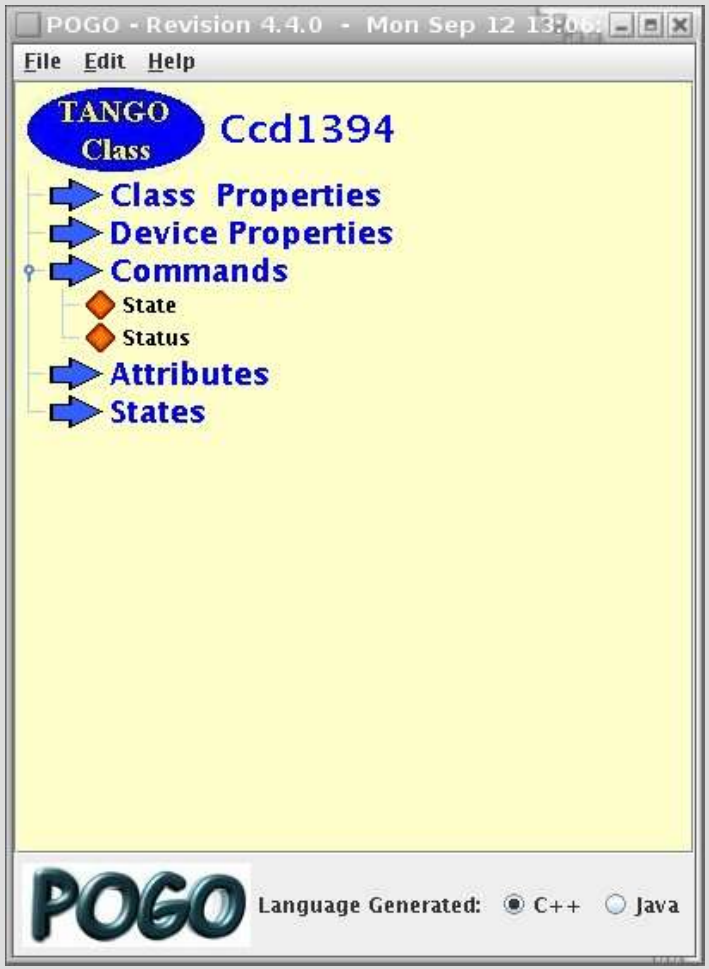
- every Device Server has a State Machine
- the State Machine :
  - restricts which commands and attributes can be executed in which states
  - every command and attribute has a boolean *is_allowed()* method which is called before execution
  - state changes are implemented in the commands or attributes

Allowed States

ON
OFF
CLOSE
OPEN
INSERT
EXTRACT
MOVING
STANDBY
FAULT
INIT
RUNNING
ALARM
DISABLE
UNKNOWN

POGO - THE ONE program to know

- Pogo – code generator for TANGO device servers :



What do you have to code ?

- init_device() - device creation method
- commands() - implementation of command
- read_attr() - read attribute
- write_attr() - write attribute

Reporting errors

- throw an exception :
  - Tango::Except::re_throw_exception(Tango::DevFailed &ex,
    string &reason,
    string &desc,
    string &origin);
- use the logging streams :
  - FATAL_STREAM, ERROR_STREAM, WARN_STREAM, INFO_STREAM, DEBUG_STREAM
  - DEBUG_STREAM << “Hola amigo, que tal ?” << endl;
- C printf style
  - LOG_FATAL, LOG_ERROR, LOG_WARN, LOG_INFO, LOG_DEBUG
  - LOG_DEBUG((“Still %d minutes until lunch !\n”,nb_minutes))

Compiling

- Linux/Solaris
  - use Pogo generated Makefile
  - change the TANGO_HOME variable
  - use Eclipse as IDE ;-)
- Windows
  - make a Visual C++ project
  - follow the instructions in the TANGO manual

Debugging

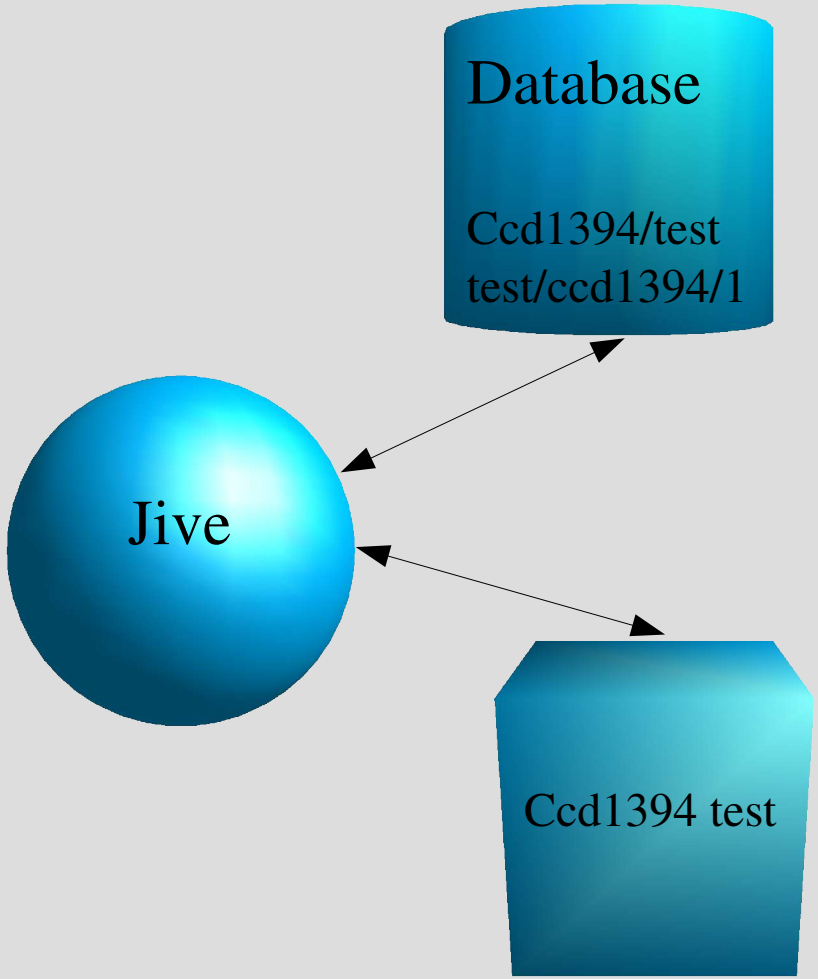
- Use an IDE
  - Eclipse on Linux
  - Visual C++ on Windows
- GDB
  - vivez la “*command line*” !
- Logviewer
  - a “*chainsaw*” application for dissecting logging messages

POGO - DEMONSTRATION

1. Generate simple class + compile + test
2. Inherit from Ccd Abstract class
3. Generate code + compile + test
4. Add code to implement reading image
5. Compile + test
6. Add attributes to concrete class
7. Generate code + compile + test
8. Generate documentation

Defining Devices in the Database

- Database makes the link between the device and the device server
- Two possibilities :
  - define the list of devices served by a server in the database using Jive
  - start the device server without any devices and use the wizard in Jive to create them on-the-fly



Sharing the code

- Put it on SourceForge cvs (project tango-ds)
- Write the documentation
- Publish it to the tango@esrf.fr mailing list

Writing more Device Servers

- This session has only given you a very brief introduction to device servers
- Build new device servers out of existing ones by answering the ***QUESTION –*** device ***"HAS A xyz?"***
- There are over 200 existing concrete classes and 5 abstract classes (project *tango-ds* on *sourceforge.net*)
- What kind of TANGO dancer will you be ?