TANGO Controls

http://www.tango-controls.org/
Outline

What is TANGO?
Language/OS/Compilers
CORBA and ZeroMQ
TANGO Database
TANGO device and device server
Communication models
  Multicast
  Polling
  Events
  Alarms
Logging system
Historical DataBase

Connecting things together

Jive/Starter/Astor
ATK/Jdraw/Synoptic
Qtango/Mango
Taurus
E-giga/Canone
TANGO Bindings
What is TANGO?

In short:

- Control system framework
- Based on CORBA and ZMQ
- Centralized config. database
- Software bus for distributed objects
- Provides unified interface to all equipments hiding how they are connected/managed

**TANGO software bus**

- Device Server
  - HV ps + serial
  - OS/drv.
- Device Server
  - Pylon LIMA
  - OS/drv.
- Device Server
  - OPC UA Modbus
  - OS/drv.
- Device Server
  - Data socket
  - OS/drv.
- Device Server
  - Motion Control
  - OS/drv.
- Device Server
  - TANGO 2 EPICS
  - OS/drv.
- Device Server
  - SNMP
  - OS/drv.

**Scientific workspaces**

<table>
<thead>
<tr>
<th>Device</th>
<th>OS/drv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HV ps + serial</td>
<td>OS/drv.</td>
</tr>
<tr>
<td>Pylon LIMA</td>
<td>OS/drv.</td>
</tr>
<tr>
<td>OPC UA Modbus</td>
<td>OS/drv.</td>
</tr>
<tr>
<td>Data socket</td>
<td>OS/drv.</td>
</tr>
<tr>
<td>Motion Control</td>
<td>OS/drv.</td>
</tr>
<tr>
<td>TANGO 2 EPICS</td>
<td>OS/drv.</td>
</tr>
<tr>
<td>SNMP</td>
<td>OS/drv.</td>
</tr>
</tbody>
</table>

**Industrial SCADA**

- TANGO binding
- TANGO binding
- TANGO binding
- TANGO binding
- TANGO binding
- TANGO Archiving System

**Native client applications**

- MATALB
- IGOR Pro
- LABVIEW
- GlobalSCREEN
- TANGO C++
- Java Python Clients (CLI/GUI)

**Clients**

- Native client applications
- TANGO C++
- Java
- Python
- Clients (CLI/GUI)

**Archiving System**

- TANGO Archiving System

**OS/drv.**

- OS/drv.

**Device**

- Device Server
- Device Server
- Device Server
- Device Server
- Device Server
- Device Server
- Device Server

**Networks**

- GigE
- TCP/IP Modbus
- RS232/Eth
- EtherCAT

**Examples**

- HV ps + serial
- Pylon LIMA
- OPC UA Modbus
- Data socket
- Motion Control
- TANGO 2 EPICS
- SNMP
TANGO release 8.1.2 (C++98, C++11)

Pre-release TANGO 9

Languages
   Server side: C++, Java, Python
   Client side: C++, Java, Python, Matlab, LabView, IgorPro, Panorama

OS – Linux (PREEMPT_RT, Xenomai hard real-time)
   Architecture: x86, PPC, ARM
   Compiler: gcc 3.3 – gcc 4.8

OS – Windows XP/Vista/7
   Architecture: x86
   Compiler: VC9, VC10, VC11

OS – MacOSX
   Architecture: x86
   Compiler: gcc 4.6 – gcc 4.8
CORBA and ZeroMQ

- Common Object Request Broker Architecture specification
- Defines the ORB and the services available for all objects
- Uses an Interface Definition Language (IDL) and defines bindings between IDL and programming languages
- An Interoperable Object Reference (IOR) identifies each object
- TANGO adopts omniORB for C++ and JacORB for Java
  - [http://www.jacorg.org](http://www.jacorg.org)

**ZeroMQ, ZMQ, 0MQ** – [http://zeromq.org](http://zeromq.org)
- An embeddable networking library that acts like a concurrency framework
- Sockets that carry whole messages across various transports like in-process, inter-process, TCP and multicast
- Used for event-based communication in TANGO ≥ 8
Everything which needs to be controlled is modeled as a Device.
Each Device is identified by the Fully Qualified Domain Name (FQDN) 
\texttt{tango://host:port/domain/family/member}
Each Device belongs to a TANGO class and exposes the \textbf{same interface:}
- **Command(s):** act on devices (e.g. power on)
- **Attribute(s):** set/get physical values (e.g. set/get motor position)
  - Attribute properties: per-attribute configuration parameters
  - State: TANGO Device finite state machine value
- **Properties:** configuration parameters

Device level
Class level
Free/Global
The DS is the process where the TANGO class(es) run
Device number and names for a TANGO class are defined within the database, **not in the code**
Which TANGO class(es) are part of a DS process is defined in the database **but also in the code**
The DS **can** host several TANGO classes, each class **can** be instantiated several times  
... **but be careful with code or DLLs not thread safe**
DS configuration is stored into the TANGO database (MySQL)
Advice: design for speed; never, ever do any assumption about the nature and the number of clients → always minimize response time

![Diagram of A Tango device server and Startup sequence](image-url)
Communication models

Two communication models available

**Client/server**: the communication between clients and servers can be synchronous and/or asynchronous
- The *client inquires* the server; the reply can be synchronous (blocking) or asynchronous (non blocking)

**Publish/subscribe**: the communication is event-driven
The device *server informs* the client that something has happened

Additionally, as a special case, **multicast** is also available through ZMQ, that uses the OpenPGM implementation of PGM protocol (RFC 3208 – reliable multicasting Protocol). Has to be configured, defining the global property `CtrlSystem->MulticastEvent` containing the following fields:

- **multicast address**, 226.20.21.22
- **port number**, 2222
- **[rate in Mbit/s]**, 20
- **[ivl in s]**, 10
- **event name**
  - `device/with/multicast/state.change`
Polling

The Polling mechanism allows the Tango device to **decouple** the real device from the client(s) request(s)
Each Tango device server may have **one or more polling thread(s)** (tuning)
Polling allows to continuously monitor the “health” of the equipment
**Attributes and/or Commands** can be polled
The polling result is stored in a **buffer with configurable depth**, just limited by available memory
A client is able to read data from:
- The real device
- The last record in the polling buffer
- The polling buffer with fall-back to the real device

**The complete buffer history is also available to the client → large buffers mean “automatic” shared memory mechanism available**

Advice: the frequency of real hardware access has to be tuned on the equipment (e.g. accessing that old reliable 9600 baud serial line...)

---

25-27.03.2015 LMC Standardization Workshop 9
Events

Implement the publish/subscribe pattern; based on ZeroMQ since Tango 8 (no more notification service)
Available on **attributes**
The client registers her interest **once** in an event (value)
The server informs the client every time an event has occurred
**Default based on device server polling**: needs configuration but does not require changes in the device server code
Additionally the event generation can be managed by the developer: **events pushed by code**
Client callback executed when an event is received
Six types of events available:
- **Change**: absolute change, relative change
- **Periodic**: period
- **Archive**: absolute change, relative change, period
- **Attribute configuration**: no parameters
- **Data ready**: managed by the developer
- **User**: managed by the developer

Heartbeat to check that the device server is alive (10s)
Alarms

Device alarms
- Warning and alarm **thresholds available** as per-attribute configuration
- TANGO changes the State of the Device and the Quality factor of the attribute depending on attribute value and thresholds

TANGO alarms
Specialized TANGO device servers, useful to handle complex alarm rules based on multiple values/multiple logics
- C++ alarm device server: event based
- Python alarm device server: polling/event (with Taurus)

Parser for arbitrary alarm formula support
```plaintext
kg01/mod/linkstabilizer_kg01.01/State == ON && kg01/mod/linkstabilizer_kg01.01/Drift1_Threshold && \nabs(kg01/mod/linkstabilizer_kg01.01/Drift1_rate) > kg01/mod/linkstabilizer_kg01.01/Drift1_Threshold
```

Support for alarm groups and alarm levels (LOG, WARNING, FAULT)
Support for external command execution (TANGO DS)

**Scalability**: any number of TANGO alarm servers can be deployed, based on requirements, architectural constraints, performance required...
TANGO groups provide the user with a **single control point for a collection of devices**. For instance, the TANGO Group API supplies a `command_inout()` method to execute the same command on all the elements of a group. 

Tango Group is also a **hierarchical object**: in other words, it is possible to build a group of both groups and individual devices. Simple and effective way to create logical views of the control system.

**Example: Beam Loss Monitors**

```python
blm = Group('radiation_protection')
blm.add('*/radiation_protection/*')
if blm->ping() == True:
    print "all devices alive"
else:
    print "at least one device dead"
```

**193 total device number**
Two kind of users (identified by system login name):
- users defined in the ACL
- users not defined in the ACL → rights fall below “All users”

Two kind of rights, at host and device level:
- Read (+ optional per-class allowed commands)
- Write

**taurel**
- write to sr/d-ct/01 and fe/*/* only from pcantares
- read all other devices only from pcantares

**verdier**
- write to sys/dev/01 from any host on 160.103.5.0/24 subnet
- read all other devices from the same subnet

**all users**
- read-only access from any host
The TANGO logging system allows a device server to send messages to:
- The console
- A file
- An application called LogViewer (GUI)
- A file on a remote host via specialized TANGO device server exposing the appropriate API

Six logging levels: DEBUG<INFO<WARN<ERROR<FATAL<OFF

LogViewer: Java graphical application to display, filter and sort logging messages
Historical Database

HDB (Java) - Set of three databases
- HDB: permanent, up to 0.1 Hz archiving rate
- TDB: temporary, up to 1 Hz archiving rate
- Snap: context save/restore
- Support for Oracle and MySQL RDBMS
- 4(+3)+3 Device servers
- **Polling** based
- GUI: Mambo, Bensikin

HDB++ (C++)
- One database for slow and fast archiving (up to 1 Khz)
- Support for existing HDB schema on MySQL
- Support for **hdb++ new schema** with improved features (μs timestamp)
- Support for **noSQL** backend (Apache Cassandra)
- 2 Device servers (EventSubscriber, ConfigurationManager)
- **Event** based
- Fast data extraction library
- GUI: HdbConfigurer, qhdbextractor (plotting)
- **Scalability**: same as TANGO, deploy as many DS as you need

TimeMachine
- System restoring tool based on context, HDB++ archived data and extraction library
Historical Database

Connecting things together

HDB++ Configurator

Device Filter:

Archive event properties:
- abs_change: Not specified
- rel_change: Not specified
- period: 3600000

Pooled attribute name = trigger_missing
Polling period (ms) = 3000
Polling ring buffer depth = 10
Time needed for the last attribute reading (ms) = 0.118
Data not updated since 238 ms
Delta between last records (in ms) = 3000, 2999, 2999, 3000
Administration: Jive

TANGO database browser and device configuration/administration/testing tool

![TANGO database browser interface](image)
Administration: Starter/Astor

Starter: TANGO DS to manage device servers on hosts
Astor: control system manager GUI
Development: Pogo

Pogo is a TANGO class generator

Generates C++, Java and Python

Source code and html documentation

The class skeleton is saved in a .xmi file

Well defined areas for programmer's code
Application ToolKit: provides a framework to speed up the development of TANGO applications

Core of any TANGO Java client

ATKpanel: generic GUI (data introspection)

Use Jdraw to draw the specialized synoptic

Design your own specific ATK application Using your favorite Java IDE

Final result...
GUI: Qtango/Mango

Qtango
- A multi-threaded framework to develop TANGO applications
- Based on Qt
- API to manage/talk to TANGO devices
- Widgets to draw the GUI
- For programmers

Mango
- An on-line designer to easily create graphical interfaces based on Qtango
- Quick development of simple GUI
- Useful for the device server programmer, the control room operator, the tests, the end-user
A library for connecting client-side apps (CLI/GUI) to TANGO device servers
Based on PyTango python bindings for TANGO
GUI built on top of PyQt python bindings for Qt
E-Giga: a WEB interface to historical archive data
Canone: a tool to develop WEB interfaces to Tango devices
Access TANGO control systems from different high level “programming” environments.

TANGO provides bindings for the following “languages”:

- C language (partial support)
- Matlab (>= R2009b)
  Windows and Linux, 32 and 64 bit
- Octave (>= 3.6.2)
  Windows and Linux, 32 and 64 bit
- LabVIEW 2010 → 2012
  Windows, Linux, MacOSX, 32 and 64 bit
- LabVIEW 2013 (2.0.0 RC2)
  TANGO 8.1.2 with patches; Windows and Linux, 64 bit
- Igor Pro (>= 6.0)
  Windows, Linux, MacOSX, 32 and 64 bit
- Panorama
  Tango 7.2.1, Windows, 32 and 64 bit
A Tango control system must be hierarchically (logically) organized.
 Devices associated with hardware equipments usually live at lower level.
 Higher level devices aim to:
   - abstract functionalities from mechanisms
   - group similar devices
   - implement services based on many low level devices (e.g. alarms)

Each domain is identified by the TANGO_HOST/port couple, e.g. by the TANGO Database.
An arbitrary number of devices may belong to a domain, just limited by available memory, processing power, network bandwidth (Operating Database limit ~ 5*10^5 devices).

...but...
Multiple domains can be configured in a control system
  - Complex systems must be splitted into different Domains
  - Each Domain must be hierarchically organized

Multiple domains + Device hierarchy + Peer-2-Peer architecture = unlimited scalability