Tango Controls – Kernel meeting 19-20 September 2016 @ ESRF

Background

A Tango Controls kernel meeting was held from the 19-20 September at the ESRF. Present were representatives of the contributor member institutes (ESRF, SOLEIL, ELETTRA and ALBA) and the core developers. The goal of the meeting was to discuss the work to be done until the end of 2016 for preparing TANGO V10. One of the main motivations was to agree upon what should be done for V10 with the resources financed by the Tango Controls Collaboration. This document summarises the outcome of the meeting.

Present

The following people were present: N.Leclercq (SOLEIL), L.Pivetta (ELETTRA/INAF/SKA), G.Cuni, C.Pascual, S.Rubio (ALBA), and E.Taurel, R.Bourtembourg, P.Verdier, J-M.Chaize, A.Götz (ESRF), I.Khokhriakov (IK)

Goals of meeting

The meeting was organised around these goals :

- Decide what are the main features of V10
- What backwards compatibility to keep ?
- How to replace CORBA + questions to answer
- What needs to be prototyped ?
- Agree on the Git repositories on GitHub
- Should V9 be feature frozen and if not
    - what features to add to V9 ?

What is TANGO ?

When discussing what should be in V10 it is important to agree upon what is TANGO today and what needs to be maintained from one major version to another without making significant changes to the model. The following (non-exhaustive) list of features are currently supported in TANGO:

- A framework for implementing distributed systems with device servers
- A database + device servers + clients
- A binary protocol (CORBA+ZMQ)
- Framework supported in C++, Python and Java
- Many TANGO sites (>40 today) some of them operate large installations with hundreds to thousands of servers + clients
- The Device Server Model made up of:
    - Device, Commands, Attributes, Events, Properties, Data types
    - One thread per client, multi-threaded, 4 serialisation models

    - Binary protocol – supports synchronous, asynchronous and events,
    - Full support i.e. client and server, for multiple languages
    - Backwards and forwards compatibility at runtime between all releases
- Tools :
    - Jive, Pogo, Astor, HDB, HDB++, Alarms
- Graphical toolkits
    - ATK, Taurus, QTango, jsTango
- Web toolkits
    - mTango

Backwards compatibility

A critical feature of Tango is the runtime compatibility between all releases i.e. V1 can talk to V9 and vice versa. This allows new major releases to be deployed in a running system without porting and/or recompiling all existing servers. Although some sites prefer to recompile all servers other sites (e.g. ESRF) depend on this feature and cannot rebuild all servers and clients without a huge effort. It is essential for V10 to ensure a high level of compatibility especially if we plan to make the protocol (i.e. CORBA) pluggable. Compatibility was a key factor when introducing ZMQ for the event system – both protocols are supported so events work between CORBA and ZMQ based events.

The following options of compatibility were studied:

- **Option 1** :
    - ***Source code compatibility with minor changes*** – this is the holy grail of compatibility. Clients and servers only need to be recompiled with no or minor changes e.g. replace CORBA::string_dup(). Runtime compatibility remains. This option was considered the goal to aim for but the amount of effort required needs to be evaluated. In this option a device server and client will link with versions of the kernel library supporting multiple protocols.
- **Option 2** :
    - ***Run time compatibility via a bridge but same Tango Device Server Model*** – in this option the source code compatibility and all the features of the Tango DSM are maintained but runtime compatibility is ensured by a bridge. This option was considered to have second priority and only be envisaged if Option 1 turned out to be too resource intensive.
- **Option 3** :
    - ***Two versions of Tango with new DSM and old API*** - in this option a new version of the Tango DSM with different or fewer concepts would be proposed e.g. a different concept of the server which implements only the protocol but not the DSM framework i.e. state, commands and/or attributes. The DSM framework is provided as optional on top of the protocol. This proposal was considered too far in the future to be implemented in V10. The priority for V10 is to make the protocol pluggable.

Scenarios

The following scenarios were considered

- Option 1 :
  - Refactor TANGO library and make CORBA a pluggable protocol (this means replacing all the CORBA features used). In this option all the CORBA code will be moved into a few files to make it easier to identify and replace. An API will be implemented to access the CORBA code. The API will have a limited number of calls which will be the basis of a protocol agnostic API. In the future new protocols can be added by implementing this API[1]. The API will be defined by the kernel developers as one of the first tasks needed to make CORBA pluggable.
- Option 2 :
  - Rewrite TANGO to implement a new pluggable protocol based on ZMQ (provide only runtime compatibility). In this option a new protocol would be implemented without trying to mimic CORBA. It would instead define an API which is protocol agnostic and the first implementation would be with ZMQ. Backwards compatibility would only be provided at runtime. Compile time compatibility would be limited.
- Option 3 :
  - Integrate omniORB/jacORB into TANGO and don't replace CORBA. In this option the main issue with CORBA i.e. that it can be difficult to install and that this could increase in the future, would be addressed by integrating the code from omniORB into the TANGO library and removing the external dependency on omniORB. For Java the external dependency on JacORB would be replaced by the ORB in OpenJDK. Issues which would need solving include setting the timeout at runtime (for Java) and identifying the minimum code needed from omniORB. The latter issue could be addressed with help from the author (Duncan Grisby).

After discussion it was decided to go for Option 1 first, with Option 3 as a possible way of ensuring the long term future of the CORBA protocol. Option 2 will require Option 1 but will not be implemented in V10. It will be left for the future to be implemented with the help of the community which needs a different protocol most. This means Tango V10 will be based on a pluggable protocol and the first protocol supported will be CORBA.

The order of actions are therefore:

1. Refactor kernel library to make protocol pluggable for V10
   - First protocol plugin will be CORBA (V10)
2. Remove external dependency on CORBA by integrating the parts we use in omniORB if possible (contact Duncan) for V9

---

1 An open question is if this API should be exposed to users who want to implement their own minimalistic device servers and clients on embedded platforms e.g. FPGAs or new languages.

3. Replace dependency on JacORB with OpenJDK
    - Fix problem with timeout (2 timeouts?)

Decisions

Based on the above discussions the following decisions were taken:

- Implement Option 1 of the pluggable protocol, study Option 3 (integrating CORBA) and keep Option 2 for a future release.
- Keep runtime compatibility and aim for a maximum amount of source code compatibility
- Runtime compatibility >= Tango V8 (drop support for Notification)
- Deprecate all CORBA:: references in API (string_dup())
- Keep support for DevVarxxx
- Remove Transient and other CORBA error messages (cleanup error messages) !
- The main limitations of CORBA are installing it and its long term availability. Technically it is doing a very good job.
- Port server part for >=V8
- Drop all classes for Java server <= V2
- Convert Makefiles and automake to Cmake > V2.8.9

Kernel library V10

The actions for kernel developers (Igor with help from Reynald and Gwenaelle) until end of 2016 are:

- Refactor code to isolate CORBA code
- Define primitives for protocol
- Do this for C++ (first) and Java (second)
- Goal is working V10 running on CORBA as a plugin
- Support Java >= 8 and require C++11
- Runtime compatibility to support CORBA + new protocol(s) !!!
- Study impact of source code compatibility

Move to Git

All changes or refactoring of the code will be done AFTER the move to github. As part of the move to github the following must be completed before starting any of the above tasks:

- Setup CI for Linux with Travis (IK) and for Windows with AppVeyor (??)

- Move project build to Cmake (IK)
- Move svn tracker tickets to github issues (RB)
- Move PyTango+itango, remove tango-cs (TC)
- Move+mavenize Jtango (IK), Astor+Starter+Pogo(PV)
- Move Atk+Jive (JLP)
- Move Hdb++ (LP+RB)
- Move yat4tango (NL)
- Create tango-doc repo (3-controls)
- Move Bindings (Labview+Matlab+Igor) (NL), C-binding (JM)
- Inform other projects (JMC+AG)
- Announce official move + freeze svn (RB+IK)
- Deadline end of November

The move to github will be mainly done by Igor and Reynald with help from Emmanuel, Pascal, Tiago, Gwenaelle and Carlos. A basic set of commonly used platforms will be supported but each institute will be responsible for the continuous integration for their specific platform(s). This document proposes the following platforms be supported (to be confirmed) :

- Debian 7 + 8 (ESRF)
- Ubuntu 14.04 + 16.04 (Elettra proposed)
- Windows 8 (SOLEIL/ESRF proposed)
- OpenSuse (ALBA proposed)
- RHEL (SOLEIL proposed)
- CentOS (MAXIV proposed)

New features

The following new features were discussed and should be implemented with high priority once the code has been refactored:

1. Self-describing data types as arguments for Commands – this has been requested by SKA. It could be solved by implementing the DevPipeBlob type as an argument for Commands. This proposal needs **TO BE CHECKED with SKA** to see if this satisfies their request and when they need it.
2. Long Term Support – V9 will be the first version to have LTS (5 years). This request depends purely on having resources to do backports of patches. The meeting proposed that the Tango Collaboration Contract pays for this feature in V9. This means V9 will continue to be supported and releases based on the code in the github repository.

Next meeting

Everybody agreed this kind of kernel meeting is very useful and should be repeated regularly. The length and size were considered to be right. The next meeting should take place in 6 months time. A dedicated meeting for the Java kernel will take place between the Java kernel developers.

Progress can be followed in the meantime via github:

https://github.com/tango-controls