ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA058

STATE OF THE TANGO CONTROLS KERNEL DEVELOPMENT IN 2019

A. Götz, R. Bourtembourg, J-M.Chaize, P.Verdier, ESRF, Grenoble, France
T. Coutinho, J. Moldes, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
L. Pivetta, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
I. Khokhriakov, O. Merkulova, IK company, Moscow, Russia
S. Gara, NEXEYA Systems, La Couronne, France
P. P. Goryl, M. Liszcz, S2Innovation, Kraków, Poland
A. F. Joubert, SARAO, Cape Town, South Africa
G. Abeille, SOLEIL, Gif-sur-Yvette, France
G. Mant, STFC, Daresbury, Warrington, Cheshire, United Kingdom
T. Braun, Byte Physics, Berlin, Germany
V. Hardion, MAXIV Sweden, Lund, Sweden
M. Bartolini, SKA Organisation, Macclesfield, United Kingdom

Abstract

This paper will present the state of of kernel developments in the Tango Controls toolkit and community since the previous ICALEPCS 2017. It will describe what changes have been made over the last 2 years to the Long Term Support (LTS) version, how GitHub has been used to provide Continuous Integration (CI) for all platforms, and prepare the latest source code release. It will present how docker containers are supported, how they are being used for CI and for building digital twins. It will describe the outcome of the kernel code camp(s). Finally it will present how Tango is preparing the next version - V10. The paper will explain why new and old installations can continue profiting from Tango Controls while maintaining their investment. In other words in Tango "the more things change the better the core concepts become".

INTRODUCTION

Tango Controls is a software toolkit for building object oriented control systems. It has been adopted at a large number of sites around the world either as the main toolkit for their control system or for a sub-system or commercially acquired systems. A growing number of commercial products and control systems are now based on Tango Controls. A few commercial companies offer paying support for anyone needing help in integrating Tango Controls into their system.

With this wide user base it is important that Tango Controls continues to remain an active project and offer a clear roadmap for the future. Tango Controls has regularly given an update at the ICALEPCS series of conferences of the state of the kernel development. This paper follows in this tradition. It will provide an update of what has been achieved over the last two years as well as provide a summary of what is planned next. Readers are encouraged to refer to following papers about kernel development presented at this conference for more details on specific topics: MOPHA051, MOPHA050, WEPHA020, WEPHA056, WESH3003, and other papers on using Tango.

The main objectives of kernel developments for Tango Controls since 2017 has been to continue consolidating the continuous integration for all major platforms, maintain the Long Term Support version V9, i.e., bug fixing and add features which are strongly needed by the community and do not break compatibility with the, improve the web development platform, continue improve the documentation and maintain the website. This paper summarises these developments.

COLLABORATION

The Tango Controls collaboration is composed of several partners sites who have committed -by signing a Collaboration Contract- to financing the development and the animation of the community. The Tango Controls collaboration is growing year after year. In 2017, the largest astronomical project in the world, the Square Kilometre Array (SKA), joined the collaboration with 2 members: SKA Organisation (Manchester, United Kingdom) and SARAO (Cape Town, South Africa). In 2019, the Extreme Light Infrastructure (ELI) institute in Czech Republic joined the collaboration, to make up 11 partners financing the core development. Each partner nominates a representative on the steering committee to vote and follow-up the budget, define the strategy and the milestones.

The ESRF hosts the collaboration body and is in charge of executing the development program voted by the steering committee. The collaboration budget allowed the collaboration to boost the software development, maintenance and animation effort by sub-contracting certain tasks to commercial companies.

KERNEL DEVELOPMENT

C++ Core Library

The stable (Long-Term Support) branch of the Tango C++ Kernel is steadily evolving. The development has accelerated since the move from Sourceforge to GitHub [1] and integration of various services like Travis for continuous integration and Sonar for code quality. The open nature of GitHub encourages collaboration within the Tango Community.

Recently a new version, cppTango 9.3.3, has been released [2]. It is the first stable release since 9.2.5 in 2017 and the move to GitHub. It contains numerous improvements over the previous major 9.2.5 release, like performance optimizations for devices with a large number of attributes or possibility to configure float or double properties to a NaN (not a number) or infinite value. A big work has been done on the event system to make it more reliable and avoid heartbeat errors when Tango is used with some special network configurations, and to work around a bug in some ZMQ versions which was preventing the event subscribers from receiving more than 1000 events. The latest release provides better compatibility with modern versions of GCC and Clang when compiling the source code in C++14 mode. This is especially important for device server developers as they can use up-to-date compiler tool-chains for their work. The test suite has been refactored into smaller, isolated test cases. Additionally, the new Tango C++ Kernel release contains many small changes, bug fixes and code quality improvements.

The API documentation has been updated to support the search feature. There is ongoing work that aims to migrate the documentation from Doxygen to Sphinx [3].

Python Core Library

Python has become the most popular programming language in the Tango Controls community in recent years. This is largely due to the excellent PyTango binding for Tango to C++. It provides a high level python friendly API which makes developing servers and clients extremely easy.

PyTango 9.3.1 has been released in August 2019 [4]. This marks the fifth release since the previous review at ICALEPCS 2017 [1]. Eight contributors have been working to the project producing more than 400 commits, in total. PyTango 9.3.1 supports the releases 9.3.x of the Tango C++ library - the minor versions are kept in sync. No major updates were required in this period, so it was largely maintenance.

The PyTango library is compatible with both Python 2 and 3. As the end-of-life for Python 2 approaches (1 January 2020) [5], other Tango projects using PyTango are transitioning to Python 3. This brought to light some issues with the Python 3 compatibility, which have largely been addressed. Some improvements were made to the asyncio [6] "green mode" support as well.

A notable addition is that pre-compiled packages for Microsoft Windows are now available directly from the Python Package Index, PyPI [7]. These are built for various CPU architectures and Python versions using the AppVeyor [8] continuous integration tool. Note that the C++ Tango library bindings are statically linked - currently using version 9.3.3.

There is ongoing work to make the high level application programming interface (API) more comfortable for Python users, i.e., more "Pythonic". Work on the Tango DevEnum data type has already been completed which maps it to a standard Python enumerated type. An area for future work is logging - the standard Python logger should be available in all device servers. Log handlers could then be configured for various targets, such the Tango Logging Service, syslog [9], or Elasticsearch [10].

The major rework of changing the C++ binding to use pybind11 is still ongoing, and discussed in a later section.

Java Core Library

As for the C++ and Python libraries, the Java library is continuously maintained. Last changes were mainly focused on bug fixing and continuous integration tooling. We have setup automated releases and from now on, on each git tag, Travis will automatically compile, test and deploy the JTango binaries into Bintray [11]. These binaries are also available in the Maven Central repository [12]. JTango is still compiled with Java 7 to maintain compatibility with current users, and we will move to Java 8 by the end of 2019. From Java 11, its default CORBA implementation has been removed from Java [13], but it not an issue as it is still available in Jacorb [14], that is already used by JTango.

PYBIND11 PORT

The current version of PyTango (9.3.1) utilises Boost as the binding layer between Python and C++. It exposes the Tango C++ API to Python and vice-versa. A project is underway to replace this layer with pybind11 [15]. This will have the advantage of a lightweight header-only implementation utilising the modern feature of C++11 and beyond. It will lead to faster compilation times and a smaller overall footprint. At the same time, much of the code can be simplified taking advantage of pybind's in-built features: automatic type conversion, STL containers, smart pointers, internal reference counting, numpy, and buffer support, to name a few. Support for overloaded functions and inheritance of virtual and pure virtual is also essential and simplified.

A simple example of our conversion to pybind11 shows a single method of a python client calling C++ to report its name. The key features here are the use of a C++ lambda function whereby the calling arguments and return type are clearly defined.

```
.def("name",[](Tango::DeviceProxy& self)->std::string {
    return self.name(); // Tango C++ signature
 })
```

Most of the client code has now been ported and the server side is well under way. Conversion has not always been straight forward with the main stumbling blocks being threading issues in the server side code and asynchronous command callbacks, however, release for preliminary testing is foreseen to be the end of the year. For further information see pybind11 documentation on readthedocs [16].

REST API

The Tango REST API is an important addition to support web interfaces "out-of-the-box" (Fig. 1). Some major improvements were made to the Tango REST API. The project was restructured to separate specification from reference implementation and tests suite. The new 1.1 version of Tango REST API specification was released as well as its Java implementation in November 2018. This version features adoption of HTTP/2.0 and Subscriptions API which allows clients to subscribe to upstream native Tango events and to be notified via event stream [17]. In conjunction with SSE, HTTP/2.0 forms a powerful bi-directional communication protocol from HTTP based clients to Tango Controls applications and vice versa.

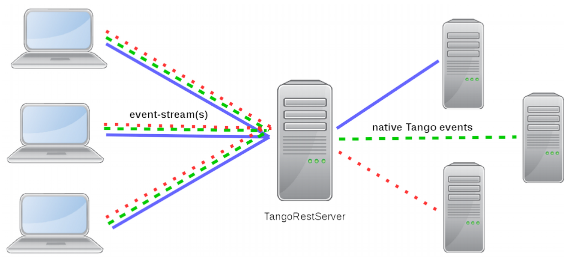

Figure 1: Tango REST API events with SSE

WEB TOOLKIT

A new web technology project enters the Tango portfolio. WebJive is a web-based application for visualising and interacting with a Tango control system. This is an attempt to recreate a basic device browser with modern web technology, similar to the desktop application Jive. The development is based on the REACT framework for the User Interface and GraphQL for the communication with Tango (Fig. 2). React is a JavaScript framework which is component oriented. This framework made by Facebook is very popular among frond-end web developers. The backend server so call TangoGQL provides a GraphQL API which is a declarative language for querying data. Returning only the data you ask for can drastically reduce the number of calls needed for getting the information. Furthermore the Tango event system has been connected to the GraphQL event-based subscriptions, making use of a unique and standard protocol. Additionally the server can be connected to a Central Authorisation System to limit the access in reading/writing attributes and executing commands.

A user interface construction tool has been developed based on the same architecture. The user can create their own dashboards online and share them. It comes with an array of basic built-in widgets, such as buttons that issue commands and plot diagrams that display attribute values over time. Furthermore, its architecture makes it straightforward for a developer to extend it with new widgets. The aim of the tool is to empower anybody who works with a Tango control system by enabling them to create advanced GUIs in a matter of minutes without any programming.

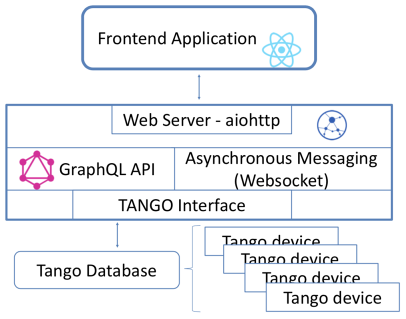

Figure 2: TangoGQL a GraphQL backend for Tango

WebJive is developed by MAX IV Laboratory and the Square Kilometer Array (SKA.). A Tango slack channel "#webjive" has been opened to facilitate the collaboration for the development.

WALTZ

During the 32nd Tango Collaboration meeting, the Tango community voted to rename TangoWebapp: it is now known as Waltz. Since 2017 Waltz reached a mature state as an end user application and as a platform for Tango web applications. Being used in production at DESY for about one year, it proved to be a first choice tool for beamline commissioning at PETRA P05 + P07. The latest release includes a wide range of useful widgets, that simplify monitoring and controlling, as well as managing, Tango devices: multiple dashboard profiles for control and monitoring, Tango hosts and servers manager, redesigned Tango device control panel, etc. [18]. As a platform, Waltz offers rich possibilities to develop user oriented widgets. At PETRA P05 and P07 it is used to configure the data acquisition layer. A comprehensive description on Waltz as a platform can be found in [19].

WEB SITE

Since December 2017 a new Tango Controls website has been online. All information of the previous website was analyzed, redesigned and restructured. This was done to make visitors feel more comfortable and navigate through the website faster. This reorganization was accompanied with a huge Tango Controls documentation reorganization and all information dedicated to the documentation was moved there from the website. New sections were added. Among them are GitHub info and a mind-map with all of the Tango Controls Ecosystem, see Fig. 3.

TANGO V10

The main quality attributes of the architecture of the current Tango version were highlighted in Tango V10 design document [20]; moreover, a wide-ranging analysis of the existing Tango Controls code base was done, to define a possible evolution strategy for Tango V10, and included in the report. A dedicated meeting was organized to discuss the outcome of the analysis. As the result of

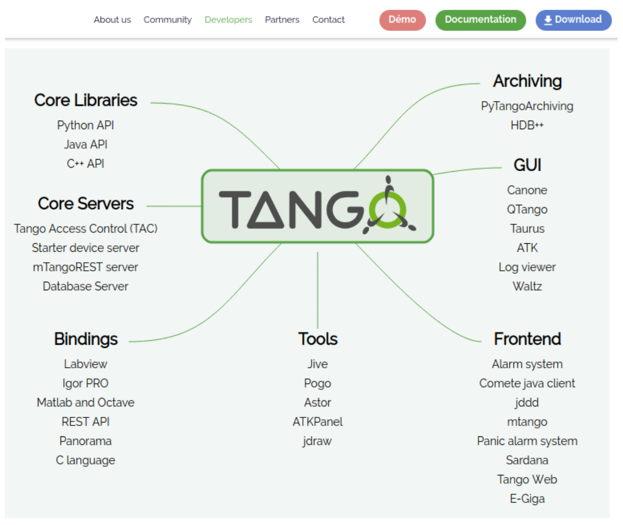

Figure 3: Tango Controls Ecosystem Mind Map

the discussion it was decided to continue the analysis and to involve more community members starting the Tango RFC project [21]. Inspired by the ZeroMQ project, the Tango community decided to adopt the Collective Code Construction Contract (C4) [22] to create and modify RFCs. In addition, RFC life-cycle has been defined to adhere to the Consensus-Oriented Specification System (COSS).

The Tango RFC project formally specifies the Tango Controls protocol as it is now (v9.3.3). It will be a base for the specification of Tango V10. The Tango V10 will probably not use CORBA transport communication. However, the main qualities and features of the Tango Controls will be preserved. It is planned that from the user perspective, the only change will be long term sustainability and better performance. For this purpose, the protocol specification is building separate RFC documents. Some of the RFC documents specify certain protocol features and actors behaviour, and some of the RFCs specify current transport implementation based on CORBA/Omniorb and ZeroMQ.

The specification builds up layers, which allows for a next step. The next step is to select specific features of the current Tango Controls protocol and propose its implementation with a new transport protocol by developing client and server prototypes.

CONTINUOUS INTEGRATION

The Tango kernel projects use *Travis* [23] for continuous integration builds for Linux platforms. *Travis* is currently set up to compile and test *cppTango* on Debian Wheezy, Jessie, Stretch and Buster. *PyTango* is built and tested by Travis for python 2.7, 3.5, 3.6 and 3.7. *cppTango* development Debian packages are generated and deployed automatically on *Bintray* [24] when a new tag is created. *Travis* is used by *JTango* to automatically deploy new releases to *Bintray* and to create automatically a Github release with a fat jar as asset when a new git tag is created. *Appveyor* [8] is used to automatically build the Tango C++ library and the *PyTango* dependencies for different Windows compilers: currently from MSVC9 to MSVC15 in 32 and 64 bits, but MSVC14 and more recent compilers in 32 and 64 bits in the next *cppTango* versions. *Appveyor* decided to change its artifacts retention policy so build artifacts older than 6 months are now automatically deleted. To overcome this limitation, the artifacts are now saved as Github releases assets. *Appveyor* builds depending on artifacts from other repositories are now downloading these files from the Github release assets.

CONTAINERS

Although nowadays there are many technologies and standards for containerising applications, Docker is the Tango Community's container of choice. Following the release of the first official Docker image with Tango and using it for the Tango Kernel continuous integration, Docker is gaining more and more adoption in the Tango Community. The Tango image has been updated to match the latest Tango Controls version and is now built directly from the Tango Controls source code, as opposed to being installed from a Debian binary package. There is ongoing work that aims to further modularise and refactor existing Tango images [25].

The Docker is also extensively used by the TangoBox virtual machine. Starting with TangoBox 9.2 release, more and more services were moved into dedicated containers. The TangoBox virtual machine image is now available on AWS [26] making it even easier to try out Tango Controls and many of the tools developed for Tango.

In version 9.3 of TangoBox, the following applications and services are running in their own containers: Archiving (HDB, HDB++, E-giga), Waltz, JLinac and Modbus simulators, JupyTango. The images with these services are automatically built in GitLab CI and published to a container registry. This allows for better manageability and software reusability and eases upgrades of the VM's base system. There are plans to move even more services, including Tango's Database device server, into containers. The goal is to have a fully dockerised Tango Controls.

Docker has proven itself to be very useful for development purposes and automated testing of device servers in CI. It allows developers to easily test software against multiple versions of Tango Controls to ensure portability and compatibility. This is especially important for companies like S2Innovation, who develop software for the industry. Often they are required to ensure compatibility with older Tango releases while and at the same time they want to be able to run the software also on the latest versions of Tango.

SKA Use Case

The SKA Project is investing in the implementation of a software development strategy based on the principles of Continuous Delivery and DevOps practices. Containers and container orchestration engines have been identified as fundamental building blocks in the realisation of an efficient development strategy and pipeline. This helps in ensuring consistency and continuity between the different environments, starting from the local development machines up to the integration, staging and production environments. Moreover, as the SKA telescopes are being built, there is a need for a completely virtualized integration and testing infrastructure for the various software components being developed. A virtual integration facility is a precious asset for the project and it will enable software development and maintenance even in the absence of the telescope. In this context the necessity to deploy a complete Tango based control system in a containerised environment emerged. As a first step in this direction, the SKA Organisation has defined a set of standards to be adopted when approaching container related technologies. To avoid vendor lock-in and leverage Open Container Initiative (OCI) compliant containers, a set of standards, conventions and guidelines for building, integrating and maintaining container technologies has been published [27]. In adherence with the guidelines a set of container images have been defined, based on the Docker technology, but in compliance with the OCI standards [28]:

- tango-dependencies: a base image containing Tango preferred version of ZeroMQ plus the preferred, patched version of OmniORB
- tango-db: a MariaDB image including Tango database schema. Data is stored separately in a volume
- tango-cpp: core C++ Tango libraries and applications
- tango-pogo: image for running Pogo and displaying Pogo help. Pogo output can be persisted to a docker volume or to the host machine
- tango-java: as per tango-cpp, plus Java applications and bindings
- tango-python: extends tango-cpp, adding PyTango Python bindings
- tango-itango: itango, a Python shell for interactive TANGO sessions
- tango-rest: an image containing tango-rest, which acts as a REST proxy to a Tango system
- tango-dsconfig: an image containing the dsconfig tool and related dependencies

These images can be orchestrated and used as the basis for developing a complete Tango based control system. The SKA Organisation has defined a set of guidelines [29] that can be followed so to orchestrate containers based on Cloud Native Computing Foundation practices and tools. At the time of this writing a first SKA integration environment is defined and deployed accordingly to those standards using Helm charts into a Kubernetes based environment.

Further investigation is needed in order to understand the trade-offs of container adoption, there are general expectations that these aspects will evolve rapidly during the early stages of SKA construction. Optimising the local development environment for developers and easing the development experience is one of the main aspects. Also, some investigation is in progress in order to understand the performance aspects of a container based Tango architecture: how to best map PODs, containers, device servers and devices is still unclear and scalability tests will be executed in the next future to better explore this space.

CONCLUSION

The Tango Controls Collaboration has again proved to essential in ensuring the continued maintenance and development of the Tango kernel over the last two years. The quality of the Long Term Support Tango kernel libraries has improved thanks to new kernel developers with strong C++, Python and Java skills joining the core development team. An ambitious project to capture the functionality of the LTS version in RFCs has been started. Developments in the area of web technologies are very active and continue to improve. Documentation has been improved thanks to the organisation of a Write-the-Doc camp. Continuous Integration has finally been addressed for all major platforms on Linux and Windows so that now the main libraries and bindings are available for both platforms. The adoption of Docker for testing is being generalised. New members in the community are playing an important role in maintaining the level of community activity high. Tango meetings and code camps remain an essential part of the community activity. The contribution of commercial companies has been essential in all the kernel activities. Without them Tango would not have advanced to the current level of stability and support.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the Tango Controls Collaboration for funding a number of the developments described here as well as the Tango Controls Community for bug reports, fixes, suggestions for new features and contributions.

REFERENCES

[1] R. Bourtembourg *et al.*, "TANGO Kernel Development Status", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 27–33. doi:10.18429/JACoW-ICALEPCS2017-MOBPL02

[2] https://github.com/tango-controls/cppTango/releases/tag/9.3.3

[3] https://github.com/tango-controls/cppTango/issues/365

[4] https://github.com/tango-controls/pytango/releases/tag/v9.3.1

[5] https://www.python.org/dev/peps/pep-0373

[6] https://docs.python.org/3/library/asyncio.html

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA058

[7] https://pypi.org

[8] https://www.appveyor.com

[9] https://en.wikipedia.org/wiki/Syslog

[10] https://www.elastic.co/products/elastic-stack

[11] https://bintray.com/tango-controls/jtango

[12] https://repo.maven.apache.org/maven2/org/tango-controls/JTangoServer

[13] https://openjdk.java.net/jeps/320

[14] https://www.jacorb.org

[15] https://github.com/pybind/pybind11

[16] https://pybind11.readthedocs.io/en/stable/basics.html

[17] I. Khokhriakov, *Exporting Tango Controls events to HTTP using SSE*, 33rd Tango Collaboration Meeting, 2019, DESY, Hamburg, Germany.

[18] github.com/tango-controls/waltz/releases/v0.7.3

[19] I. Khokhriakov *et al.*, “Waltz – a platform for Tango Controls web applications”, presented at the 17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS’19), New York, NY, USA, Oct. 2019, paper WESH3003.

[20] https://github.com/tango-controls/tango-v10-design-doc

[21] https://github.com/tango-controls/rfc

[22] https://github.com/unprotocols/rfc

[23] https://travis-ci.org

[24] https://bintray.com/tango-controls/debian/cppTango

[25] https://github.com/tango-controls/tango-cs-docker/pull/9

[26] https://tango-controls.readthedocs.io/en/latest/installation/amazon-cloud.html#tangobox-9-3-ami

[27] http://developer.skatelescope.org/en/latest/development/containerisation-standards.html

[28] https://gitlab.com/ska-telescope/ska-docker

[29] http://developer.skatelescope.org/en/latest/development/orchestration-guidelines.html