ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

THE TANGO CONTROLS COLLABORATION STATUS IN 2021

A. Götz, R. Bourtembourg, D. Lacoste, N.Leclercq, ESRF, Grenoble, France
S.Rubio, C. Pascual-Izarra, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
V. Hardion, B.Bertrand, MAXIV Sweden, Lund, Sweden
L. Pivetta, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
P.P. Goryl, M. Liszcz, S2Innovation, Kraków, Poland
A.F. Joubert, J.Venter, SARAO, Cape Town, South Africa
G. Abeille, SOLEIL, Gif-sur-Yvette, France
T. Braun, Byte Physics, Berlin, Germany
G. Brandl, Forschungszentrum Jülich, Garching, Germany

Abstract

The Tango Controls collaboration has continued to grow since ICALEPCS 2019. Multiple new releases were made of the stable release V9. The new versions include support for new compiler versions, new features and bug fixes. The collaboration has adopted a sustainable approach to kernel development to cope with changes in the community. New projects have adopted Tango Controls while others have completed commissioning of challenging new facilities. This paper will present the status of the Tango-Controls collaboration since 2019 and how it is helping new and old sites to maintain a modern control system.

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.

The main objectives of kernel developments for Tango Controls since 2017 has been to consolidate 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 but do not break compatibility with V9 release, improve the web development platform, continue to improve the documentation and website, and prepare the next major release of Tango (V10). This paper summarises these developments.

COLLABORATION

The Tango Controls Collaboration is in charge of ensuring the sustainability of Tango Controls. It currently has 11 members who contribute financially and in-kind to the maintenance of Tango Controls as a modern reliable controls toolkit for small and large facilities. In addition to the collaboration members a number of individuals and some companies contribute new developments to the ecosystem e.g. see section Rust binding below.

The Tango Controls Collaboration contract has been in operation since 5 years. The initial collaboration contract ended in 2020. All partners agreed the collaboration was fulfilling an essential role for the sustainability of Tango and should be continued for another 5 years. All partners signed the new contract running from 2021 to 2025. The new contract maintains the objectives and missions of the previous one i.e. all members provide the same financial contribution to maintaining the kernel, while some partners contribute in-kind resources too. The ESRF is in charge of sub-contacting tasks on behalf of the other members. A major change in sub-contracting took place in 2021 with a Call For Tender for sub-contractors who could provide services to Tango for the next 3 to 5 years. After a selection process 4 companies were chosen based on their competence and knowledge of the Tango kernel. Setting up contracts for 3-5 years will help ensure the sustainability of Tango controls. The new collaboration contract foresees a rotation of the role of coordinator amongst the members every year.

New Projects

New projects continue to adopt Tango as their controls toolkit. Some examples of major new projects are the LOFAR 2.0 project (see [1]), the JINR 200 MeV LINAC (see [2]), the PEPC plasma electrode Pockels cell (see [3]) to mention a few. Some of the large projects which are based on Tango have completed successfully, for example the ESRF-EBS, the first 4th generation storage ring (see [4] and [5]). Other large projects like the SKA are ramping up to full speed and are already well advanced in their developments and will soon start construction (see [6]). A number of other 4th generation storage rings which will be based on Tango (e.g. ELETTRA, SOLEIL and CELLS) are in the planning phase. The above projects are only a small subset of projects using Tango: due to the way open source code can be downloaded by everyone without registering not all projects are declared or known to the community. They illustrate how vibrant the Tango Controls community is and the strong need to sustain and continue developing the Tango Controls toolkit for the coming decades.

KERNEL DEVELOPMENT

Migrating from GitHub to GitLab

The tango-controls GitHub organization [7] had been used since 2016 for hosting the various tango source code repositories as well as for managing issues and code review. The Continuous Integration was based on Travis-CI [8], also integrated in GitHub. A change in the terms of use of Travis-CI in December 2020 prompted the migration of the whole tango-controls organization to the GitLab platform. GitLab was found better suited to the open source nature of the Tango collaboration, offered more support for Continuous Integration and the fact that its Open Source version can be installed on-premises (as it is the case in many of the facilities involved in the Tango Collaboration). GitLab therefore offers better protection against potential vendor lock-in. The GitLab.com organisation graciously accepted to host the tango-controls project with gold status free of charge. Gold status allows up to 100 seats for developers of which currently 62 are in use.

The migration is being done on a per-project basis (at the moment of writing, 49 have been migrated, out of 67 originally in the tango-controls organization), and the main code review, continuous integration and issue handling is already done in GitLab.com. In terms of implementation, the migration has been relatively simple thanks to the automated import of GitHub projects provided by GitLab. The only aspects that needed some manual intervention were adapting the CI configuration from Travis-CI to GitLab-CI and the coordination to ensure that the contributors had linked their GitLab and GitHub accounts in order to preserve the cross-references and contribution statistics.

See paper [9] for more details.

C++ Core Library

Part of the TANGO C++ core library development is being subcontracted to Byte Physics and S2Innovation for the TANGO collaboration. The Tango C++ Kernel library cppTango [10] is actively maintained with the excellent expert help of these companies.

Since the last ICALEPCS conference, cppTango 9.3.4 [11] has been released and a new Tango Source Distribution Release [12] has been prepared. This new cppTango release provides several bug fixes for race conditions, memory leaks, issues related to events, compilation warnings and better support of recent cppzmq versions [13]. It is fully ABI and API compatible with the previous 9.3.x cppTango releases. This version still does not require C++11 support from the compiler, allowing it to be built on older platforms, including Debian 7.

Since the introduction of the TANGO 9 Long Term Support, the Tango C++ Kernel developers are working on improving the code quality by refactoring the source code to make it easier to maintain. This work is being done in the cppTango repository main branch from which cppTango 9.4 will be released at some point. cppTango 9.4 will not be binary compatible with cppTango 9.3.x but will still be API compatible with cppTango 9.3.x versions. The main development branch requires at least C++14 and will be compatible with C++17 and C++20 . This branch is being automatically compiled and tested with GitLab CI on Ubuntu 20.04 LTS, Debian 9, 10, and 11 platforms as well as using the latest releases of GCC and Clang. The migration to GitLab was an opportunity to replace SonarCloud services with CI jobs using Clang Static Analyzer [14] and clang-tidy [15] to detect code quality issues. Also a new CI job that generates a code coverage report was added. Additionally, some steps were taken to improve the memory safety and the overall stability of the C++ core library. As a result a set of sanitizer CI jobs, including address [16], undefined behavior [17] and thread sanitizers [18], was enabled for the main branch.

Python Core Library

Python is still enjoying good popularity in the Tango Controls community. 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 9.3.3 has been released in December 2020 [19]. This marks only the second release since the previous review at ICALEPCS 2019 [20], with the 9.3.4 release is expected this year. Fourteen contributors have been working on the project producing more than 150 commits, in total. PyTango 9.3.3 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.

PyTango fully supports both Python 2 and Python 3 language versions, although Python 2 is nowadays officially obsolete and its usage have been already discouraged for new developments. Having compatibility with both language versions has helped during the transition to Python 3 of legacy python-based projects. The biggest frameworks using PyTango, like Taurus and Sardana, are already running in production using Python 3. Older Python tools and device servers (fandango, pytangoarchiving, panic) are currently in the process of migration using GitLab CI/CD for validating the migrated code. Unfortunately, the slow pace at which institutes are able to transition all their code to python 3 means that PyTango needs to maintain Python 2 supports for some years still.

The long running port [20] of the C++ extension code from Boost [21] to pybind11 [22] has not progressed significantly. There were few resources to allocate to this work. As there is no major risk of Boost's Python support becoming obsolete, the Tango Steering Committee has decided to pause this effort.

Future work includes making PyTango available as a binary wheel for Linux. This will significantly simplify installation for users.

Java Core Library

Java core library (aka JTango), which is 100% Java, relies on Jacorb [23] and JeroMQ [24], is now built upon Java 8

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

and Java 11. The latest developments have focused on its quality; by fixing bugs, replacing deprecated Java code with Java 8 code, and adding automatic unit and integration tests. JTango artifacts were deployed automatically to Bintray [25] and manually on Maven Central [26] repositories. Since Bintray has announced its shutdown [27] and we were in the process of migrating to GitLab, the CI/CD pipeline were completely refurbished, to optimise its execution time and to automatically deploy artifacts to Maven Central repository (see details in paper [9] at this conference).

POGO

Pogo is the Tango code generator for device servers. It allows users to define a Tango class model through its graphical user interface and save it in an .xmi file. Starting from this Tango class model, Pogo is able to generate a device server skeleton in C++, Java or Python and the relevant HTML documentation. The code generation part is based on EMF (Eclipse Model Framework) associated with Xtext and Xtend classes [28]. Pogo source code is available on GitLab [29], and binary distribution is maintained on maven central [30].

In the past years Pogo has continued to be improved. Python support greatly evolved, two flavors of python device servers can be generated. With the introduction of PythonHL, device servers code is now more concise, more readable, easier to maintain, and faster to develop.

Latest developments focus on improving Pogo for automation. Current development workflows rely more and more on CI/CD and Pogo needs to integrate and keep up to date with the latest strategies. This has lead to:

- A ready to use docker image is now available. It will speeds up deployment strategy and can be used for automatic testing.
- Improved cmake integration. Modern cmake is the de facto standard for building C++ projects, it simplifies building and managing dependencies.

PACKAGING

RPM

Tango RPM packages used to be built internally by MAX IV and provided to the community via the MAX IV's repository [31]. The Tango SPEC file repository [32] has been moved to the tango-controls group on GitLab. Building is now performed using Copr [33], a build system and infrastructure provided by Fedora. Packages are produced for CentOS 7, CentOS 8, Fedora 32 and 33. The latest version of Tango can be installed on those distributions directly from the Copr repository. RPMs are still available from the MAX IV's repository as well.

Conda

Conda [34] is an open-source solution for dependency and environment management for any language. It is cross-platform and runs on Windows, macOS and Linux. Some packages (pytango, itango, tango-test) were already

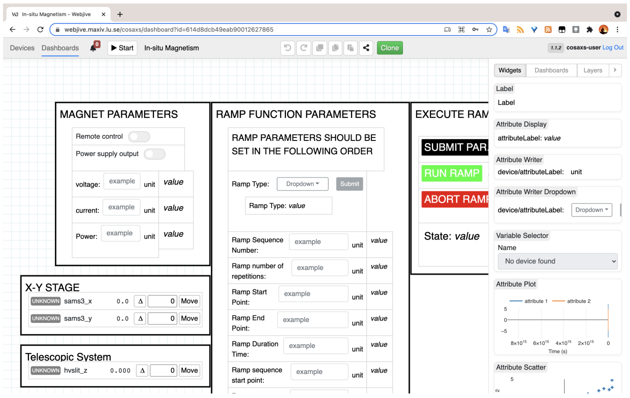


Figure 1: Example of a web interface developed with Taranta - display of status of MAXIV magnets.

available using the tango-controls conda channel [35]. Some effort were made during this year to publish those packages and add new ones to conda-forge [36]. conda-forge is a community-led collection of recipes, build infrastructure and distributions for the conda package manager. It is very active and has become the de facto channel when using conda. Using conda-forge has several advantages:

- existing conda-forge infrastructure makes it easy to build on all supported platforms
- it ensures compatibility with conda-forge packages by using the same build environment and baseline software versions defined in conda-forge-pinning [37]
- makes it easier for people to discover tango packages

The current Tango packages available on conda-forge are: tango-idl, cpptango, tango-database, tango-admin, tango-test, pytango and itango. This move was also beneficial to the wider Tango community allowing software like Sardana, which depends on pytango, to be available on conda-forge. Only Linux is currently supported but work for publishing Windows packages is planned.

WEB TOOLKIT

During the last 2 years the pandemic situation has demonstrated the importance of using web technology in the Tango Control System. There are currently mainstream 2 solutions maintained by the community members, Taranta [38] and Waltz. Taranta (shown in Fig. 1) is now an official tool of the Tango ecosystem and has moved to the tango-control GitLab repository. Following the example of the Tango REST API which has a formal specification (see [39]), a similar project is planned for the specification of the GraphQL Tango gateway in order to allow different back-end implementations (see [40] for example) to connect to multiple front-end applications using GraphQL.

New initiatives have chosen the ubiquitous web browser for their User Interface (UI) like the Solaris Synchrotron [41] who developed the interlock and machine status UI with the Vue.js [42] framework on top of TangoGQL [43].

In the meantime S2Innovation has developed IC@MS a new web UI for the standard Tango Alarm system (see [44]).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

```
use tango_client::*;

let mut dev = DeviceProxy::new("tango://localhost:10000/sys/tg_test/1")?;
let instr = CommandData::from_str("This is a minimal Tango test client.");
let argout = dev.command_inout("DevString", instr)?;
println!("Command exec result: {}", argout.into_string()?);
```

Figure 2: Tango Rust binding example code.

This is an example how the traditional Tango tools are getting a lifting with modern technology.

BINDINGS

Rust Binding

Rust [45] is a fairly new systems programming language. Because it puts a focus on efficiency while ensuring memory and thread safety at compile time, it has the potential to be used, also in the scientific sector, where code is currently being written both in C/C++ and Python, trying to provide the speed of the former together with the usability of the latter. [46] The Tango Rust binding [47] is a Rust library ("crate") that provides Rust code with client access to Tango devices, wrapping the C binding. So far, we have implemented a `DeviceProxy` API up to Tango 8 and rudimentary a `Database` interface. Figure 2 shows a minimal code example calling a method on a TangoTest server. The library is available on Rust's central package server crates.io [48] and requires an installation of the C++ libtango to be present on the system to build against, as well as a C compiler to compile the bundled C binding. It is currently only tested on Linux systems.

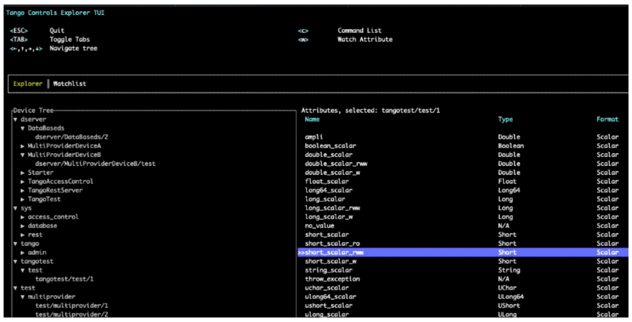
TUI

A small text based command line utility have been developed to explore running Tango devices. Devices are presented in a tree structure, from which they can be traversed and selected (see screenshot in Fig. 3). Once a device is selected, their attributes and commands can be seen.

The tool is written in Rust making use of the Rust bindings. The Rust bindings have some limitations that may cause issues when using the tool e.g. enum types are currently not supported. For more details on the tool including its limitations, installation instructions etc. see the project GitHub page [49].

COMMUNITY WORKSHOPS

The 35th Tango Community Meeting was held on September 14 and 15 2021. Despite the pandemic and the remote nature of this event, +100 people registered and +25 talks has been given. The traditional session dedicated to Projects Status allowed some major Tango-based projects to report on their progress. Among them, were SKAO, the world's largest radio telescope under construction and the ESRF-EBS, the first-of-a-kind fourth-generation



```
Tango Controls Explorer TUI

<ESC>        Quit                                   <C>     Command List
<TAB>        Toggle Tabs                            <W>     Watch Attribute
<←,↑,→,↓>    Navigate tree

 Explorer | Watchlist

Device Tree                                   Attributes, selected: tangotest/test/1
▼ dserver                                      Name                      Type      Format
  ▼ DataBaseds
      dserver/DataBaseds/2                     ampli                     Double    Scalar
  ▶ MultiProviderDeviceA                       boolean_scalar            Boolean   Scalar
  ▼ MultiProviderDeviceB                       double_scalar             Double    Scalar
      dserver/MultiProviderDeviceB/test        double_scalar_rww         Double    Scalar
  ▶ Starter                                    double_scalar_w           Double    Scalar
  ▶ TangoAccessControl                         float_scalar              Float     Scalar
  ▶ TangoRestServer                            long64_scalar             Long64    Scalar
  ▶ TangoTest                                  long_scalar               Long      Scalar
▼ sys                                          long_scalar_rww           Long      Scalar
  ▶ access_control                             long_scalar_w             Long      Scalar
  ▶ database                                   no_value                  N/A       Scalar
  ▶ rest                                       short_scalar              Short     Scalar
▼ tango                                        short_scalar_ro           Short     Scalar
  ▶ admin                                     >>short_scalar_rww         Short     Scalar
▼ tangotest                                    short_scalar_w            Short     Scalar
  ▼ test                                       string_scalar             String    Scalar
      tangotest/test/1                         throw_exception           N/A       Scalar
▼ test                                         uchar_scalar              UChar     Scalar
  ▼ multiprovider                              ulong64_scalar            ULong64   Scalar
      test/multiprovider/1                     ushort_scalar             UShort    Scalar
      test/multiprovider/2                     ulong_scalar              ULong     Scalar
```

Figure 3: TUI - text based generic client written in Rust.

high-energy synchrotron. The technical sessions related to the Tango ecosystem provided the attendees with latest news of the kernels, the tools and the services around Tango. PyTango - the Python binding based on the C++ kernel - remains the most popular platform for both server and clients development, especially for newcomers. The trend towards the usage of web-based solutions for controls systems GUIs continues, re-enforced by the need for remote controls during the COVID-19 pandemic. The Tango Community Meeting was also an good opportunity to introduce the subcontractors who were selected during the Call for Tender managed by the ESRF.

In addition to regular Tango Kernel Meetings, Tango Kernel Training Sessions and the yearly Tango Community Meeting, the HDB++ Archiving collaboration meetings are now scheduled every two months to boost the collaboration between institutes. These regular follow-ups of the project have enabled the continuous improvement of the main library, the testing and evaluation of the HDB++ core libraries over multiple database engines (PostgreSQL, MySQL, MariaDB, TimeScaleDb, ...) [50] as well as the development of a common AbstractReader class for data extraction.

TANGO V10

Tango V10 is the code name of the next long term version of Tango which should bring a change in the communication protocol. Since the backward compatibility is one of the main attractions for Tango, the community has decided to collectively understand the consequence of this major change. The first step was to define what is the essence of Tango in the form of RFCs.

The Tango Request For Comment RFC project [51] specifies the data model of Tango, the expected behaviour of the Tango elements and the communication between 2 components, as it is today with version 9. The project of writing these specifications has involved all representatives of the Tango consortium in such a way that the knowledge is built collectively. The draft of the specifications is almost complete before being compared to the existing implementation. At the current state, it covers the essentials of the Tango Model with 12 specifications.

The main communication protocol is the most complex to specify as a large part of it is given by the CORBA implementation. A careful review of the cppTango library allows extracting the main principle, although separate the abstraction from the implementation is a difficult challenge.

The RFCs are published on ReadTheDoc (see [52]). These documents have status *draft*.

The final work in progress concerns RFC-10/The Request-Reply Protocol. The RFC-13/Publisher-Subscriber protocol implementation with ZMQ is waiting for review. Three other RFCs are planned to be written if needed.

The Tango Community plans to organise a workshop for the final edition of the RFCs and plan the next steps.

The Community will follow COSS and use the RFCs as a base for building a prototype of Tango V10. This will verify the specification for completeness and any ambiguity. When a new version of Tango Controls is ready, the documents will be marked as stable [53].

TRACING VIA LOGS

The SKA project developed a Python library [54] that generates logs with tracing information. A log entry is created upon the entry and exit of the context manager.

The main items in the log messages are:

transaction ID Tango device command arguments are generally type string (JSON). The JSON may include a common transaction ID that is passed along as part of the arguments. By searching for this ID in the logs, the execution path can be traced between devices.

Enter|Exit name Whether this is the start or end of the context, and an arbitrary name for the log pair.

marker A random string used to match "Enter" and "Exit" log entries.

For more details see paper [55]

CONCLUSION

The Tango Controls Collaboration has again proved to essential in ensuring the continued maintenance and development of the Tango kernel for the coming years and decades thanks to securing high quality sub-contractors on multi-annual contracts. The high quality of the in-kind contributions continue to contribute to the successful development of Tango Controls. Major new projects are based on Tango which puts a strong requirement on sustainability for the coming decades. A new binding is now available for Rust. Rust is a promising language for the future which addresses the problems of memory leaks which traditional languages like C++, C and Python suffer from. The web developments have been pushed ahead with the wide-spread adoption of the Tango GraphQL protocol. The next step will be to fix the specification of Tango GraphQL to ensure compatibility between multiple implementations. The collaboration is setup for the next 5 years and is open to new members who rely on Tango or plan to use Tango for their facilities and products.

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] T. Juerges, J. Mol, T. Snijder "LOFAR2.0: Station Control Upgrade", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOAR03, this conference.

[2] A. Trifonov, M. Gostkin, V. Kobets, M. Nozdrin, A. Zhemchugov, P. Zhuravlyov "The Control System of the Linac-200 Electron Accelerator at JINR", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper TUAR03, this conference.

[3] L. Li, J. Luo, Z. Ni "Fast Creation of Control and Monitor Graphical User Interface for Pepc of Laser Fusion Facility Based on Icsff", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper THPV007, this conference.

[4] S.M. White *et al.*, "Commissioning and Restart of ESRF-EBS", in *Proc. IPAC'21*, Campinas, SP, Brazil, May 2021, pp. 1–6. doi:10.18429/JACoW-IPAC2021-MOXA01

[5] https://indico.cells.es/event/619/contributions/1480/attachments/1034/1691/ESRF-EBS-Status-Tango-Community-Meeting-Sep.2021.pptx

[6] J. Santander-Vela, M. Bartolini, M. Miccolis, N. Rees "From SKA to SKAO: Early Progress in the SKAO Construction", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOAL03, this conference.

[7] https://github.com/tango-controls

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

[9] M. Liszcz et al "Migration of Tango Controls Source Code Repositories", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOPV034, this conference.

[10] https://github.com/tango-controls/cppTango

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEAR01

[11] https://github.com/tango-controls/cppTango/releases/tag/9.3.4

[12] https://github.com/tango-controls/TangoSourceDistribution/releases/tag/9.3.4

[13] https://gitlab.com/tango-controls/cppTango/-/blob/9.3.4/CHANGELOG.md

[14] https://clang.llvm.org/docs/ClangStaticAnalyzer.html

[15] https://clang.llvm.org/extra/clang-tidy

[16] https://clang.llvm.org/docs/AddressSanitizer.html

[17] https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html

[18] https://clang.llvm.org/docs/ThreadSanitizer.html

[19] https://gitlab.com/tango-controls/pytango/-/releases/v9.3.3

[20] A. Götz, G. Abeillé, M. Bartolini, R. Bourtembourg, T. Braun, J.M. Chaize, *et al.*, "State of the Tango Controls Kernel Development in 2019", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1234–1239. doi:10.18429/JACoW-ICALEPCS2019-WEPHA058

[21] https://www.boost.org/doc/libs/1_76_0/libs/python/doc/html/index.html

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

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

[24] https://github.com/zeromq/jeromq

[25] https://bintray.com/.

[26] https://search.maven.org/artifact/org.tango-controls/JTangoServer

[27] https://jfrog.com/blog/into-the-sunset-bintray-jcenter-gocenter-and-chartcenter/.

[28] https://tango-controls.readthedocs.io/en/latest/tools-and-extensions/built-in/pogo/index.html

[29] https://gitlab.com/tango-controls/pogo

[30] https://mvnrepository.com/artifact/org.tango.tools.pogo.gui/Pogo

[31] http://pubrepo.maxiv.lu.se/rpm/el7/x86_64/.

[32] https://gitlab.com/tango-controls/tango-spec

[33] https://copr.fedorainfracloud.org/coprs/g/tango-controls/tango/.

[34] https://docs.conda.io

[35] https://anaconda.org/tango-controls

[36] The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem, doi:10.5281/zenodo.4774216

[37] https://github.com/conda-forge/conda-forge-pinning-feedstock

[38] M. Canzari *et al.*, "Satisfying wishes for SKA engineers: how Taranta suite meets users' needs", in *Software and Cyberinfrastructure for Astronomy VI*, SPIE, 2020, pp. 700–706. doi:10.1117/12.2562585

[39] https://gitlab.com/tango-controls/rest-api

[40] J-L.Pons, "TangoGraphQL: A GraphQL binding for Tango control system Web-based applications", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper MOPV025, this conference.

[41] https://indico.cells.es/event/619/contributions/1462/attachments/1044/1728/Vue_TangoGQL_SOLARIS.pdf

[42] https://vuejs.org/.

[43] https://gitlab.com/tango-controls/web/tangogql

[44] https://indico.cells.es/event/619/contributions/1463/attachments/1039/1708/IC@CMSStatusTangoCommunityMeeting.pdf

[45] https://www.rust-lang.org/.

[46] Jeffrey M. Perkel, "Why scientists are turning to Rust", *Nature*, vol. 588, pp. 185–186, 2020. doi:10.1038/d41586-020-03382-2

[47] https://gitlab.com/tango-controls/tango-rs

[48] https://crates.io/crates/tango-client

[49] https://github.com/SKAJohanVenter/tango-controls-tui

[50] R. Bourtembourg, G. Cuní, M. Di Carlo, G.A. Fatkin, S. James, L. Pivetta, *et al.*, "Pushing the Limits of Tango Archiving System using PostgreSQL and Time Series Databases", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1116–1121. doi:10.18429/JACoW-ICALEPCS2019-WEPHA020

[51] https://gitlab.com/tango-controls/rfc

[52] https://tango-controls.readthedocs.io/projects/rfc/en/latest/.

[53] P.P. Goryl, V. Hardion *et al.*, "Tango Controls RFCs", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper TUBL03, this conference.

[54] https://gitlab.com/ska-telescope/ska-ser-log-transactions

[55] S.N. Twum *et al.*, "Implementing an Event Tracing Solution With Consistently Formatted Logs for the SKA Telescope Control System", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2021), Shanghai, China, October 2021, paper TUBL02, this conference.