ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

THE TANGO CONTROLS COLLABORATION STATUS IN 2023

T. Juerges, SKA Observatory, Jodrell Bank, United Kingdom
R. Bourtembourg, A. Götz, D. Lacoste, N.Leclercq, ESRF, Grenoble, France
G. Cuni, C. Pascual-Izarra, S. Rubio-Manrique, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
B. Bertrand, V. Hardion, A.F. Joubert, MAXIV Sweden, Lund, Sweden
Y. Matveev, DESY, Hamburg, Germany
L. Pivetta, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
T. Noga, M. Nabywaniec, L. Zytniak, S2Innovation, Kraków, Poland
G. Abeillé, SOLEIL, Gif-sur-Yvette, France
T. Braun, Byte Physics e. K., Annaburg, Germany
R. Auger-Williams, T. Ives, Observatory Sciences Ltd, St Ives, United Kingdom

Abstract

Since 2021 the Tango Controls collaboration has improved and optimised its efforts in many areas. Not only have Special Interest Group meetings (SIGs) been introduced to speed up the adoption of new technologies or improvements, the kernel has switched to a fixed six-month release cycle for quicker adoption of stable kernel versions by the community. CI/CD provides early feedback on test failures and compatibility issues. Major code refactoring allowed for a much more efficient use of developer resources. Relevant bug fixes, improvements and new features are now adopted at a much higher rate than ever before. The community participation has also noticeably improved. cppTango switched to C++14 and the logging system is undergoing a major refactoring. Among many new features and tools is jupyTango: Jupyter Notebooks on Tango Controls steroids. PyTango is now easy to install via binary wheels, old Python versions are no longer supported, the build-system is switching to CMake, and releases are now made much closer to stable cppTango releases.

INTRODUCTION

In 1998, the European Synchrotron Radiation Facility (ESRF) submitted a paper about TANGO, a control system framework based on the paradigm of distributed objects, to ICALEPCS 1999 [1]. Today, Tango is much more than a piece of software. First, it is a well organized, very friendly and amazingly fruitful collaboration gathering eleven major institutes [2]. It is also the place of constant development, enhancement, refactoring and innovation [3].

The success of the Tango collaboration relies on its organisation, the members and developers and also on the underlying contract that governs it. The financial contribution by the core members allows the collaboration - represented by its steering committee - to oversee its own budget and finance events and technical subcontracting. The latter definitively boosts the development of Tango. It is now much more than a just framework. It is a rich and very active software ecosystem in constant evolution. The original technology still exists, but the core developers are preparing the future on a daily basis. Tango constantly improves and tries to benefit from the latest technical developments while maintaining a strong backward compatibility for its users.

The present paper provides the reader with the latest news from the Tango Controls collaboration. It notably proposes a wide overview of both the organisational and the technical activity around Tango.

CPPTANGO

Switch to Fixed Release Cycles

In the past years official cppTango releases were infrequent and usually years apart. These long release cycles resulted in a number of issues:

- Users were not easily convinced to update, because the old release had "somehow worked" for "n years" and "we know the bugs". Facilities that run in production mode are very conservative when it comes to updating core production software. They do not deploy a new release just because it is the latest version and has more features. This leads to the more serious problem of outdated installations that at some point cannot be updated anymore at all.
- The change set of a cppTango release were usually quite big, because many bug fixes, new features and additions made it into a single release. Users would see this as a mountain too big too climb, because "it has changed everywhere". An update simply appeared to be too expensive and this contributed again to production systems running outdated cppTango versions.
- The amount of changes made it hard for the users to find the useful features that would make their lives much easier.
- Packaging of cppTango was turned into a time consuming and thus expensive problem, because dependencies had already moved on a long time ago. This re-created dependency-hell for every release and costs the cppTango team every time dearly. Distributions dropped cppTango dependencies and suddenly a substitute had to be found, investigated and adopted before a new release. This caused an incredible amount of additional pressure just before a release.
- Long release cycles bore the risk of Tango Controls

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

packages being out of sync with distributions that also have longer release cycles. This could easily lead to a previous release of Tango Controls ending up in a new distribution release. An example was Debian's then current release "Bullseye". That it contained packages of the then current Tango Controls release was pure coincidence and had not been planned. Shorter release cycles, e.g. every six months, could guarantee that even distributions which have short release cycles of six months (Ubuntu) would contain Tango Controls packages that are in the worst case six months minus a couple of days old.

- It took a very long time for bug fixes to make it into a user's production system. That is unless they clone/build/deploy/cross fingers the repository which worked for some but for others it does not.
- Long release cycles made it necessary to maintain backport branches for a very long time. Tango Developers had to scatter their limited time on too many issues. More branches that needed to be maintained meant more work and that meant less efficient use of resources. The direct impact was that new cppTango releases, which would contain major refactoring, major rewrites or would break old APIs or ABIs, were moved to a later date in the future again and again.
- Other Tango parts would then be forced to also maintain a backports branch for backwards compatibility with cppTango. The resources there were also partially bound to maintain old code branches.

In February 2022 a proposal for cppTango was made to set up a fixed release cycle of six months. Releases would be made, regardless how many bugs were fixed or how many new features were implemented. The plan was to have everything ready on the release date. The expected benefits were that cppTango developers could better focus on relevant bug fixes for the upcoming release, better plan new features and as a result have a roadmap for future releases.

The proposal posed the risk that users would still not update more often but the gains on the side of the cppTango developers were viewed to be worth the change.

In an early April 2022 it was decided at a Tango Controls kernel meeting, that a six month release cycle would be adopted. Since the release of cppTango 9.4.0 on 2022-10-02, the first release in the new six month release cycle, the team has stayed true on the path and made three more releases (9.3.6, 9.4.1, 9.4.2) with a fourth one scheduled for 2023-10-09, with an initial release date 2023-10-02 but delayed by one week.

So far the cppTango team is very satisfied with the switch to fixed release cycles. Having a clear road map has indeed allowed to be able to focus the resources much better. There were a couple of bumps in the road that exposed weaknesses in the current testing which are now being addressed. Those weaknesses could not have been made visible if years would pass between releases, because workarounds would be found and problems not reported back to the cppTango team. This directly contributes to the improvement of the quality of the cppTango software.

Refactoring/Templatisation

CppTango development's rate increased in all directions in order to provide a better software for our users. Be it with newer features, or bug fixes, but we also dedicated a lot of effort into code refactoring. Because a smaller and more maintainable code base is a way for us to dedicate more time to develop the next feature, we have to work now on improving our code for the future. In this effort we merged a branch that refactored the code to use templates in some places that lead to 7000 less lines of code for the exact same functionality. And another one coming could further remove another 4000 lines of code. So much less to worry about as a developer.

Observability

The Tango logging service currently adopts the log4cpp [4] design - which is based on the concept of a "logger" routing the log to different targets through "appenders". For the Tango 10 series, it has been decided to extend the Tango logging service towards an **observability** one. Beyond the buzzword coming for the world of microservices, the aim is to instrument the Tango kernel to offer OpenTelemetry [5] like features for traces, log and metrics. In other words, the idea is to propagate some **context information** along the distributed call stack so that the "observation data" emitted by the involved (and totally independent) components can easily be associated to a uniquely identified transaction. Adopting OpenTelemetry - and the underlying W3C standard for context propagation - offers the opportunity to use existing tools and services to store, retrieve and display the information. However, in order to relax the level of dependency on the OpenTelemetry API and SDKs, the Tango observability features will be available through a dedicated API hiding the implementation details. The first Tango 10 release will first focus on tracing and profiling.

macOS Support

Full macOS support has been added with cppTango 9.4.0. This gives software engineers the freedom to choose a development platform (Linux, macOS, Windows) that they are used to. Enabling tests to run on macOS is still a work in progress. It is not expected that production systems will run cppTango Device Servers on macOS.

PYTANGO

Python's worldwide popularity is excellent, and still on an upward trend [6]. This makes the PyTango Python-to-C++ binding a very important part of the Tango Controls ecosystem. PyTango provides a high-level application programming interface which makes developing servers and clients extremely easy.

Release Process

PyTango 9.4.2 was released in July 2023 [7]. Since the previous review at ICALEPCS 2021 [8], there have been 6 re-

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

leases, with more than 144 merge requests and 615 commits. There was one major release, 9.4.0, where Tango C++ library support was switched to 9.4.x releases; NumPy became a hard requirement and Python 2 support was removed.

The PyTango maintainers have committed to release a new PyTango version at least twice per year, and within 1 month of cppTango releases.

The release process now includes one or more release candidates. These allow downstream projects to test for regressions before the final release. We try to test with the continuous integration suites of some large projects using PyTango. It has been beneficial, but it has also been a challenge to get feedback from downstream projects timeously.

PyTango installation is now very simple, with a wide variety of binary packages available. See the sections on Conda and PyTango wheels below.

Support Policy

Starting from 9.4.2, PyTango implements a Python and NumPy version support policy: any new release of PyTango will support all minor versions of Python released 42 months (but at minimum the two latest minor versions) prior to PyTango release date and all minor versions of NumPy released at that date that meet the requirements in "oldest-supported-numpy" [9]. The policy is partly based on NumPy's own policy [10]. As Python minor versions are released annually, this means that PyTango will drop support for the oldest minor Python version every year, and also gain support for a new minor version every year.

This resulted in PyTango 9.4.2 being officially released for Python 3.9, 3.10 and 3.11. Such a policy may cause users to go to considerable lengths to update their systems to newer Python versions, but the work will have to be done anyway, the question is how long it will be delayed. At the same time, this opens up the possibility of adding new features to PyTango.

Examples of planned features which are now possible:

- New cmake-based build system (requires at least Python 3.8). This should simplify PyTango installation in case if there are not corresponding wheels, especially on Windows.
- Declaration of attributes, properties and commands with type hints (requires at least Python 3.9). This should not only improve code readability, but should also allow the use of static code checkers.

Unlike cppTango, PyTango does not provide so-called long-term-support. While cppTango provides fixes and releases on both the 9.3.x and 9.4.x branches, PyTango only has the resources to target the most recent cppTango release.

People

There has been an increase in the number of PyTango contributors. The 2019 to 2021 releases had 13 contributors, compared to the 28 contributors for the 2021 to 2023 releases. A new maintainer has joined the team. We started regular meetings for developers, twice a month. This has improved the communication between all the developers and also provides an accountability/reminder mechanism for work in progress. The meeting schedule [11] and minutes [12] are publicly available to aid transparency and to make it easy for new people to join.

JTANGO

The Java core library relies on open source libraries Jacorb [13] and JeroMQ [14]; it is available in the de-facto Java standard Maven Central repository [15]. While the server API is used in a few institutes of the Tango collaboration, generic tools such as Jive, ATKPanel or Astor rely on the client API of JTango. JTango is quite stable, recent releases (latest is 9.7.2) deliver bug fixes and small improvements. The road map for the next years will firstly focused on improving the quality of the client API by introducing logging that is already available on the server side (based on SLF4J [16]) and adding unit tests. Secondly, we will concentrate on implementing the new Tango v10 functionalities detailed in the paragraph "Community meetings" further down (alarm event, observability...).

TANGO DATABASE

The TangoDatabase serves as phone book and configuration storage device server for a given Tango setup. Work items in the last iterations contained the move to cmake [17] as build system, dropping the existing build scripts for automake/autotools and Visual Studio solutions. The current version now builds on Linux, Windows and MacOSX (X86_64 and AARCH64). An ongoing effort to add unit tests was also started, these are currently run under Linux with MySQL/MariaDB and cover roughly one quarter of the source code. One of the side benefits of this effort is also that it now builds much more reliable against various MySQL/MariaDB versions. Besides various bug fixes two more new noteworthy features are the switch to InnoDB as database engine, thus making the response times much smaller and the addition of an IP-based access control list.

HDB++

The HDB++ project [18] is a high performance event-driven archiving system designed and built for Tango Controls. HDB++ supports several backends, including MySQL/MariaDB, PostgreSQL, TimescaleDB, and recently SQLite, exploiting an abstraction library, named *libhdb++*, and a specific implementation library for each of the supported backends. Recent developments added the support for a new backend, based on SQLite database engine, and a Python extraction library. HDB++ developers and users meet online every two months on average to discuss issues. In 2022 an in-person SIG workshop was organised and hosted by ASTRON, with special focus on archiving and HDB++.

POGO

Pogo [19], the code generator for tango device servers is a central piece of Tango Controls. It generates the boiler

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

plate code when developing device servers to let developers focus solely on the interaction with the hardware. It can generate code for three main flavours of Tango: Python, Java and C++. Following the recent upgrade in the community both for Java tools and their deployment to Maven, and the accelerated pace of development for cppTango, Pogo was greatly improved recently, and more improvements are coming. First, as it is a tool written in Java, it followed the recent improvement in the way Java tools from the JTango community are distributed and is now available directly on Maven Central [13]. It benefited from the work done on the Java stack, as, from inception it was decided to make the deployment of Tango Controls tools via Maven central common. More details about this in the section on CI/CD. Another big step to the ease of deployment for Pogo, is that it is now available on Conda. Thanks again to a major effort from the community to bring as many tools from Tango Controls to Conda, this is now the case for Pogo too. Work was done on testing Pogo as well. As CI/CD is now used for deployment, there was also some work put on developing some basics tests for Pogo. As the codebase is not easy to work on, in the past a fix somewhere could break in other places, hence tests were needed for some time now. So far the coverage is limited but there are issues to improve upon it and a plan to test more thoroughly the code generated, be it in Python, Java, or C++. As the recent versions of cppTango simplified the cppTango's server API, it was necessary for Pogo to follow this development and let developers generate device servers using the latest version of cppTango. This was done in parallel with cppTango's development, and version 9.8 of Pogo, full compliance with cppTango 9.4 was ready at the same time that cppTango was released. However this change of API is not without consequences, and there are now support two branches for Pogo, the 9.7 branch supporting the cppTango 9.3 branch, and the 9.8 supporting the newer cppTango versions. Both branches are under active development and will receive bugfixes if needed.

Learning from cppTango's eager adoption of cmake, our code generator is currently tested with further cmake integration. The next minor release is expected to completely rethink the way code for device servers is organized, with cmake at its core. In the current setup a device server contains one or more tango classes, but they are generated in a flat hierarchy, there is no clear separation between what is a class and a server, and all the code is built into a single binary. With the forthcoming changes there will be a clear separation of the code between the server and the Tango classes. It will be possible (and in certain cases recommended) to compile the latter as libraries to be able to reuse them across multiple device servers. The code will be organized in a more classic hierarchy, with a separation between implementation files and public header. To finish, cmake will be the only build system supported for C++ projects, but the integration will be much more robust and modern than what is currently done. With the use of modern cmake, fully leveraging the power of targets, find modules and so on. Another important step expected before the end of the year is the support of Java 17. Java 8 had a long live cycle, but after it took some time to move to Java 11, the new releases seem to be adopted faster by the common distributions supported, and so Pogo need to follow. This development was delayed a bit as one of the main dependencies used struggled to get up to date, but now everything seems to be in place for Pogo to have a Java 17 release.

Pogo is having its fair share of development, and a lot of time was spent to keep it up to date. But with all of it came the major realization that to go faster and further will not be possible. Pogo uses the xtext/xtend [20] framework that relies on community support. One of the major dependencies is not maintained any more and furthermore advises not to use this kind of approach for code generation [21]. In the light of all this, development was started to completely rewrite Pogo from scratch. This rewrite will be done in Python, rely on templates to generate the codes via jinja2 [22] and will be command line only, at least for the first releases. An effort will be made to maintain compatibility with the old version, by supporting the same input files for instance, but the protected regions paradigm will not be used anymore, and this will surely break the backward compatibility. In the long run the goal is to reduce the dependency on big frameworks, drastically reduce the code base and make it cleaner and easier to extend with new features. Such as supporting other types of input files like json or yaml that could be easily crafted by hand or generated by other tools. Or add the possibility to generate code for a server in multiple languages, generating a simulated device that would let a developer test a device server right after it was generated through Pogo. All of these would prove really difficult to do with the current Pogo implementation, yet it would add tremendous value to Pogo. This rewrite will hopefully not take too much time, and Pogo will stay a major part of Tango in the future.

CI/CD

With the move to Gitlab as git forge, the tango project also gained out-of-the-box support for the builtin CI/CD infrastructure. This was quickly adopted and replaced Travis/Appveyor. With the development speeding up prior to the releases, it became quickly apparent that the limited number of freely available CI/CD minutes from Gitlab would not suffice. It was therefore decided to leverage the power of the community and host Gitlab runners at each institute. Currently four institutes provide Linux Docker runners, with more institutes currently rolling out windows runners. With these unlimited number of CI/CD minutes we are now able to test each PR of the major projects, run the tests and deploy the releases. A recent example is the automatic release creation on repository tagging of the TangoSourceDistribution and the fully automatic release package upload for the cppTango windows binaries.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

TANGOBOX

The TangoBox, which is a virtual machine image running the pre-installed and pre-built Tango environment, has been upgraded to version 10. Due to the steadily increasing demand for Python 3 integration with Tango Controls and the growing number of web applications and services connected with the control system, the newest version includes full support of Python 3 and web applications such as Taranta. Packages that are a part of the core ecosystem but are not Python-related, have also been updated to their newest released versions. Among the installed packages are:

- Tango Source Distribution along with PyTango and Tango Access Control - 9.3.5
- Taurus - 5.1.5 (including HDB++ support)
- Sardana - 3.3.5
- PANIC - 9.0.1
- Taranta - 1.3.12
- HDB++

The image also provides full support of Docker for increased local development capabilities. TangoBox is running on Ubuntu 22.04 LTS, which is the latest LTS release, and its graphical user interface is Xfce - the lightweight alternative to GNOME. The only piece of software that user needs is a virtualization engine that supports the Open Virtualization Format (.ova files), to which they should import the pre-built TangoBox virtual machine image. Everything that is inside has been pre-configured and ready to go.

TANGO CONTROLS COLLABORATION

The Tango Controls Collaboration has continued to thrive thanks to the collaboration contract which was renewed for another 5 years by all 11 partners in 2021. The subcontracting contracts were also renewed for another 2 years until 2026 with 3 companies. The support of the 3 companies is essential to ensuring that Tango continues to get regular releases and improvements. The community has been innovative in organising a new series of so-called SIG meetings on special topics (details below). The regular community meetings have been held in hybrid mode with as many attendees in person as remotely.

Community Meetings

The, now annual, community meeting offers the Tango experts and users to share the latest news on their respective activities. Since ICALEPCS'21, the 36^{th} [23] and 37^{th} [24] edition of the series have been hosted by MaxIV (Lund, Sweden) and SKAO (Jodrell Bank, UK). Each event gathered more than one hundred attendees from major institutes representing various research domains - like synchrotron radiation sources, laser facilities and observatories. Beyond the traditional status report from the institutes and the latest Tango ecosystem news, the Max-IV event demonstrated the increasing interest for web-based solutions. Taranta [25], the Tango offer for web-based dashboards, has notably reached a level of maturity that will definitely boost its adoption. The most remarkable highlight of the SKAO event is certainly the official announcement of Tango technical roadmap for the upcoming years. Here are the foreseen features for the next major releases.

- Tango v10
  - New alarm event
  - Warning and alarm hysteresis
  - New observability service based on OpenTelemetry (for tracing and profiling)
  - Extended version information of the full software stack on which a device server relies
- Tango v11
  - New datatype: DevDict. A python like dictionary of key/value pairs.
- Tango v12
  - Multi-parameter commands breaking the current limit of one argument per command. The DevDict provided by V11 will certainly help here.
- Tango v13
  - New multi-dimensional arrays breaking the current limit of the 2D matrix.

SIG Meetings

Facing regular requests for discussions dedicated to specific topics, the Tango steering committee to launch a series of Special Interest Group (SIG) meetings. Beyond the idea of focusing on a particular subject, the aim is also to rationalize the organization of each event by adopting a common workshop format. This allows to set up a clear agenda to guide the discussions and ensure their fruitfulness. So far, four events have been organized [26].

Jupyter for Controls - SKAO Headquarter - UK - September 2022 In the Tango community, several tools relying on Jupyter [27] have been developed and evaluated as an interface to a Tango control system. For the attendees, the aim of this meeting was to share their respective experience with Jupyter and evaluate its potential usage as a client platform. *jupyTango* [28], a feature rich Tango client running in the notebook, has been presented. This tool notably allows asynchronous tasks to run in their own cell without blocking the whole interface - offering the ability to update plots and widgets in the background. It had been identified as a potential tool for systems commissioning. ASTRON [29] also demonstrated its usage of Jupyter for scripting and more.

The Tango Archiving Database (HDB++) - ASTRON - Netherlands - November 2022 The second Tango SIG Meeting [30] was organized as a hybrid (face to face and remote) meeting by ASTRON in Dwingeloo, The Netherlands. This 1.5 day event focused on practical demonstrations of attendees' recent work on HDB++ core and the related tools. This workshop also offered the opportunity to exchange about the future of Tango archiving system. The idea of

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

creating a SQLite backend for the CI tests was notably discussed. Since then, a first version of this backend has been delivered. See paper [18] at this conference for more details on the latest developments of HDB++.

The Future of GUIs (TAURUS) - ESRF - France - March 2023 In the TANGO community, TAURUS [31] is the de facto standard for those searching for a powerful python-based solution for CLIs and GUIs. Initially developed at the ALBA Synchrotron (Spain), TAURUS is also a successful open-source project widely adopted by major scientific facilities for their daily operation. After 15 years of constant development, the project has reached a key point in its history. The emergence of web-based solutions tends to challenge the existence of traditional GUIs frameworks by inviting them to redefine their role or to justify their maintenance beyond the catalog of existing applications. The historical TAURUS maintainers team is also changing and could offer new opportunities in terms of involvement in the life cycle of the project. During this event, speakers and attendees had the opportunity to share their experience with TAURUS and present their strategy and thoughts regarding the future of their graphical interfaces. The organizational and technical future of TAURUS has also been discussed. The main finding of this SIG meeting is the growing adoption of python-based and web-based solutions to the detriment of Java.

Roadmap to Tango 10 (a.k.a. IDLv6 Meeting) - ALBA - Spain - May 2023 This event was dedicated to the roadmap for the first releases of the Tango 10 series. The main aim was to identify the new features and project them on a timeline of releases. See the Community Meetings paragraph for more details about the Tango roadmap.

PACKAGING

Tango Source Distribution

The TangoSourceDistribution bundles a large number of tango related projects into one package. As with other C++ tango projects the build system was moved from automake/autotools and Visual Studio projects to CMake [17]. This also made it possible to build a Windows installer in CI (see next section) for every pull request and release.

Windows Installer

A Windows installer has been created to simplify the installation of Tango on Windows, which up until now has always been a manual process. The installer is built automatically by a GitLab CI job meaning that it is quick and easy to create a new release of the installer. The CI job downloads all of the required files for the installer from GitLab, builds the C++ applications, creates a Windows Inno Setup [32] script file, and finally builds the installer executable (`exe`) file. A user can then simply use this executable to install all of the Tango applications on a Windows machine.

Debian

The Tango debian packages have recently been upgraded to cppTango/pytango versions 9.4.2 by Freexian [33]. As the debian packages serve as basis for derived distributions like Ubuntu, the next Ubuntu release will also include cppTango/pytango version 9.4.2.

RPM

Tango RPM packages are still built using Copr [34], a build system and infrastructure provided by Fedora. CppTango 9.3 is available for both CentOS 7 and 8, but 9.4 was only built for CentOS 8. Support for CentOS 7 was dropped due to cmake being too old.

Conda

Two years ago, only a few Tango packages could be installed from conda-forge [35] and only for Linux x86_64: tango-idl, cpptango, tango-database, tango-admin, tango-test, pytango and itango. Since then, more packages, C++, Python or Java, and more platforms have been added: Linux, including x86_64, aarch64 and ppc64le architectures, Windows and macOS (both Intel and Apple silicon) as shown in Fig. 1.

Packages	Linux			macOS		Windows
tango-idl	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
cpptango	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
cpptango-dbg	linux-64	linux-aarch64	linux-ppc64le			
tango-database	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
tango-admin	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	
tango-access-control	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
tango-starter	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
tango-test	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
pytango	linux-64	linux-aarch64		osx-64	osx-arm64	win-64
jive	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
tango-astor	linux-64	linux-aarch64	linux-ppc64le	osx-64	osx-arm64	win-64
pogo	linux-64			osx-64		win-64

Figure 1: Main Tango packages on conda-forge.

On Linux, cppTango debug symbols have been moved to a separate package, keeping the main package small but allowing users to debug just by installing *cpptango-dbg*. Tango device servers for HDB++ Event Subscriber and Configuration Manager can also be installed from conda-forge (only with timescale support). On the Python side, some work was done to be able to install taurus and sardana (including Qt) on most platforms and architectures. This required re-building some of their dependencies like pythonqwt or guiqwt. Some effort was also done to provide Java applications. The initial goal was to publish individual jars to build jive and astor. This proved not possible as they depend on each other. In the end, they were published as fat JAR. Nonetheless, jtango, tango-atk, tango-atk-panel and tango-atk-tuning are also present. Astor was published as *tango-astor*, as *astor* was already taken by a Python package.

It has never been easier to install the Tango packages on all platforms.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

PyTango Wheels

PyTango wheels have been available for Windows for a while (since 9.2.0), but even for 9.3.6, they were only built for Python less than 3.8. On other platforms, installing pytango with pip wasn't advised as it had to be built from the source distribution. This required to have the proper dependencies and build tools available, and was slow. Since version 9.4, we build wheels for Linux (i686, x86_64 and aarch64), macOS (x86_64 and arm64) and Windows (win32 and win_amd64), supporting Python 3.9 to 3.11. That's 21 Python wheels for PyTango 9.4.2. This is a huge improvement but wasn't trivial. Building Windows wheels on AppVeyor currently takes more than 5 hours. Linux and macOS wheels are built on GitLab CI and the pipeline takes about 90 minutes. And that doesn't take into account the four hours to build the docker images with all the dependencies. The main issue is the aarch64 architecture which is done using emulation and explains the compilation time. This is worth it as it makes installing pytango with pip really easy on most platforms and architectures. Using pip is the now the recommended way to start with PyTango.

CONCLUSION

The Tango Controls toolkit has experienced a major boost in kernel development thanks to additional kernel developers and the continued support provided by dedicated external companies. All the libraries have seen new releases with a fixed period release cycle every 6 months being introduced. Packaging has improved considerably for all platforms (including Windows) and it has never been easier to install Tango. A new version of the network interface (IDLv6) is under development and will be released as part of Tango V10. Major refactoring has taken place to make the C++ library easier to maintain in the long term which has increased the lifetime of the current implementation. The Tango Controls community is thriving with new projects adopting Tango and new Special Interest Group (SIG) meetings on topics of common interest.

ACKNOWLEDGEMENTS

The Tango Controls community acknowledges the financial and in-kind contributions provided by the 11 partners who are members of the Tango Controls Collaboration - ALBA, DESY, ELETTRA, ESRF, INAF, MAX-IV, SKAO, SARAO, SOLARIS, SOLEIL and ELI-ERIC which has enabled much of the work above to happen.

REFERENCES

[1] J.-M. Chaize, A. Götz, W.-D. Klotz, J. Meyer, M. Perez, and E. Taurel, "TANGO - An Object Oriented Control System Based on CORBA", in *Proc. ICALEPCS'99*, Trieste, Italy, Oct. 1999, paper WA2I01, pp. 475–479.

[2] About Tango, https://www.tango-controls.org/about-us

[3] The Tango gitlab repository, https://gitlab.com/tango-controls

[4] log4cpp, https://log4cpp.sourceforge.net

[5] OpenTelemetry, https://opentelemetry.io

[6] Stack Overflow Developer Survey 2023 - most popular languages, https://survey.stackoverflow.co/2023/#most-popular-technologies-language

[7] PyTango 9.4.2 release, https://gitlab.com/tango-controls/pytango/-/releases/v9.4.2

[8] A. Götz *et al.*, "The Tango Controls Collaboration Status in 2021", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 544–549. doi:10.18429/JACoW-ICALEPCS2021-WEAR01

[9] oldest-supported-numpy package on PyPI, https://pypi.org/project/oldest-supported-numpy/

[10] NEP 29 — Recommend Python and NumPy version support as a community policy standard, https://numpy.org/neps/nep-0029-deprecation_policy.html

[11] Tango Controls meetings - Indico, https://indico.tango-controls.org

[12] Minutes from PyTango Developers' Meetings, https://gitlab.com/tango-controls/meeting-minutes/pytango

[13] JacORB, https://www.jacorb.org/

[14] JeroMQ, https://github.com/zeromq/jeromq

[15] Maven central repository, https://mvnrepository.com/artifact/org.tango-controls

[16] SLF4J, https://www.slf4j.org/

[17] CMake, https://cmake.org

[18] D. Lacoste *et al.*, "New developements on HDB++, the high-performance data archiving for Tango Controls", presented at ICALEPCS'23, Cape Town, South Africa, 2023, paper THMBCMO01, this conference.

[19] Pogo repository, https://gitlab.com/tango-controls/pogo/

[20] Xtext/Xtend home page, https://eclipse.dev/Xtext/

[21] Protected regions repository, https://github.com/danieldietrich/xtext-protected-regions

[22] Jinja, https://palletsprojects.com/p/jinja/

[23] The 36th Tango community meeting (Indico event), https://indico.tango-controls.org/event/51/

[24] The 37th Tango community meeting (Indico event), https://indico.tango-controls.org/event/57/

[25] The Taranta gitlab repository, https://gitlab.com/tango-controls/web/taranta

[26] Tango Controls Indico, https://indico.tango-controls.org

[27] Project Jupyter, https://jupyter.org

[28] The ASTRON radio telescope, https://www.astron.nl

[29] jupyTango Gitlab repository, https://gitlab.com/tango-controls/jupyTango

[30] HDB++ GitLab group, https://gitlab.com/tango-controls/hdbpp

[31] Taurus SCADA, https://taurus-scada.org/index.html

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

[32] Inno Setup, https://jrsoftware.org

[33] Freexian, https://www.freexian.com

[34] Tango Copr, https://copr.fedorainfracloud.org/coprs/g/tango-controls/tango

[35] Conda-Forge Community, "The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem", *Zenodo*, 2015. doi:10.5281/zenodo.4774216