ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

THE TANGO CONTROLS COLLABORATION STATUS IN 2025

T. Juerges, SKA Observatory, Jodrell Bank, United Kingdom
R. Bourtembourg, A. Götz, D. Lacoste, ESRF, Grenoble, France
S. Rubio-Manrique, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
B. Bertrand, A.F. Joubert, MAX IV Laboratory, Lund, Sweden
Y. Matveev, DESY, Hamburg, Germany
L. Pivetta, G. Scalamera, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
G. Kowalski, T. Madej, M. Celary, L. Zytniak, S2Innovation, Kraków, Poland
G. Abeillé, G.Pichon, P.Madela 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 the last status update in 2023, the Tango Controls collaboration has undertaken a major effort to add new features to cppTango, the core of Tango Controls, and two other official language bindings, JTango and PyTango. Significant development efforts have been dedicated to the implementation and prototyping of community-requested features. Observability is a trending topic in software development, and we have listened to our community adding OpenTelemetry support. Continuing with cppTango refactoring, we switched to C++17 and catch2 as the new testing framework to improve code quality and test coverage. PyTango has undergone a major overhaul by switching from boost-python to pybind11, which has been a welcome modernization of the code base and has allowed us to remove obsolete APIs. Special Interest Group (SIG) meetings continued to be a great success. Several have been held, among them one that addressed the request of our users for a much improved documentation. Encryption has also been a SIG topic, and a prototype for complete end-to-end encryption of all communication in Tango Controls has been developed. CI/CD has again received major updates and gained more computing power to run more tests in less time, thanks to the Gitlab runner contributions of the collaboration members. Thanks to the continuous community effort on keeping a modern and well maintained core, the future road map of Tango Controls looks promising and achievable.

INTRODUCTION

In 1998, the European Synchrotron Radiation Facility (ESRF) [1] submitted a paper on Tango, a control system framework based on the paradigm of distributed objects, to ICALEPCS 1999 [2]. Today, Tango is much more than a piece of software. First, it is a well organized, friendly, and amazingly fruitful collaboration of ten major facilities [3] and a growing community of users and developers. It is also the place of constant development, enhancement, refactoring, and innovation [4].

The success of the Tango collaboration relies on its organization, 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, finance events and technical subcontracting. The latter definitely boosts the development of Tango. It is now much more than just a framework. It is a rich and very active software ecosystem that is constantly evolving. That means Tango improves and tries to benefit from the latest technical developments while maintaining strong backward compatibility for its users.

The present paper provides the reader with the latest news from the Tango Controls collaboration and its ecosystem. We aim to present a broad overview of both the organizational and technical activity around Tango.

CPPTANGO

cppTango is the C++ implementation of the Tango kernel and is considered the reference implementation of Tango. Due to the performance and efficiency of C++, cppTango is often used for device servers (DS) that interface with hardware that require high speed communication or have to handle high volume data throughput. Examples are detectors or motors in synchrotron beamlines.

The latest major release (10.0.0) added observability features by integrating with the OpenTelemetry [5] framework for unified monitoring across many device servers. The goal is to make it easier to trace and diagnose problems in complex installations by correlating logs and metrics from multiple sources. Another major feature that has been implemented is the alarm event, which provides an automatic signaling by the device server when the device state changes to or from Alarm. New is also that device server developers have the possibility to add extended version information to device servers. Examples could be the source code version, versions of used libraries or firmware revisions of connected hardware. The latest cppTango patch release (10.1.0) has added performance monitoring of the ZeroMQ event subsystem, new fully asynchronous event subscriptions, and has fixes for all known memory leaks. Despite all the bug fixes, improvements, changes of internal and external APIs and ABIs, cppTango has maintained backward compatibility with existing Tango device servers which were developed

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

many years ago for Tango 8 or 9. They can still be compiled with the latest cppTango release with minimal or no changes, due to careful adherence to API stability.

PYTANGO

Within the last decade, Python has become one of the most popular programming languages in the world [6], not least in scientific computing. With the PyTango project [7], the Tango collaboration provides a Python binding to the Tango C++ kernel, allowing extremely easy development of servers and clients with a high-level Python-friendly API.

Releases

The PyTango library started following the release cycle of the C++ kernel, ensuring that Python developers can almost immediately use all new features of the C++ kernel. Currently, PyTango is released for Python >= 3.9 for Linux/Windows/MacOS, (x86 and 64-bit) and distributed as conda-forge [8] packages and binary wheels on PyPI.

In the last two years there were two major releases of PyTango: 9.5.x [9] and 10.0.x [10]. Note that PyTango does not use semantic versioning [11]. Among numerous bug fixes, the PyTango team has added several new features.

Version 9.5 added the option of declaring attribute/command/property data types with Python type hints. This allows simpler, more readable device code and allows doing static typing checks with tools like mypy [12].

In version 10.0 the core interface definition language (IDL) specification [13] was updated from v5 to v6. This provides clients with better device information, a new category of events: alarm events, and W3C Trace Context [14] propagation needed for distributed tracing. PyTango adopted all of these changes. The most powerful feature is built-in support for OpenTelemetry, significantly improving distributed system observability with zero code changes from users. PyTango updated the asyncio server implementation, moving away from the deprecated coroutine decorator solution. It also means that on-the-fly conversion of synchronous functions to coroutines in PyTango servers is now deprecated and will be removed in an upcoming release.

Code maintenance

However, the main efforts within the last 1.5 years have been put in improving the code base. The test coverage was extended to almost 80 percent, and continuous integration reports the coverage for both C++ and Python code. The developer team hopes that this will help to maintain code quality, and compatibility of the new releases with existing user code.

Also, this year the long running port [15] of the C++ extension code from Boost [16] to pybind11 [17] was finally accomplished. By doing this, the binding code was revised and cleaned. But the most important is that it opened the opportunity to improve both PyTango and the cppTango code. As an example, after the recent move from raw to smart pointers for attribute values in the cppTango code, the old PyTango Boost code does not compile anymore.

Asynchronous Tango device servers

The asynchronous programming paradigm is gaining ground in the Python community as it allows writing more responsive, scalable code thanks to its better concurrency model for I/O-bound tasks [18]. The green modes of PyTango [19] in their different flavors (gevent, concurrent.threads, asyncio) have been used successfully by Tango clients along with asynchronous web frameworks [20].

New PyTango high-level device servers are also using PyTango asyncio green mode to interface hardware using asyncio-based communication libraries, either at TCP, serial or lower-level communication layers (DMA); effectively forwarding hardware-triggered events to Tango clients with no need of internal polling. This fully asynchronous architecture allowed to implement responsive device servers with hundreds of attributes and response times in the range of milliseconds [21].

JTANGO

JTango is the Java language implementation of Tango Controls. It enables the development of Tango device servers and clients in Java, integrating Tango into Java-based projects. While not as universally used for device servers as C++ or Python, JTango plays an indispensable role: many of Tango's client tools and graphical user interfaces are built on Java and rely on the JTango client libraries. For instance, applications like Jive (the Tango device browser and configurator), ATKPanel (a generic Tango device control panel), and Astor (a tool for managing and starting/stopping device servers) are all Java programs that use JTango under the hood. Thus, even if a site doesn't develop custom Java device servers, they are likely using JTango via these tools for daily operations. JTango's core library implements the Tango API in Java and uses analogous technologies to Tango's C++ core: it relies on JacORB (an open-source CORBA implementation for Java) to handle network requests, and JeroMQ (a pure Java ZeroMQ library) for event transport. This parallels the cppTango approach (omniORB and ZeroMQ in C++), ensuring compatibility at the protocol level. The JTango project has made its artifacts readily available to Java developers by publishing them on Maven Central – the standard repository for Java libraries. This means that integrating Tango into a Java application is as simple as adding a dependency. This significantly lowers the barrier of entry for Java projects that wish to use Tango.

JTango's latest release implements the Tango IDL v6 specification mentioned above (OpenTelemetry, Alarm events, enhanced device information for clients). The development will continue to follow all specifications of the community. In parallel, CI/CD pipelines were updated (Jacorb IDL compilation project and JTango project) to comply with the latest changes imposed by Maven Central. In addition the code

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

quality tool link configuration (SonarCloud) has also been fixed.

TANGO DATABASE

The Tango Database is implemented as a Tango device server, therefore it is often called Databaseds in configurations, that typically runs on a designated host in the control system. It uses a MySQL or MariaDB relational database backend to store information about devices, servers, Tango properties, IP addresses and ports of device servers, and more. This allows any client or server in the Tango Controls system to query the Tango database or receive connection details for a Tango device.

From a user perspective, the Tango database server greatly simplifies configuration management: adding a new device or device server is as easy as updating the database (usually done with tools like Jive or tango-admin). In addition to storing property values, attribute values can be persisted. When a device server starts up it automatically downloads initial values for attributes from the Tango Database, so that calibration factors, saved states or anything that requires persistence is restored.

For development exists a technology preview of a Python-only implementation using SQLite [22] called pytango-db [23].

WEB INTERFACES AND APIS

As control system infrastructure becomes more distributed, web technologies have become important because they allow one to run frontends anywhere. Tango has embraced this by developing web interfaces and HTTP/JSON APIs so that Tango can be accessed not only by desktop client applications, but also via web browsers and standard web protocols. The Taranta framework has been created for custom application development and several stand-alone web applications have been developed to access specific Tango services (like HDB++ archiving [24, 25] and Panic alarm system [26, 27]).

Taranta

Taranta is Tango's web application. It allows users to create graphical interfaces for Tango without knowledge of web development. Key points:

- Users can design a dashboard with widgets (charts, gauges, buttons) and connect them to Tango device attributes or commands.
- Taranta focuses on letting users build interfaces tailored to their needs without writing code. They can be as simple as a summary screen displaying a pressure or temperature or more complex with a control panel for an entire subsystem.
- It supports various diagram types and control elements, and these can interact with Tango in real-time.
- Taranta is part of the push to make Tango more accessible via standard web technologies. With it, one could run an experiment from a tablet or any browser, as opposed to needing a desktop application.

Tango REST API

Before the advent of Taranta, Tango defined a RESTful API. It exposes Tango devices and their objects (attributes, commands, pipes, etc.) through a standard HTTP interface. One could `GET` a URL to read an attribute or `POST` to execute a command. This API is useful for integrating web apps or simple scripts in any language without using the full Tango client libraries.

The Tango REST API is currently in "life support mode", which means that it is not actively developed or maintained by the Tango developers. This indicates that while it works, the focus has shifted to more modern approaches such as GraphQL or direct web frameworks.

TangoGraphQL and TangoGQL

GraphQL is a modern query language for APIs, and in Tango exist two projects that implement a GraphQL interface for Tango: TangoGraphQL [28] (implemented at ESRF in C++) and TangoGQL [29] (implemented at MAX IV in Python). Both are services that allow clients to query Tango devices using GraphQL queries over HTTP/WebSocket, but use different GraphQL schemas. GraphQL has several advantages:

- Allows clients to request exactly the data they need (e.g. "give me the values of attributes X, Y, Z of device D, and also the state of device E, in one request"). Compared to REST, this can reduce the number of round-trips.
- Naturally supports a subscription model (GraphQL subscriptions over WebSocket), which maps well to Tango's event subscription.
- Provides a schema that documents the API (GraphQL has introspection), which is excellent for discoverability. Tools like GraphiQL can be used to explore the Tango device schema and experiment with queries, which is more user-friendly than raw REST.
- One can build very rich web applications. For example, a React web app could use a GraphQL query to fetch many attributes in one go, and subscribe to updates, all through a single unified endpoint.

TangoGQL was initially developed based on the Graphene [30] Python library. Due to dependencies issues and pitfalls in the implementation, where some queries could cause huge amounts of traffic against the control system and take a long time, a rewrite based on the Ariadne [31] library was started in 2023 and became the official implementation used by Taranta end of 2024.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

HDB++

In experiments and industrial controls it is often necessary to log or archive the evolution of certain device parameters over time – for example, to monitor trends, debug issues, or store engineering or experimental data. Tango's solution for this is the HDB++ (Historical Database Plus Plus [32, 33]) archiving system. HDB++ is a Tango event-driven archiving service that records Tango attribute values with their timestamps in a database for later retrieval and analysis. It is designed to handle high volumes of data with minimal impact on the front-end device servers.

HDB++ is not a single program but a set of components: it defines a schema and libraries (LibHDB++) and comes with example archiver device servers that subscribe to Tango events from other devices and commit those values to storage. One of HDB++'s strengths is its modular backend support – it can work with different database systems depending on the site's preference and performance needs. Current supported backends are MySQL/MariaDB and PostgreSQL, the latter often combined with the TimescaleDB [34] extension for time-series optimization. Recently, SQLite was added as a new backend option as a lightweight file-based database; this addition is useful for small installations or testing scenarios where running a full database server is not desired. Thanks to an abstraction layer, the archiving service can write to any of these supported backends without changing the higher-level logic.

Using the same backend-agnostic approach, pyhdbpp [35] provides a unified data extraction API for Python clients. It provides transparent access to HDB++ data independently of the backend and allows to join data from multiple sources. Pyhdbpp has been used to integrate HDB++ data sources in Grafana.

POGO

Developing a new device server from scratch involves writing a lot of boilerplate code, especially for the lower-level languages such as C++. To streamline this process, the Tango community provides Pogo, a code generator for Tango device servers. Pogo [36] is a tool that allows you to graphically design your device class (define its attributes, commands, properties, pipes, and state machine) and then generate skeleton source code in C++, Python, or Java, which can be filled in with the device-specific logic. This significantly speeds up development and ensures consistency in how device servers are structured.

Pogo has long been a central piece of the Tango toolkit. A developer using Pogo can avoid repetitive coding; for example, if you need a new Motor device with certain attributes, you can input those in Pogo and it will produce a ready-to-compile C++ class with all the necessary Tango glue code. Pogo was traditionally a Java application (Eclipse-based), leveraging frameworks like Xtext to model the device classes and generate code. It has a GUI where developers enter the details of devices and then generate code with a button click. The generated code includes all required classes, project files, and even basic implementations that can be directly compiled or executed after minor editing.

CI/CD

Continuous integration (CI) and delivery (CD) is now fully embraced across the tango ecosystem. For every merge request for cppTango, PyTango, TangoSourceDistribution, TangoDatabase, tango-doc and tango_admin, tests are run in order to catch issues before a merge request is merged. A **matrix of CI runners** has been added: Linux (various distros), Windows, and macOS. Cross-platform build and test issues can now be identified before merging code. Continuous delivery is also embraced. One example is that every successful merge into the main branch can trigger an automatic package build and upload (for nightly or RC builds). In particular did the PyTango team leverage GitLab CI to produce nightly wheels and test them against the latest cppTango development version.

In summary, the development workflow for Tango has been transformed: Where previously a single developer has had to manually cut a release and upload binaries, which resulted in a significant delay of the releases, now **CI pipelines handle most of the heavy lifting**: From building installers to running thousands of tests and packaging for multiple distribution channels, the pipelines do the work. This results in more reliable releases and faster turnaround times. When a critical bug is fixed, a patch release can be packaged and delivered via Gitlab releases/Conda/PyPI within a day.

TANGOBOX

TangoBox [37] is a pre-packaged Tango Controls environment and distributed as a virtual appliance. Its purpose is to provide a reliable, out-of-the-box experience of a working Tango Controls system, with a full suite of Tango tools pre-installed and correctly configured. The TangoBox virtual machine (VM) comes with a desktop full of shortcuts to all essential Tango applications and includes tutorial resources. This allows newcomers and interested parties to immediately **experience a fully configured control system with the latest tools**. In practice, TangoBox has become an **easy-entry sandbox** for Tango Controls: It lowers the barrier for trying out and testing the complete Tango Controls ecosystem. The VM images can be run on a local VirtualBox [38] or VMware [39] hypervisor, or they can even be deployed to cloud providers. This flexibility gives developers and institutions access to a ready-made Tango environment on a laptop or in the cloud without any complex setup process.

TANGO CONTROLS COLLABORATION

Web Presence and Community Engagement Tools

The Tango website [41] has been migrated to wagtail CMS. The forum [42] has also been migrated and is now running on Discourse. These migrations were approved and funded by the Steering Committee to improve long-term access to community knowledge. The discussion platform has been

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

migrated from Slack to an open source Mattermost hosted by Helmholtz Zentrum Dresden-Rossendorf (HZDR [43]), to allow greater participation without the restrictions of a free Slack instance.

Community Meetings

The annual Tango Community Meeting remains a highlight for users and developers alike. Since 2021, these meetings have been held online or hybrid, attracting more than 100 participants and numerous talks. Community meetings provide a platform for member institutes to report on their deployments and for new programs (like the SKA Observatory joining Tango) to share their experience.

Since ICALEPCS'23 [44], the 38th [45] and 39th [46] editions of the Tango Community meeting have been hosted by SOLEIL (Saint-Aubin, France) and INAF (Giulianova, Italy).

Special Interest Group Meetings

The Tango community launched Special Interest Group meetings (SIGs) to tackle specific topics that require more dedicated discussions than the biweekly kernel calls allow. The SIG format – one-off events, with a narrow focus on a topics – proved to be very successful for the community and the development of Tango and will continue in the future.

As an example the SIG meeting *"Roadmap to Tango 10 (IDL v6)"*, hosted by ALBA [47] in May 2023, has proven to be the most influential so far. At this workshop, core developers charted out the features and timeline for Tango's next major versions (v10 through v13), producing the official **Tango Technical Roadmap** announced at the 37th Tango Community Meeting [48], at ICALEPCS'23 and a refined version at the 39th Tango Community Meeting.

Cyber security, May 2024 Large control systems are not immune to cyber security threats. The Tango Control system framework was created at a time when networks could be assumed protected by perimeter security, like firewalls. Times have changed and currently the best practice is a zero-trust architecture [49]. In other words, one assumes that the network has already been compromised. A SIG-meeting was held in May 2024 to work on improving the security model offered by Tango Controls.

At the meeting a cppTango prototype with full encryption of the ZeroMQ and CORBA traffic was implemented. A draft request-for-comment document [50] with the formal specification of encryption in Tango Controls was created. Other sessions at that SIG meeting investigated secrets management, software supply chain, and user access control.

Documentation, October 2024 The Tango documentation [51] has recently undergone a major overhaul following feedback from the community that it needed to be updated. An initial review of the current material was carried out and identified areas that were in need of updating or were no longer applicable to the most recent versions of Tango. The decision was also taken to restructure the documentation following the Diataxis [52] approach to documentation, which breaks the document down into four key sections: tutorials, how-tos, explanations, and references. The aim of this restructure was to provide a clearer entry point to new users/developers while still providing more detailed material for those with experience with Tango, a paradigm that can be achieved using the Diataxis approach.

A SIG-meeting was held in October 2024 to initiate the restructuring process and stimulate effort into updating the documentation content. This was followed by a further six months of effort from a dedicated team who had regular remote meetings to continue to make these updates and also worked on improving the user experience when navigating the documentation.

Some new features of the documentation include the use of sub-projects, meaning that documentation from other projects directly related to the core Tango projects (e.g. PyTango, Jive, AtK, etc.) can be accessed under the same 'tango-controls' domain name. As such, searches from the main documentation have been configured to include sub-project documentation, providing a more complete and easier searching facility. A lot of effort was also put into developing helpful and robust how-to examples to address some of the common tasks users might want to achieve using Tango. These have been made available in all three of the core Tango languages: C++, Python, and Java. Beyond these changes, there have been updates to improve the general usability of the documentation, including an update to the theme, improvements to cross-referencing links, and an enhanced use of the glossary for explaining fundamental Tango terms.

The new and improved documentation went live in May 2025 and has been well received by the community. An effort will now be made to maintain the documentation so that it stays up-to-date and continues to be a useful source of information for Tango users worldwide.

PACKAGING

Platform packaging of Tango has improved dramatically over the past years. Not only are now Linux distribution packages available, also a Windows installer and Python wheels are supplied by the community. Conda packaging has seen a boost with the availability of almost all Tango tools, libraries and programs across all major platforms.

Debian and RPM

Tango's Debian/Ubuntu packages were updated to the latest Tango releases, e.g. cppTango 10.0.2 in Debian "trixie". Maintenance of these packages was outsourced to Freexian [53]. On the RPM side, Tango continues to be available via Fedora COPR [54] repositories for RedHat 8 and 9, while the support for CentOS has been discontinued.

PyTango wheels

PyTango publishes Python wheels on PyPI for all major platforms such as Linux amd64, macOS amd64 and arm64 and Windows amd64. This means that PyTango is just a

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

quick `pip install --require-virtualenv pytango` away.

Conda

The community keeps maintaining and adding packages related to the Tango ecosystem, on conda-forge covering C++, Python, and Java for multiple platforms (Windows, Linux, macOS on Intel and ARM). One can not only install `pytango` and `cppTango` via Conda, but also Tango's Database server, Starter, and even GUI applications like Jive and ATK. These Java applications were packaged as "Fat JARs" (i.e. they include all dependencies) to simplify their deployment.

This multi-platform, one-command installation capability is a huge convenience for users. A scientist can

```
conda install -c conda-forge \
    pytango tango-database jive
```

on Windows/Linux/Mac and get a working Tango environment in minutes. Conda packages support a wide range of use cases, including production deployment, local development with complex dependencies such as compilers, and continuous integration (CI) testing.

New tools such as micromamba [55] and pixi [56] significantly accelerate and simplify the creation and management of conda environments. The PyTango repository has adopted pixi, which enhances the developer experience by providing predefined tasks to facilitate compilation and testing across different Python and even cpptango versions.

Tango OCI images

SKAO [57] maintains and host a collection of OCI [58] images containing a wide collection of Tango tools, including Jive, Pogo and HDB++. They also provide images and instructions tailored for building Tango OCI images, so that you can containerize your own Tango device servers. The OCI image documentation [59] provides a collection of example docker compose files, making a complete Tango control system deployable with a simple `docker compose up`.

Tango Source Distribution

This project bundles the core C++ and Java package sources and serves as the basis for the Windows Installer and the Debian packages. Most of the work was in the area of streamlining the workflows so that rc releases are available simultaneously with the cppTango rc releases. This included making it the responsibility of each package maintainer to make requests for package updates. An up-to-date `RELEASE_NOTES.md` file in the repository now also ensures that users can quickly see what changed. The only new package is `UniversalTest` which is a specialized test device server that uses dynamic attributes.

Windows Installer

The official Windows installer for Tango bundles all the necessary Tango components. This installer is rebuilt automatically for every TangoSourceDistribution release as part of the release pipeline. We switched Visual Studio version to 2022 to standardize across projects and be able to use C++17 as required by cppTango.

SUCCESS STORY

The most powerful new feature in Tango 10.0 is the addition of OpenTelemetry support. This allows much improved observability of our distributed controls systems.

The MAX IV synchrotron in Lund, Sweden started using this feature in production on a beamline in early 2025 [60]. The performance impact is negligible from the beamline users' perspective, and experiments proceeded as normal.

The telemetry traces and logs could easily be viewed from a web page, and this helped diagnose some issues, as well as giving us a better understanding of the complexity of the system.

More work is needed to set up a large enough backend to accept traffic from all beamlines and the accelerator, as well as creating dashboards and discovering the best ways to analyze the data.

CONCLUSION

Between 2023 and 2025 Tango Controls has significantly updated its technology stack:

- Tango 10.0 brings internal improvements (tracing, events, refactored code, new IDL for future improvements) while keeping the user APIs mostly stable.
- Observability via OpenTelemetry integration addresses a crucial need for debugging and performance tuning in distributed setups.
- Web technologies (Taranta, REST/GraphQL) make Tango more accessible and integrable, enabling modern UIs and API-driven usage of Tango devices.
- Containerization and CI/CD have modernized deployment and development, ensuring that Tango can be delivered consistently and scaled with orchestration tools.

These changes are largely driven by the collaboration of the community - for example, SKAO requirements pushing container and GraphQL development, core developers addressing technical debt and telemetry, and the general recognition that Tango must interface well with contemporary software tools. The result is that Tango in 2025 is not only a proven control system for hardware but also a modern software platform aligned with current best practices.

ACKNOWLEDGEMENTS

The Tango Controls community acknowledges the financial and in-kind contributions provided by the 10 partners who are members of the Tango Controls Collaboration - ALBA, DESY [61], ELETTRA [62], ESRF, INAF [64], MAX-IV [63], SKAO, SARAO [65], SOLARIS [66] and SOLEIL [67] which has enabled much of the work above

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

to happen. We also would like to thank everybody who has contributed to Tango Controls by starting discussions, asking questions, opening issues, or submitting merge requests. A special thanks to our subcontractors who help us greatly to take Tango Controls a step forward with every line of code they provide (or remove ;-)).

REFERENCES

[1] European Synchrotron Radiation Facility (ESRF), https://www.esrf.fr/

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

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

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

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

[6] Stack Overflow Developer Survey 2023 - most popular languages, https://survey.stackoverflow.co/2025/technology/#1-programming-scripting-and-markup-languages

[7] PyTango documentation, https://tango-controls.readthedocs.io/projects/pytango

[8] 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

[9] PyTango 9.5.0 release, https://gitlab.com/tango-controls/pytango/-/releases/v9.5.0

[10] PyTango 10.0.0 release, https://gitlab.com/tango-controls/pytango/-/releases/v10.0.0

[11] Semantic Versioning, https://semver.org

[12] mypy - Optional Static Typing for Python, https://www.mypy-lang.org

[13] Tango CORBA Interface Definition Language specification, https://gitlab.com/tango-controls/tango-idl

[14] W3C Trace Context recommendation, https://www.w3.org/TR/trace-context/

[15] A. Götz *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

[16] Boost.Python documentation, https://www.boost.org/doc/libs/1_89_0/libs/python/doc/html/index.html

[17] pybind11, https://github.com/pybind/pybind11

[18] PEP 492 – Coroutines with async and await syntax, https://peps.python.org/pep-0492/

[19] PyTango Green Modes, https://tango-controls.readthedocs.io/projects/pytango/en/latest/tutorial/green_modes.html

[20] M. Eguiraun *et al.*, "Taranta, the No-Code Web Dashboard in Production", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 1017–1022. doi:10.18429/JACoW-ICALEPCS2021-FRAR01

[21] E. Morales, *et al.*, "Towards asynchronous control systems, an asyncio implementation of OPC UA using TANGO green modes", presented at *ICALEPCS'25*, Chicago, IL, USA, Sep. 2025, paper THBR006, this conference.

[22] SQLite Home Page, https://www.sqlite.org

[23] PyTango Database on PyPI, https://pypi.org/project/pytango-db/

[24] HDB++ Grafana Python plugin, https://gitlab.com/tango-controls/hdbpp/libhdbpp-grafana-connector

[25] ArchViewer: A simple web interface to the HDB++ archiving system, https://gitlab.com/tango-controls/hdbpp/archviewer

[26] Panic Alarm System for Tango, https://gitlab.com/tango-controls/panic

[27] L. Zytniak *et al.*, "IC@MS – web-based alarm management software", presented at ICALEPCS'25, Chicago, IL, USA, Sep. 2025, paper THPD058, this conference.

[28] TangoGraphQL, https://gitlab.com/tango-controls/TangoGraphQL

[29] Tangogql Ariadne, https://gitlab.com/tango-controls/web/tangogql-ariadne

[30] graphene-python, https://graphene-python.org

[31] Ariadne, https://ariadnegraphql.org

[32] L. Pivetta *et al.*, "HDB++: A New Archiving System for TANGO", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 652–655. doi:10.18429/JACoW-ICALEPCS2015-WED3O04

[33] D. Lacoste *et al.*, "HDB++, a retrospective on 5+ years using Timescale", presented at ICALEPCS'25, Chicago, IL, USA, Sep. 2025, paper TUMR004, this conference.

[34] Timescale, https://www.timescale.com

[35] pyhdbpp Python API for HDB++, https://gitlab.com/tango-controls/hdbpp/libhdbpp-python

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

[37] TangoBox 10, doi:10.5281/zenodo.16964610

[38] VirtualBox, https://www.virtualbox.org

[39] VmWare, https://www.vmware.com/

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

[41] Tango Controls website, https://www.tango-controls.org

[42] Tango Controls forum, https://forum.tango-controls.org

[43] Tango Controls Mattermost, https://mattermost.hzdr.de/tango-controls

[44] T. Juerges *et al.*, "The Tango Controls Collaboration Status in 2023", in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023., paper TH1BCO03, pp. 1100-1107. doi:10.18429/JACoW-ICALEPCS2023-TH1BCO03

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEAG001

[45] The 38th Tango community meeting, https://indico.tango-controls.org/event/261/

[46] The 39th Tango community meeting, https://indico.tango-controls.org/event/422/

[47] ALBA Synchrotron, https://www.cells.es/en/

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

[49] NIST: Zero Trust Architecture, https://www.nist.gov/publications/zero-trust-architecture

[50] Tango Controls RFC 17 - Encryption - Draft, https://gitlab.com/tango-controls/rfc/-/merge_requests/172

[51] Tango Controls Documentation, https://tango-controls.readthedocs.io

[52] Diataxis, https://diataxis.fr

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

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

[55] micromamba, https://mamba.readthedocs.io/en/latest/user_guide/micromamba.html

[56] pixi, https://pixi.sh

[57] The Square Kilometre Array Observatory (SKAO), https://skao.int

[58] Open Container Initiative, https://opencontainers.org

[59] SKA Tango OCI Images, https://developer.skao.int/projects/ska-tango-images/en/stable

[60] A. F. Joubert, L. Zhu, and B. Bertrand, "Better software observability using Tango Controls with OpenTelemetry - experience at MAX IV", presented at ICALEPCS'25, Chicago, IL, USA, Sep. 2025, paper WEAG006, this conference.

[61] Deutsches Elektronen-Synchrotron DESY, https://www.desy.de/

[62] Elettra Sincrotrone Trieste, https://www.elettra.eu

[63] MAX IV Laboratory, https://www.maxiv.lu.se

[64] Istituto Nazionale di Astrofisica (INAF), http://www.inaf.it/

[65] South African Radio Astronomy Observatory (NRF/SARAO), https://www.sarao.ac.za/

[66] Narodowe Centrum Promieniowania Synchrotronowego SOLARIS, https://synchrotron.uj.edu.pl/

[67] Source Optimisée de Lumière d'Énergie Intermédiaire du LURE (SOLEIL), https://www.synchrotron-soleil.fr/