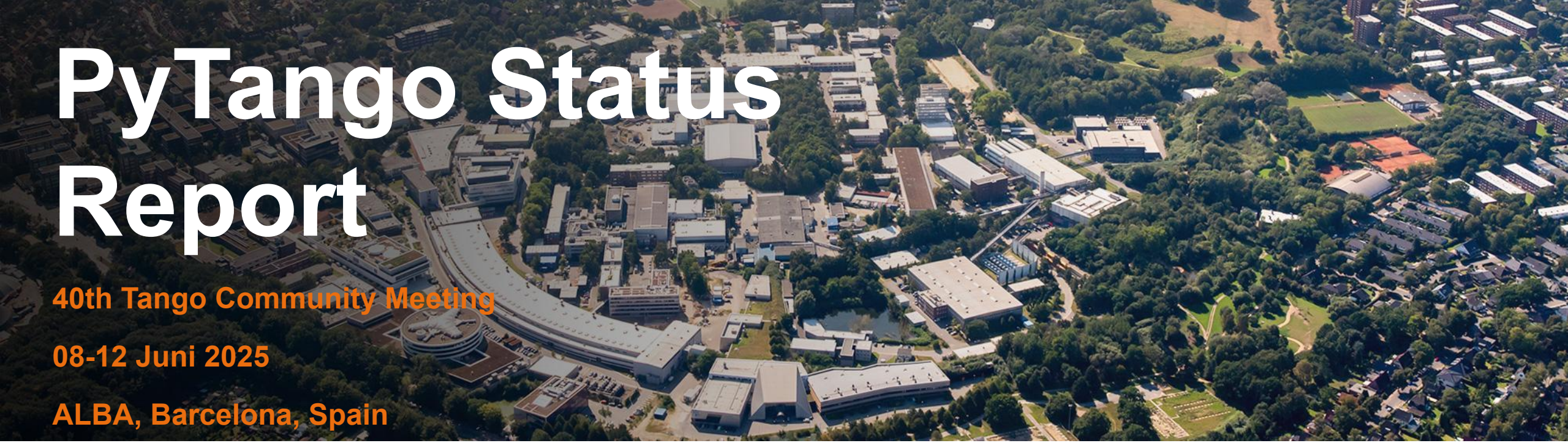
PyTango Status Report



40th Tango Community Meeting

08-12 Juni 2025

ALBA, Barcelona, Spain

Yury Matveev
Deutsches Elektronen-Synchrotron DESY

Anton Joubert
MAX IV Laboratory

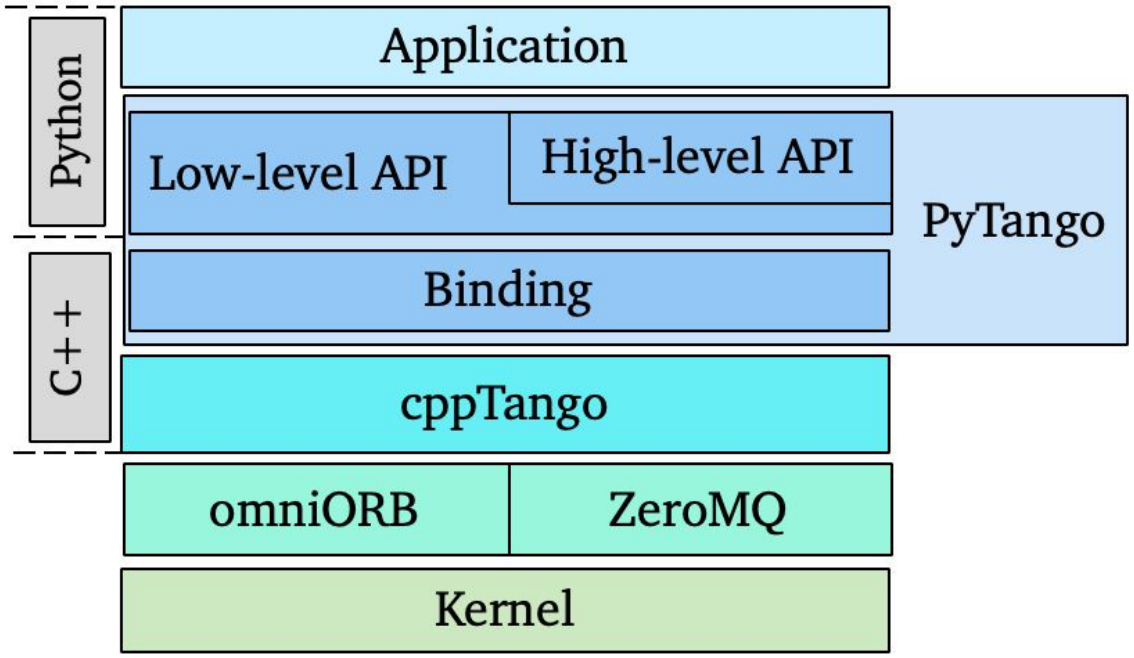
HELMHOLTZ

TANGO

DESY.

PyTango? Quick reminder

- ✔ Python library
- ✔ Binding over the C++ Tango library
- ✔ Multi OS: Linux, Windows and macOS
- ✔ Python 3.10 to 3.14

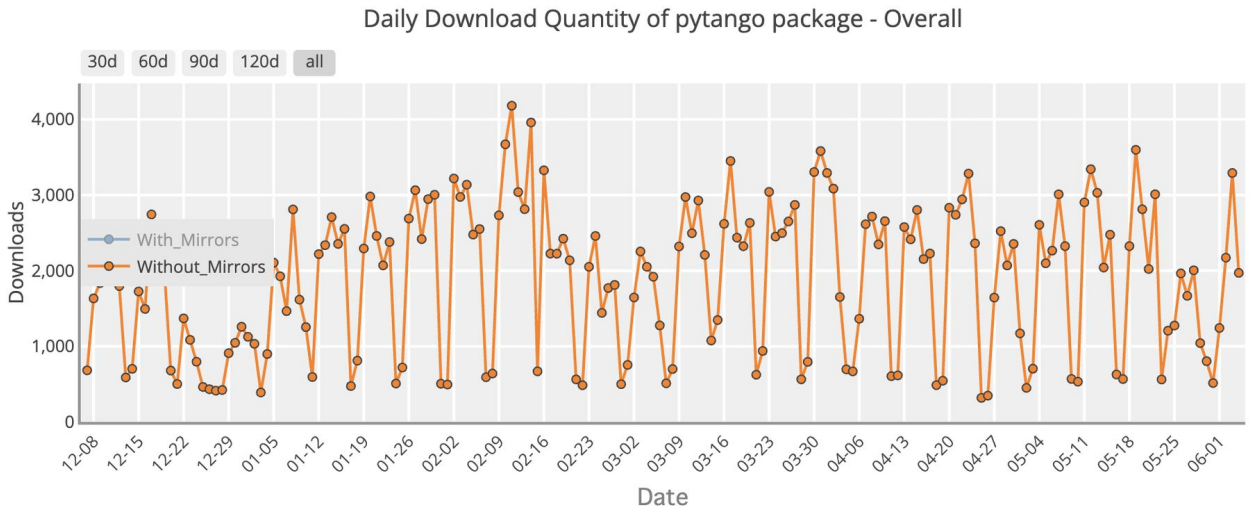


PyTango usage: some statistics

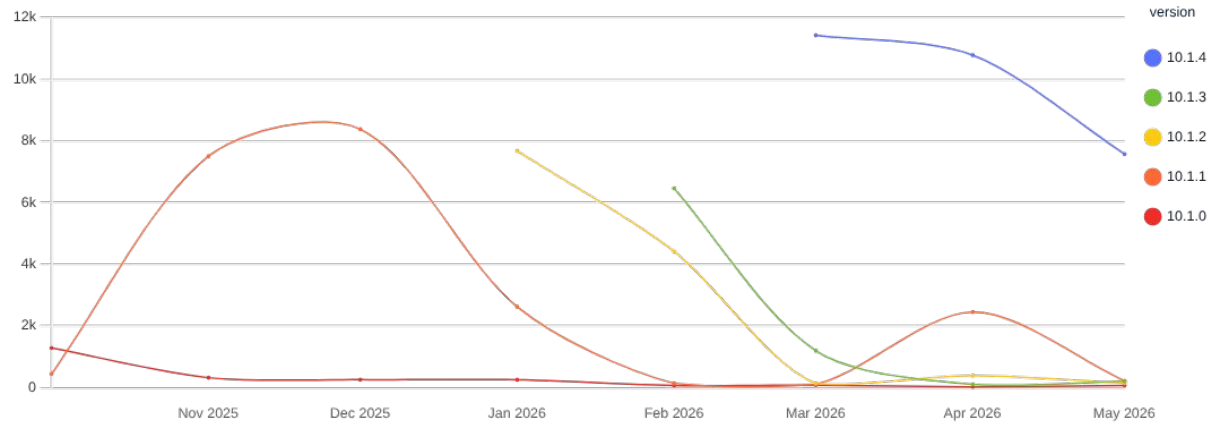
- ✔ Almost 3K downloads per day (most probably CI jobs)
- ✔ 99,9% Linux (90% Debian/Ubuntu)
- ✔ new releases instantly adopted (again, seems CI jobs)

```
=== Download statistics for 'pytango' ===
| Operating system | PyPI (~180d) | conda-forge |
| ---------------- | ------------ | ----------- |
| Linux            |      304,641 |     170,830 |
| macOS            |        1,557 |       8,956 |
| Windows          |        5,983 |       6,917 |
| noarch / other   |       24,039 |           0 |
| TOTAL            |      336,220 |     186,703 |
```

Downloads statistics: PyPi



Downloads statistics: Conda



PyTango Team

Regular attendees of our developers' meetings, which started in September 2022:

- Anton Joubert (MAX IV)
- Benjamin Bertrand (MAX IV)
- Yury Matveev (DESY)
- Jose Antonio Ramos Andrades (ALBA)
- Jairo Moldes Fuentes (ALBA)
- Thomas Ives (Observatory Sciences, SKAO)
- Thomas Juerges (SKAO)
- Thomas Braun (Byte Physics)
- Corne Lukken (ASTRON)

We meet twice a month - 1st and 3rd Thursdays. 15:00 to 16:00 CET/CEST.

The #pytango channel on Tango Controls Mattermost (do not join anymore; wait for sussessior) ☺️

Meeting minutes: https://gitlab.com/tango-controls/meeting-minutes/pytango

Current major release - 10.3.0

June 2026

- Device telemetry can now be reconfigured at runtime (presentation by Anton Joubert)
- Telemetry topics can be used (presentation by Anton Joubert)
- DeviceProxy now caches attribute configuration by default to speed up writes (one IO instead of two)
- It possible to restart a Tango server within the same process -> tests don't needed *--forked* anymore (however, only if you test the same green_mode !)
- Fixed silent coercion of typos (e.g. `'"ture"`, `'"yse"` or ['true, true']) to `False` when parsing string device property values as booleans. String-to-boolean conversion now strictly accepts only `*y/yes/t/true/on/1*` and `*n/no/f/false/off/0*`
- Equality operators (`==` / `!=`) are now exposed for *CommandInfo, AttributeInfo, AttributeInfoEx, AttributeEventInfo*, and *AttributeAlarmInfo*
- New *PyTangoThread* class wraps threading.Thread inside an *EnsureOmniThread* context - easier to use

Packages for 10.3.0

- Python 3.10 to 3.14
- Source on PyPI
- Binary wheels on PyPI: bundled with cppTango 10.3.2; omniorb, zmq, etc.
    - Windows: 32-bit, 64-bit
    - Linux: x86_64, i686, aarch64
    - macOS: x86_64, arm64
- Conda binary (pytango on conda-forge channel):
    - Linux: x86_64, aarch64
    - Windows: 64-bit
    - macOS: x86_64, arm64

Previous major release - 10.1.0 (yanked, use 10.1.4)

October 2025

- moving the C++ Python bindings from Boost.Python to Pybind11. **There are a few breaking changes.**
- New asynchronous mode for subscribing to events
- References to event subscription callback methods are immediately cleared on unsubscription (Sardana can remove its hack-patch of our hack-patch)
- Clients and servers now release the GIL in most cases when waiting on cppTango
- Better types hints for the extension code
- When defining attributes, you can indicate that they push events in place; no need for extra calls to set_<….>_event
- The client identity is now available in devices. If OpenTelemetry is enabled, the client details will be included in spans
- Added fast overload for attribute write methods with which the additional IO to server can be avoided

Pybind11: API changes

1. **Pipes** and all related method **were removed**
2. **Enums:** *__repr__* : if you did *repr(DevState.ON)* with boost you got "*tango._tango.DevState.ON"*, now *"<DevState.ON: 0>“*
3. **dim_x and dim_y** kwargs for *Attribute.set_value*, *Attribute.set_value_date_quality*, *Device.push_<>_event* are not supported
4. all **docstrings** for classes, methods and enums in pybind11 **aren't mutable**
5. Vectors *StdStringVector, StdLongVector, StdDoubleVector* are now implicitly convertible to Python lists, no need to convert
6. *StdGroupAttrReplyVector, StdGroupCmdReplyVector, StdGroupReplyVector* aren't exported any more. Instead, user receives *list[GroupAttrReply], list[GroupCmdReply], list[GroupReply]*, respectively
7. Attribute configuration (*AttributeConfig, AttributeAlarm*, etc) structs interface frozen

Summary with last year:

Release	MRs	Commits	Files touched	Lines edited (max(removed, added))
10.0.3	11	21	38	593
10.0.4	2	11	10	65
10.1.0	109	537	292	27643
10.1.1	5	11	14	142
10.1.2	3	4	10	204
10.1.3	1	4	10	103
10.1.4	7	8	31	226
10.3.0	42	170	209	26244
Total	**182**	**766**	**614**	**55220 (whole PyTango itself ~88000)**

Contributors: Anton Joubert, Benjamin Bertrand, Thomas Ives, Yury Matveyev, Tom Moynihan, Johan Forsberg,

Jairo Moldes, Szymon Kupiecki, Marcelo Alcocer, Antonio Bartalesi, Mark Ashdown

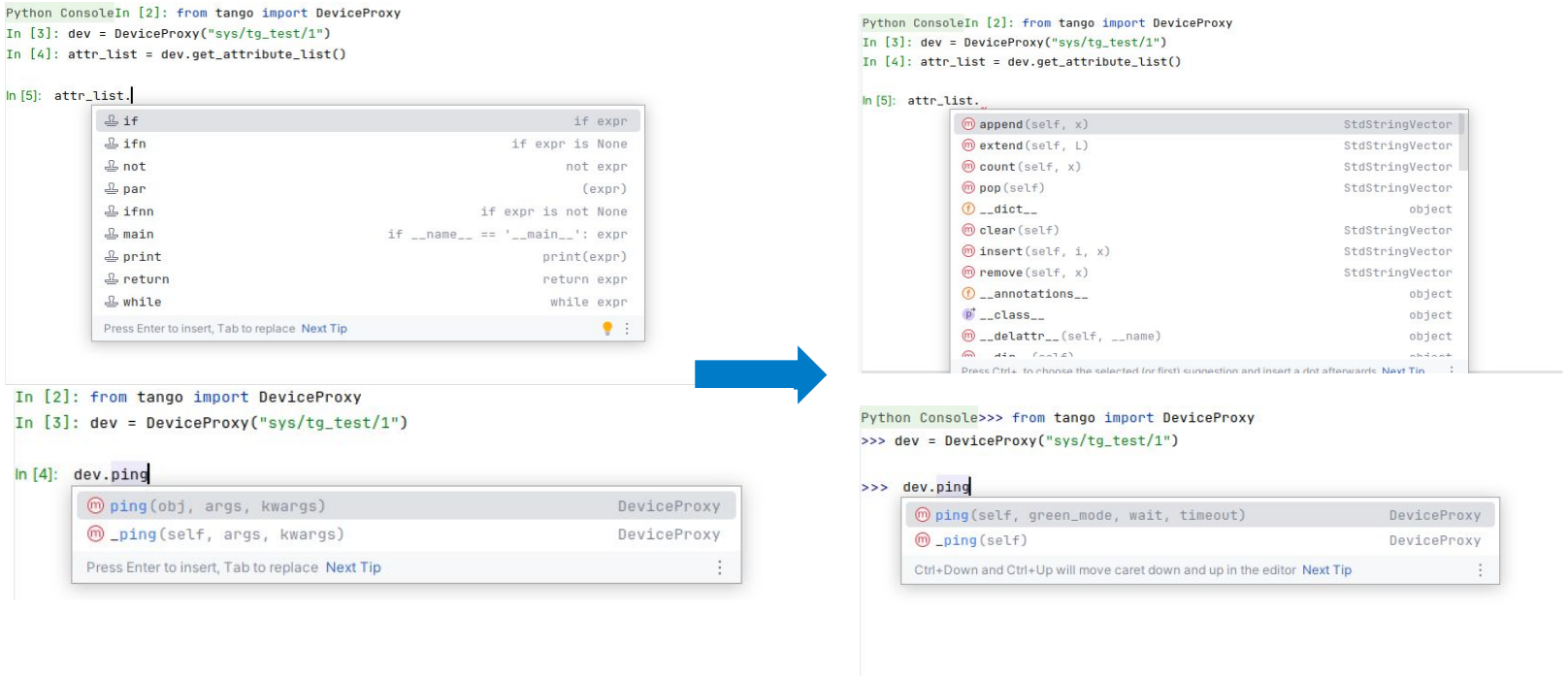
Highlights

- Better pprint of PyTango structures

```
DeviceInfo[
     dev_class = 'PowerSupply'
      dev_type = 'PowerSupply'
       doc_url = 'Doc URL = http://www.tango-controls.org'
   server_host = 'host.domain'
     server_id = 'PowerSupply/test'
server_version = 6
  version_info = {'Build.PyTango.Boost': '1.87.0', 'Build.PyTango.NumPy': '2.1.3', 'Build.PyTango.Python':
```

```
DeviceInfo[
    dev_class = "PowerSupply"
    dev_type = "PowerSupply"
    doc_url = "Doc URL = http://www.tango-controls.org"
    server_host = "host.domain"
    server_id = "PowerSupply/test"
    server_version = 6
    version_info = {
        "Build.PyTango.Boost": "1.87.0",
        "Build.PyTango.NumPy": "2.1.3",
        "Build.PyTango.Python": "3.13.0",
        "Build.PyTango.cppTango": "10.0.2",
        "NumPy": "2.1.3",
        "PyTango": "10.1.0.dev0",
        "Python": "3.13.0",
        "cppTango": "10.0.2",
        "cppTango.git_revision": "unknown",
        "cppzmq": "41000",
        "idl": "6.0.2",
        "omniORB": "4.3.2",
        "opentelemetry-cpp": "1.18.0",
        "zmq": "40305"
    }
]
```

- Better stubs:



Highlights

- Declaring attributes, that emit events programmatically by kwords

```python
from tango.server import Device, attribute

class MyDevice(Device):
    def init_device(self):
        super().init_device()
        self.set_change_event("voltage", implemented=True, detect=False)

    @attribute
    def voltage(self) -> float:
        return 1.23
```

```python
from tango.server import Device, attribute

class MyDevice(Device):

    @attribute(change_event_implemented=True, change_event_detect=False)
    def voltage(self) -> float:
        return 1.23
```

- Command handler docstrings can now populate the in_type_desc and out_type_desc fields

```python
from tango.server import Device, command


class MyDevice(Device):
    @command(doc_in="val_in (int): The input integer value.",
        doc_out="returns (int): An integer result after processing.")
    def example(self, val_in: int) -> int:
        return val_in
```

```python
from tango.server import Device, command

class MyDevice(Device):
        @command
        def example(self, val_in: int) -> int:
            """
            Example command, using Google docstring syntax

            Args:
                val_in (int): The input integer value.

            Returns:
                int: An integer result after processing.
            """
            return val_in
```

Highlights

- The client identity is now available in devices when handling command or attribute read/write calls

```
from tango.server import Device, command
from tango.test_context import DeviceTestContext


class MyDevice(Device):

    @command()
    def cmd(self):
        print(self.get_client_ident())


if __name__ == "__main__":
    with DeviceTestContext(MyDevice, process=True) as dev:
        dev.cmd()
```

```
> python my_device.py
Ready to accept request
CPP or Python client with PID 28680 from host localhost

Process finished with exit code 0
```

- API documentation was almost completely re-worked: unified style + better readability + enabled cross-links were was possible

push_change_event(self, attr_name: str, except: Exception)
push_change_event(self, attr_name: str, data: Any) → None
push_change_event(self, attr_name: str, data: Any, time_stamp: float, quality: AttrQuality) → None
push_change_event(self, attr_name: str, str_data: str, data: Any) → None
push_change_event(self, attr_name: str, str_data: str, data: Any, time_stamp: float, quality: AttrQuality) → None

Push a change event for the given attribute name.

Parameters:

- **attr_name** (`str`) – attribute name
- **data** (`typing.Any`) – the data to be sent as attribute event data. Data must be compatible with the attribute type and format. for SPECTRUM and IMAGE attributes, data can be any type of sequence of elements compatible with the attribute type
- **str_data** (`str`) – special variation for DevEncoded data type. In this case 'data' must be a str or an object with the buffer interface.
- **except** (`Exception` | `DevFailed`) – Instead of data, you may want to send an exception.
- **time_stamp** (`float`) – the time stamp
- **quality** (*AttrQuality*) – the attribute quality factor

Raises:

`DevFailed`: If the attribute data type is not coherent.

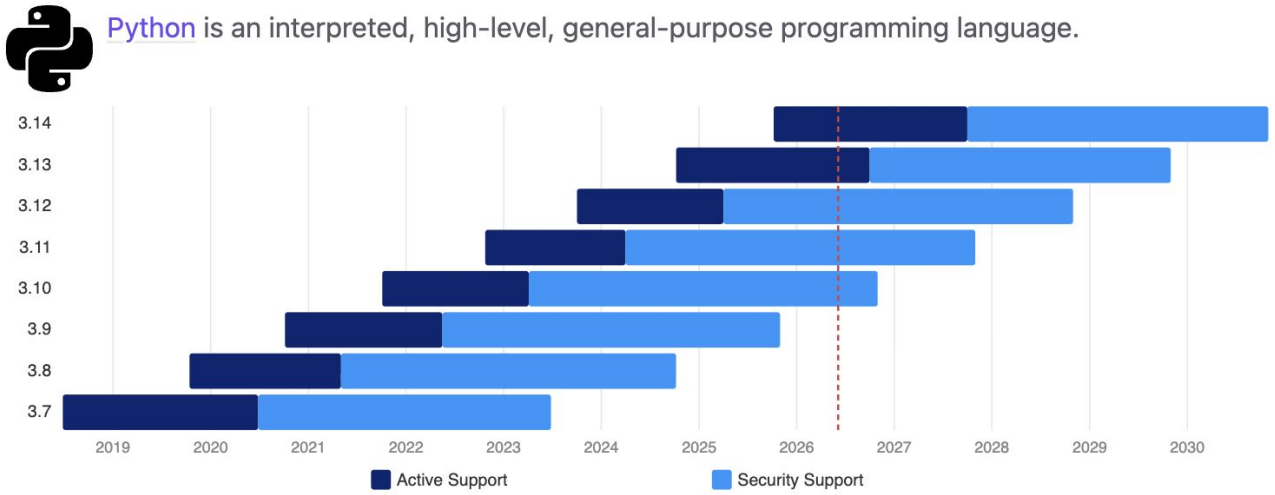
Changed in version 10.1.0: Removed optional 'dim_x' and 'dim_y' arguments. The dimensions are automatically determined from the data.

Next release: 10.4.0

- cppTango 10.4.0 will do a lot of refactoring -> breaking changes in cppTango -> possible also breaking changes in PyTango?

- due to the previous reason can be considerable delayed after cppTango 10.4

- Python 3.10 to 3.15



- asyncio.iscoroutinefunction is deprecated in 3.14 and will be removed in 3.16 . It is used in our copied legacy asyncio methods to convers sync methods into async in runtime. As we promised, as soon as this code breaks again - we will remove it. So 10.4 will be the last release, where sync methods can be used in async servers.

  Update your code!!!

PyTango development

Issues

- Questions: use the Mattermost or TANGO Forum.
- Specific issues: report on GitLab - the more detail the better (ideally, example code).

Contributing

- Please join in!
- Developers' meeting twice a month.
- Typical branched Git workflow. Main branch is develop
- Fork the repo, make it better, make an MR. Thanks!
- More info in how-to-contribute, and our webinar.

Thank you

Contact

Deutsches Elektronen-
Synchrotron DESY

www.desy.de

Yury Matveev
Photon Science Experiment Control Group
yury.matveev@desy.de