9 JUNE 2026

40TH TANGO COMMUNITY MEETING
ALBA, BARCELONA

Runtime configuration for OpenTelemetry

Anton Joubert

MAX IV

Agenda

Introduction

Examples

Runtime configuration

Tango telemetry “topics”

Demo

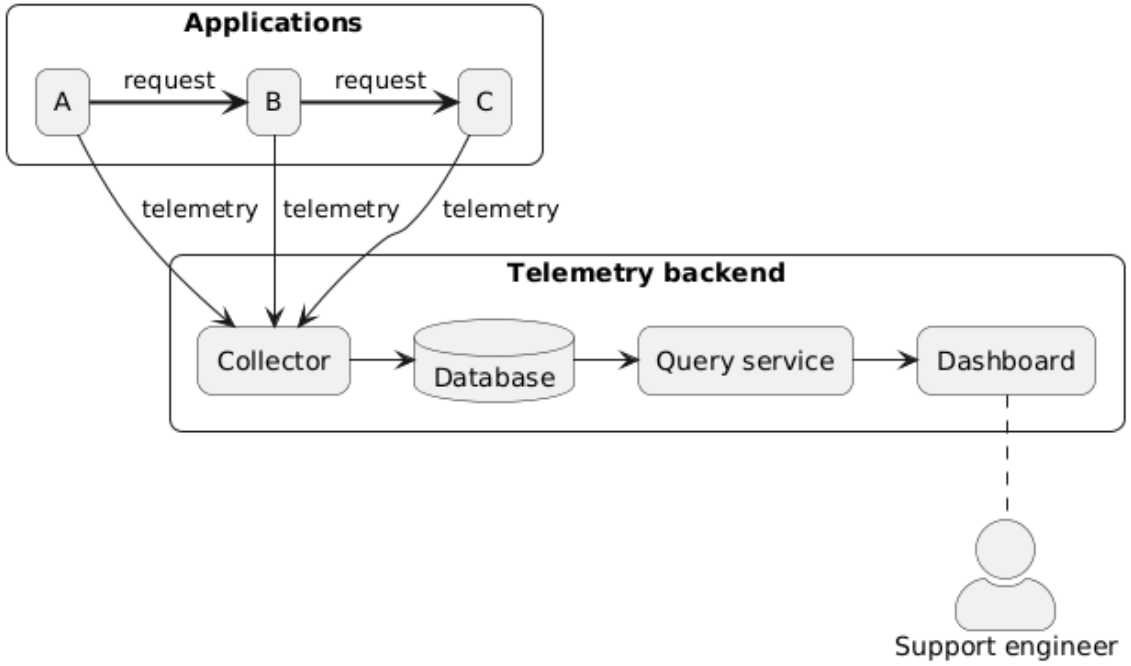
Benchmarks

Conclusion

Introduction

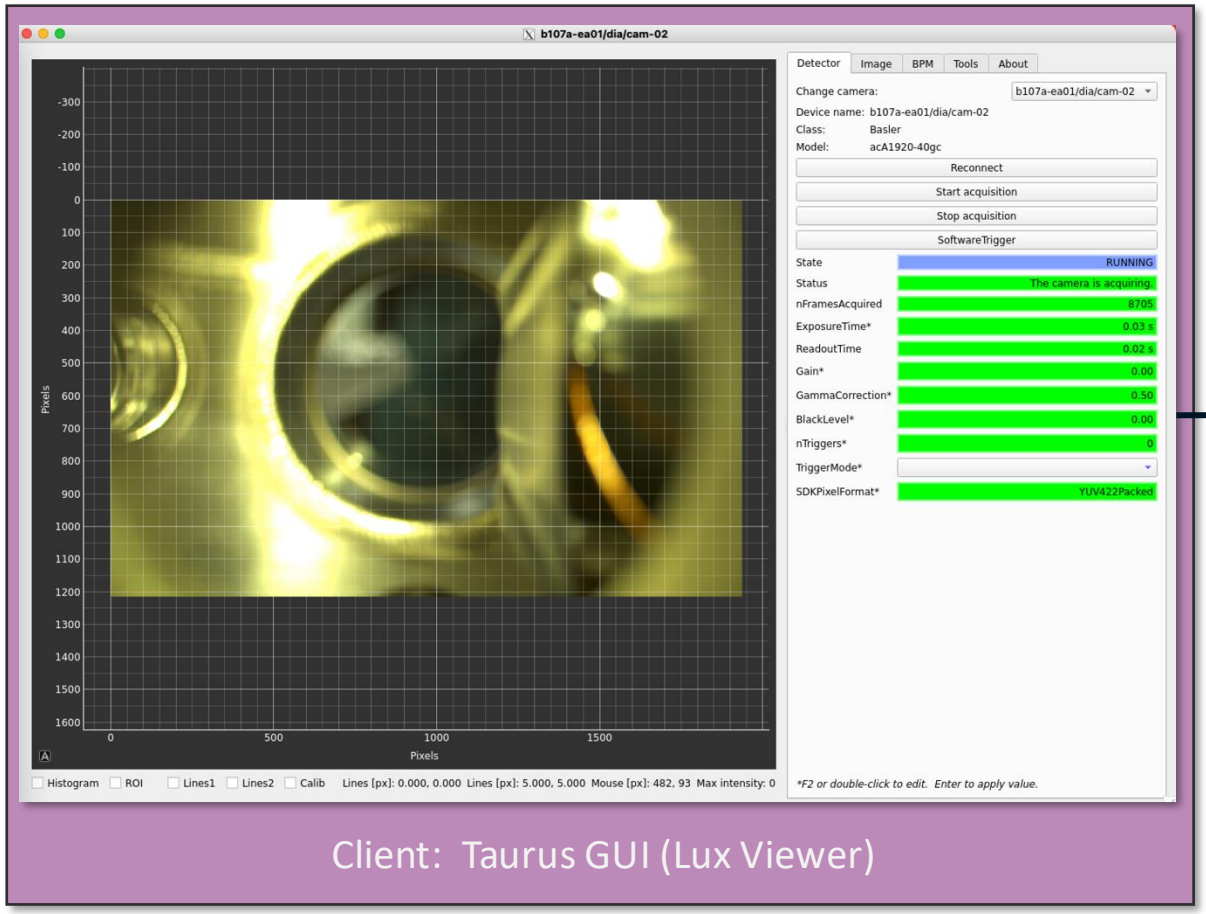
Observability
- Understand a complex system from outside
- Traces, metrics, logs

OpenTelemetry
- Observability framework
- Vendor agnostic
- Many programming languages



Examples

GUI and Tango device for camera

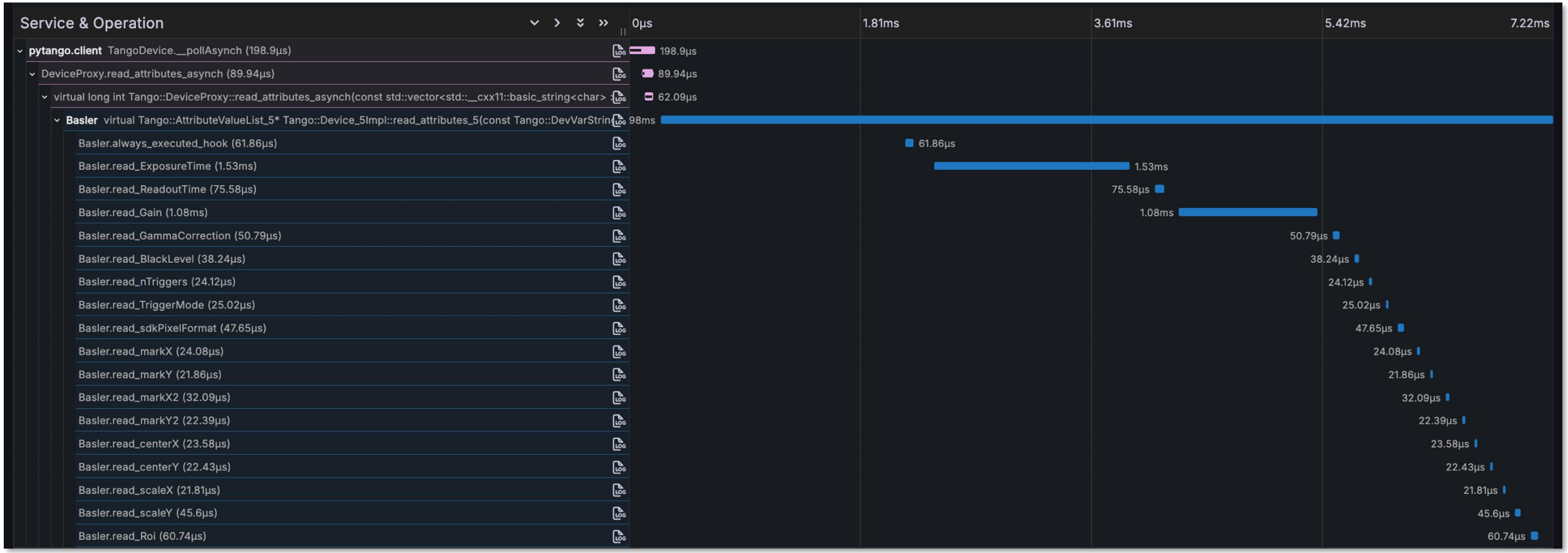


Client: Taurus GUI (Lux Viewer)

Tango device (Basler)

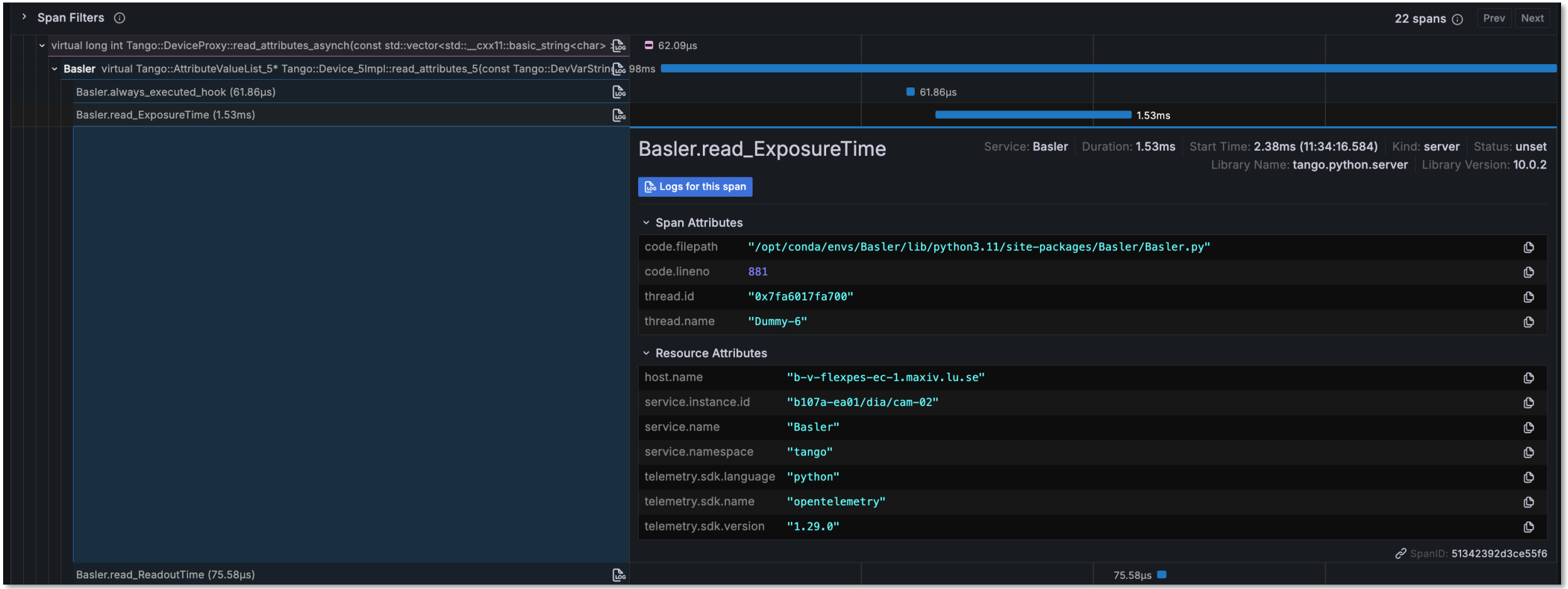
Camera hardware

Trace detail

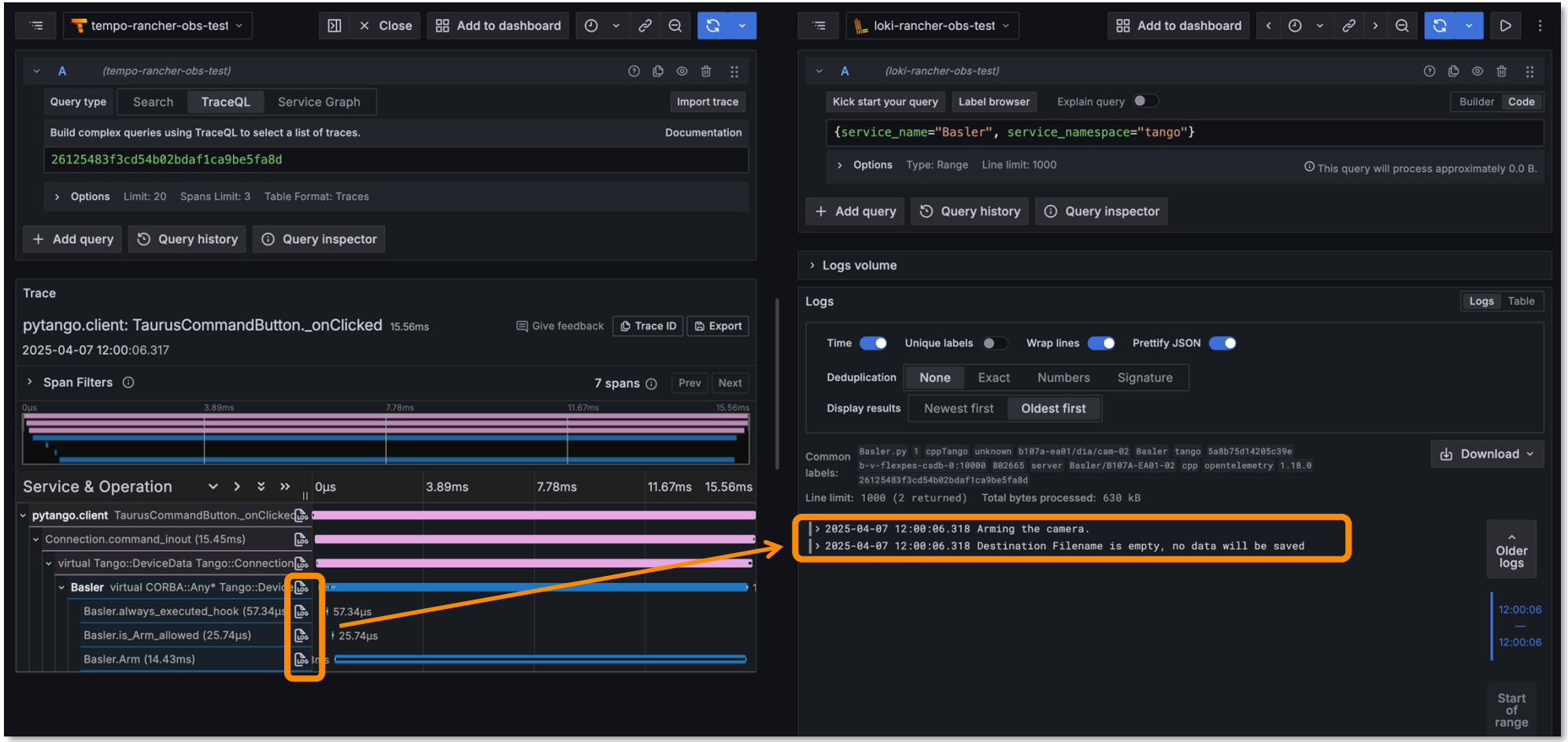


User interface: [Grafana Tempo]

Span detail


Span Filters
22 spans
Prev
Next
virtual long int Tango::DeviceProxy::read_attributes_asynch(const std::vector<std::__cxx11::basic_string<char>
62.09µs
Basler virtual Tango::AttributeValueList_5* Tango::Device_5Impl::read_attributes_5(const Tango::DevVarStrin
98ms
Basler.always_executed_hook (61.86µs)
61.86µs
Basler.read_ExposureTime (1.53ms)
1.53ms
Basler.read_ExposureTime
Service: Basler
Duration: 1.53ms
Start Time: 2.38ms (11:34:16.584)
Kind: server
Status: unset
Library Name: tango.python.server
Library Version: 10.0.2
Logs for this span
Span Attributes
code.filepath "/opt/conda/envs/Basler/lib/python3.11/site-packages/Basler/Basler.py"
code.lineno 881
thread.id "0x7fa6017fa700"
thread.name "Dummy-6"
Resource Attributes
host.name "b-v-flexpes-ec-1.maxiv.lu.se"
service.instance.id "b107a-ea01/dia/cam-02"
service.name "Basler"
service.namespace "tango"
telemetry.sdk.language "python"
telemetry.sdk.name "opentelemetry"
telemetry.sdk.version "1.29.0"
SpanID: 51342392d3ce55f6
Basler.read_ReadoutTime (75.58µs)
75.58µs

Logs


tempo-rancher-obs-test
Close
Add to dashboard
(tempo-rancher-obs-test)
Query type
Search
TraceQL
Service Graph
Import trace
Build complex queries using TraceQL to select a list of traces.
Documentation
26125483f3cd54b02bdaf1ca9be5fa8d
Options
Limit: 20
Spans Limit: 3
Table Format: Traces
Add query
Query history
Query inspector
Trace
pytango.client: TaurusCommandButton._onClicked
15.56ms
Give feedback
Trace ID
Export
2025-04-07 12:00:06.317
Span Filters
7 spans
Prev
Next
0µs
3.89ms
7.78ms
11.67ms
15.56ms
Service & Operation
pytango.client TaurusCommandButton._onClicked
Connection.command_inout (15.45ms)
virtual Tango::DeviceData Tango::Connection
Basler virtual CORBA::Any* Tango::Devic
Basler.always_executed_hook (57.34µs)
57.34µs
Basler.is_Arm_allowed (25.74µs)
25.74µs
Basler.Arm (14.43ms)
loki-rancher-obs-test
Add to dashboard
(loki-rancher-obs-test)
Kick start your query
Label browser
Explain query
Builder
Code
{service_name="Basler", service_namespace="tango"}
Options
Type: Range
Line limit: 1000
This query will process approximately 0.0 B.
Add query
Query history
Query inspector
Logs volume
Logs
Logs
Table
Time
Unique labels
Wrap lines
Prettify JSON
Deduplication
None
Exact
Numbers
Signature
Display results
Newest first
Oldest first
Common labels:
Basler.py 1 cppTango unknown b107a-ea01/dia/cam-02 Basler tango 5a8b75d14205c39e
b-v-flexpes-csdb-0:10000 802665 server Basler/B107A-EA01-02 cpp opentelemetry 1.18.0
26125483f3cd54b02bdaf1ca9be5fa8d
Download
Line limit: 1000 (2 returned)
Total bytes processed: 630 kB
2025-04-07 12:00:06.318 Arming the camera.
2025-04-07 12:00:06.318 Destination Filename is empty, no data will be saved
Older logs
12:00:06
12:00:06
Start of range

Metrics from traces

Runtime configuration

How did it use to work?

Tango 10.0 and 10.1

Set environment variables before startup, for example:

```
TANGO_TELEMETRY_ENABLE=on
TANGO_TELEMETRY_TRACES_EXPORTER=http
TANGO_TELEMETRY_TRACES_ENDPOINT=https://monitoring.institute.org:443/v1/traces
TANGO_TELEMETRY_LOGS_EXPORTER=http
TANGO_TELEMETRY_LOGS_ENDPOINT=https://monitoring.institute.org:443/otlp/v1/logs
```

Can set environment variables in `/etc/tangorc`

How does it work now?

In Tango 10.3.0

Env vars renamed

Old	New
TANGO_TELEMETRY_TRACE**S**_EXPORTER	TANGO_TELEMETRY_TRAC**ING**_EXPORTER**S**
TANGO_TELEMETRY_TRACE**S**_ENDPOINT	TANGO_TELEMETRY_TRAC**ING**_ENDPOINT**S**
TANGO_TELEMETRY_LOG**S**_EXPORTER	TANGO_TELEMETRY_LOG**GING**_EXPORTER**S**
TANGO_TELEMETRY_LOG**S**_ENDPOINT	TANGO_TELEMETRY_LOG**GING**_ENDPOINT**S**

New env vars

```
TANGO_TELEMETRY_TYPES  => tracing,logging | tracing | logging
TANGO_TELEMETRY_TOPICS => all | user,database,events,polling
```

Clients

DeviceProxy, AttributeProxy, Group, Database

Still have to set environment variables, can’t toggle at runtime ☹

```
$ export TANGO_TELEMETRY_ENABLE=on
...
$ python client.py
```

No logging from clients

Servers

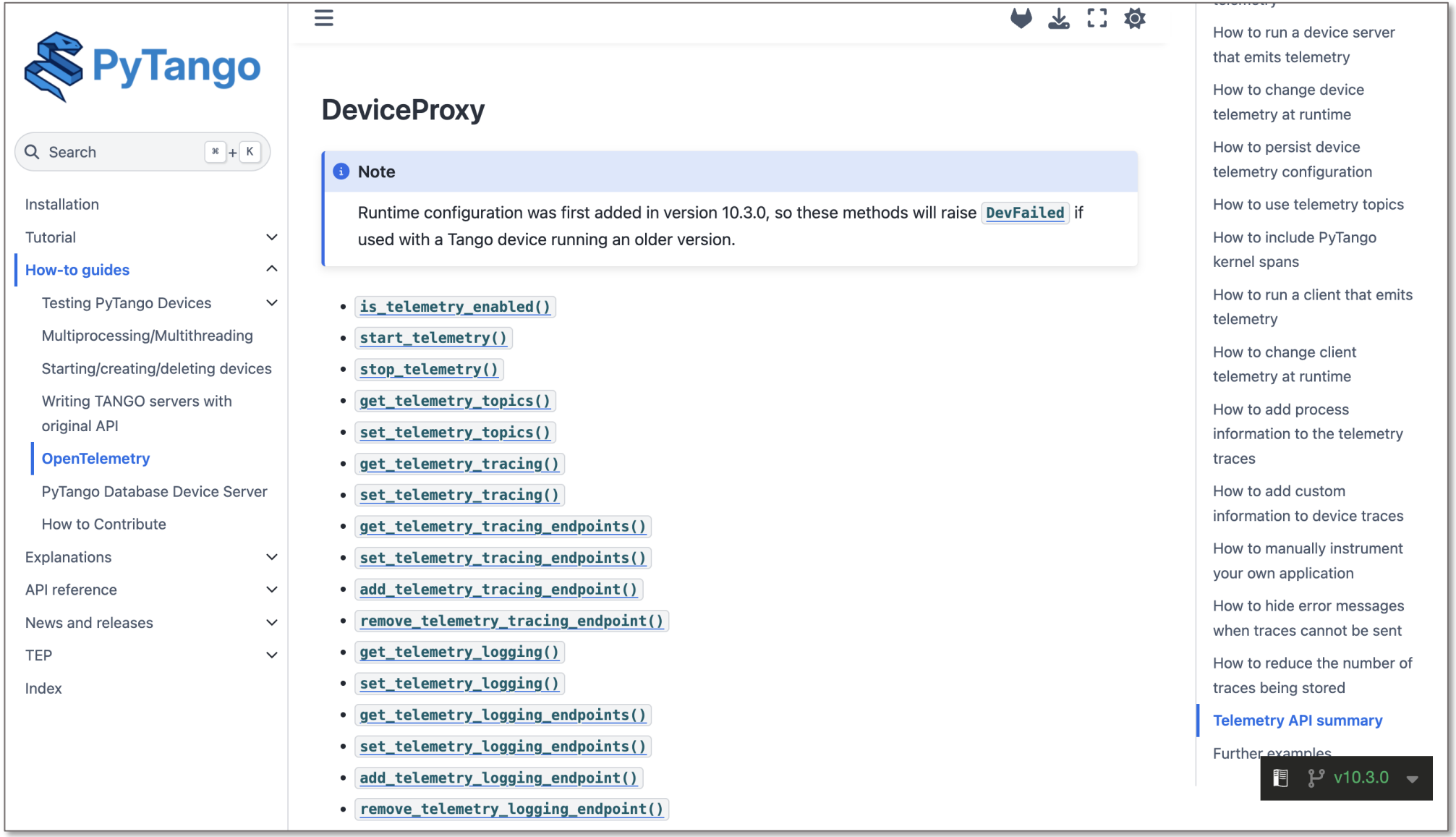
At startup, these are checked, in order:

environment variables
device properties

Can modify at runtime via DServer (admin device) commands, or more conveniently, DeviceProxy helper methods.

API

[Docs]


PyTango
Search ⌘ + K
Installation
Tutorial
How-to guides
Testing PyTango Devices
Multiprocessing/Multithreading
Starting/creating/deleting devices
Writing TANGO servers with original API
OpenTelemetry
PyTango Database Device Server
How to Contribute
Explanations
API reference
News and releases
TEP
Index
DeviceProxy
Note
Runtime configuration was first added in version 10.3.0, so these methods will raise DevFailed if used with a Tango device running an older version.
is_telemetry_enabled()
start_telemetry()
stop_telemetry()
get_telemetry_topics()
set_telemetry_topics()
get_telemetry_tracing()
set_telemetry_tracing()
get_telemetry_tracing_endpoints()
set_telemetry_tracing_endpoints()
add_telemetry_tracing_endpoint()
remove_telemetry_tracing_endpoint()
get_telemetry_logging()
set_telemetry_logging()
get_telemetry_logging_endpoints()
set_telemetry_logging_endpoints()
add_telemetry_logging_endpoint()
remove_telemetry_logging_endpoint()
How to run a device server that emits telemetry
How to change device telemetry at runtime
How to persist device telemetry configuration
How to use telemetry topics
How to include PyTango kernel spans
How to run a client that emits telemetry
How to change client telemetry at runtime
How to add process information to the telemetry traces
How to add custom information to device traces
How to manually instrument your own application
How to hide error messages when traces cannot be sent
How to reduce the number of traces being stored
Telemetry API summary
Further examples
v10.3.0

```
(tangomeeting2026-livedemos:test) ➜  tangods_monny git:(f14c468) ✗ TANGO_TELEMETRY_TRACING_EXPORTERS=http TANGO_TELEMETRY_LOGGING_EXPORTERS=http MonnyDS demo -v3
2026-06-05T19:45:07,701454+0200 DEBUG (device.py:29) demo/rr/1 -> ResourceReporter.init_device()
2026-06-05T19:45:07,718236+0200 DEBUG (device.py:29) demo/rr/1 <- ResourceReporter.init_device()
Ready to accept request
```

python (python)

```
>>> import tango
>>> dp = tango.DeviceProxy("demo/rr/1")
>>> dp.is_telemetry_enabled()
False
>>> dp.start_telemetry()
>>> dp.is_telemetry_enabled()
True
>>> dp.stop_telemetry()
>>> dp.is_telemetry_enabled()
False
>>> dp.get_telemetry_tracing()
True
>>> dp.get_telemetry_logging()
True
>>> dp.get_telemetry_tracing_endpoints()
(TelemetryEndpoint(exporter=<TelemetryExporter.HTTP: 1>, endpoint='http://localhost:4318/v1/traces'),)
>>> dp.get_telemetry_logging_endpoints()
(TelemetryEndpoint(exporter=<TelemetryExporter.HTTP: 1>, endpoint='http://localhost:4318/v1/logs'),)
>>> dp.get_telemetry_topics()
('all',)
>>>
```

Recommendation

Set these in `/etc/tangorc` or in your environment:

```
TANGO_TELEMETRY_TRACING_EXPORTERS
TANGO_TELEMETRY_TRACING_ENDPOINTS
TANGO_TELEMETRY_LOGGING_EXPORTERS
TANGO_TELEMETRY_LOGGING_ENDPOINTS
```

Then you can easily start and stop telemetry with a single DeviceProxy method.

Tango telemetry “topics”

What are topics?

A way to limit the number of spans, a bit like logging severity

Set independently on each server and client

v10.3.0 options:

```
user, database, events, polling
```

meta-topics: `all *`

v10.4.0 proposed:

```
user, configuration, pub_sub, request_reply, lifecycle, polling, audit, misc
```

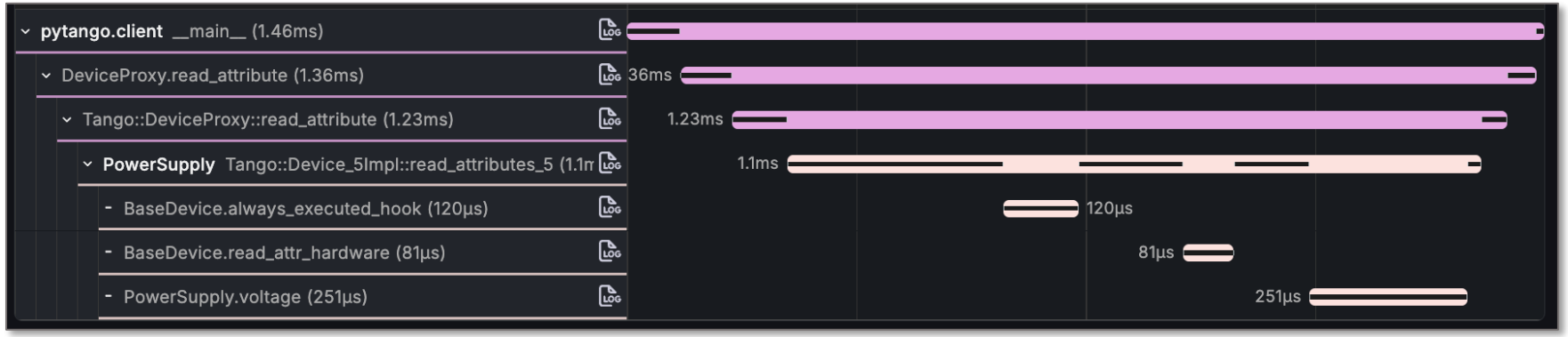
meta-topics: `all, none`

* the only useful topic in v10.3

See RFC-19 merge request

Examples

topic: `all`
client + server



topic: `user`
client + server
(broken in v10.3*)

topic: `user`
client + server
(expected in v10.4)

* See [cppTango issue 1645]

When will a span/log be emitted to OTel?

```
def on_span(span):
  if telemetry_enabled and telemetry_tracing_enabled:
    if span.topic in enabled_topics:
      emit_span(span)
```

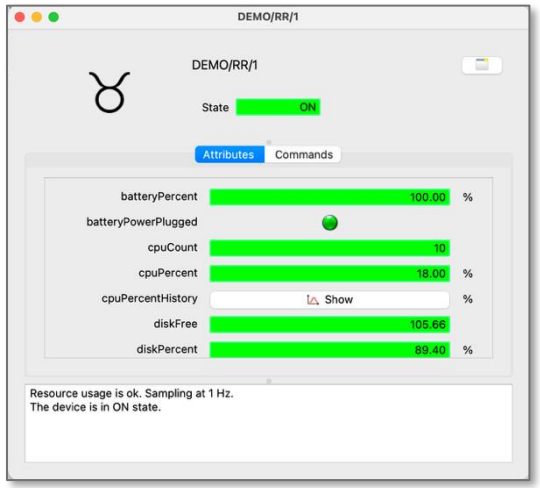
```
def on_log_message(log):
  if telemetry_enabled and telemetry_logging_enabled:
    if log.level <= logging_level:
      emit_log(log)
```

Demo

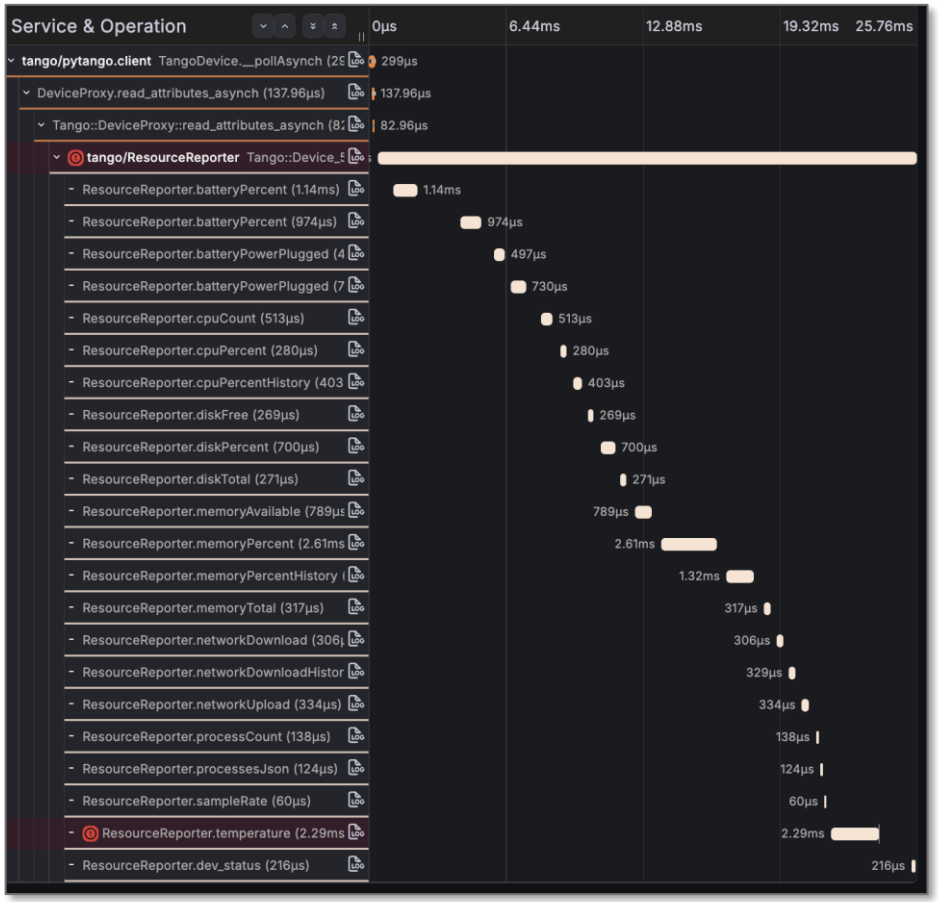
Using ResourceReporter device before it pushed events.

```
(tangomeeting2026-livedemos:test) ➜  tangods_monny git:(main) TANGO_TELEMETRY_TRACING_EXPORTERS=http TANGO_TELEMETRY_LOGGING_EXPORTERS=http MonnyDS demo -v3
2026-06-07T10:02:49,606172+0200 DEBUG (device.py:29) demo/rr/1 -> ResourceReporter.init_device()
2026-06-07T10:02:49,619949+0200 DEBUG (device.py:29) demo/rr/1 <- ResourceReporter.init_device()
Ready to accept request
```

```
(taurus-py314) ➜  ~ TANGO_TELEMETRY_ENABLE=on TANGO_TELEMETRY_TRACING_EXPORTERS=http taurus device demo/rr/1
```



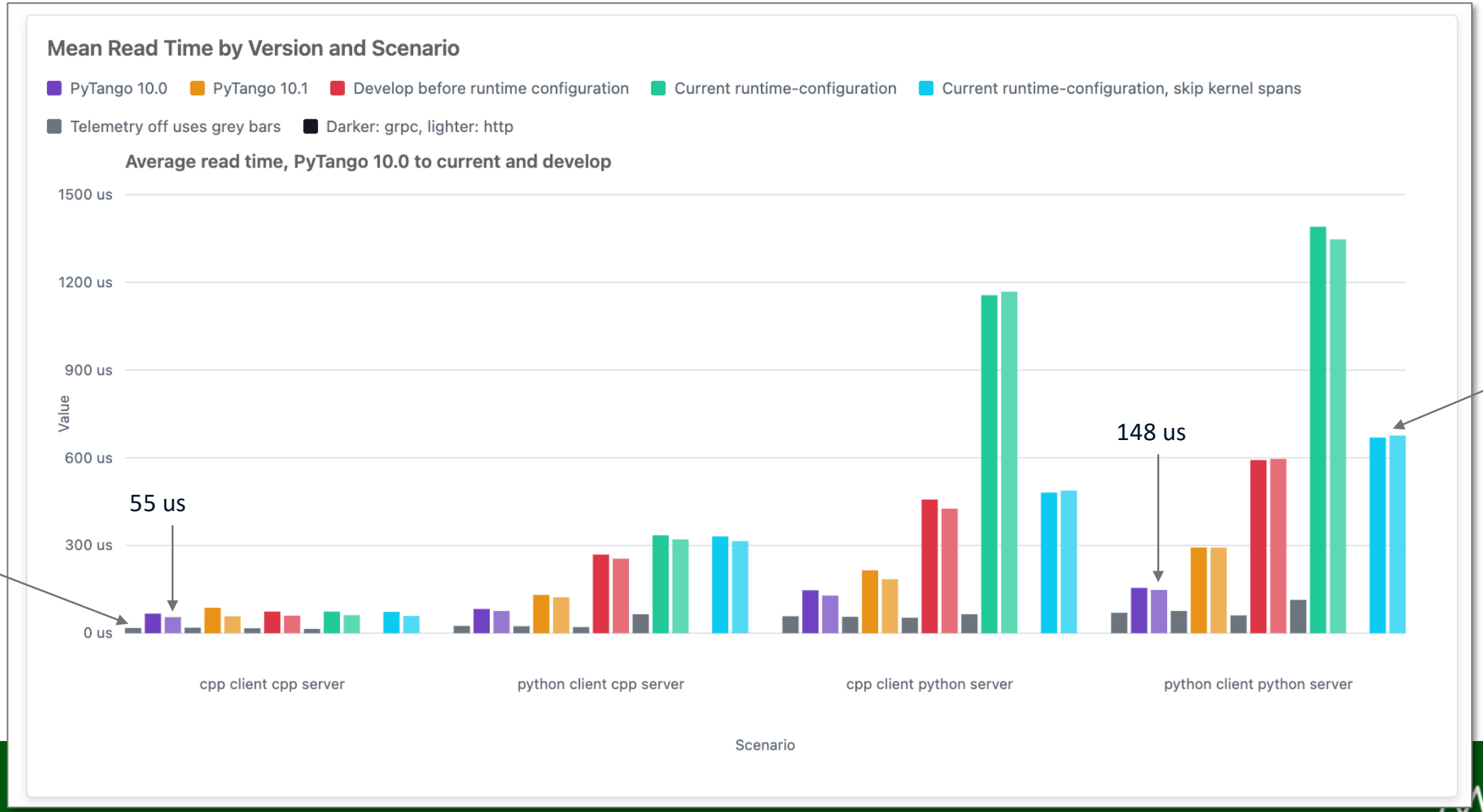

```
(pytango-stable) ➜  ~ TANGO_TELEMETRY_ENABLE=on TANGO_TELEMETRY_TRACING_EXPORTERS=http python
Python 3.14.0 | packaged by conda-forge | (main, Oct 22 2025, 23:45:45) [Clang 20.1.8 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tango
>>> dp = tango.DeviceProxy("demo/rr/1")
>>> dp.is_telemetry_enabled()
False
>>> dp.start_telemetry()
>>> dp.is_telemetry_enabled()
True
>>> dp.stop_telemetry()
```

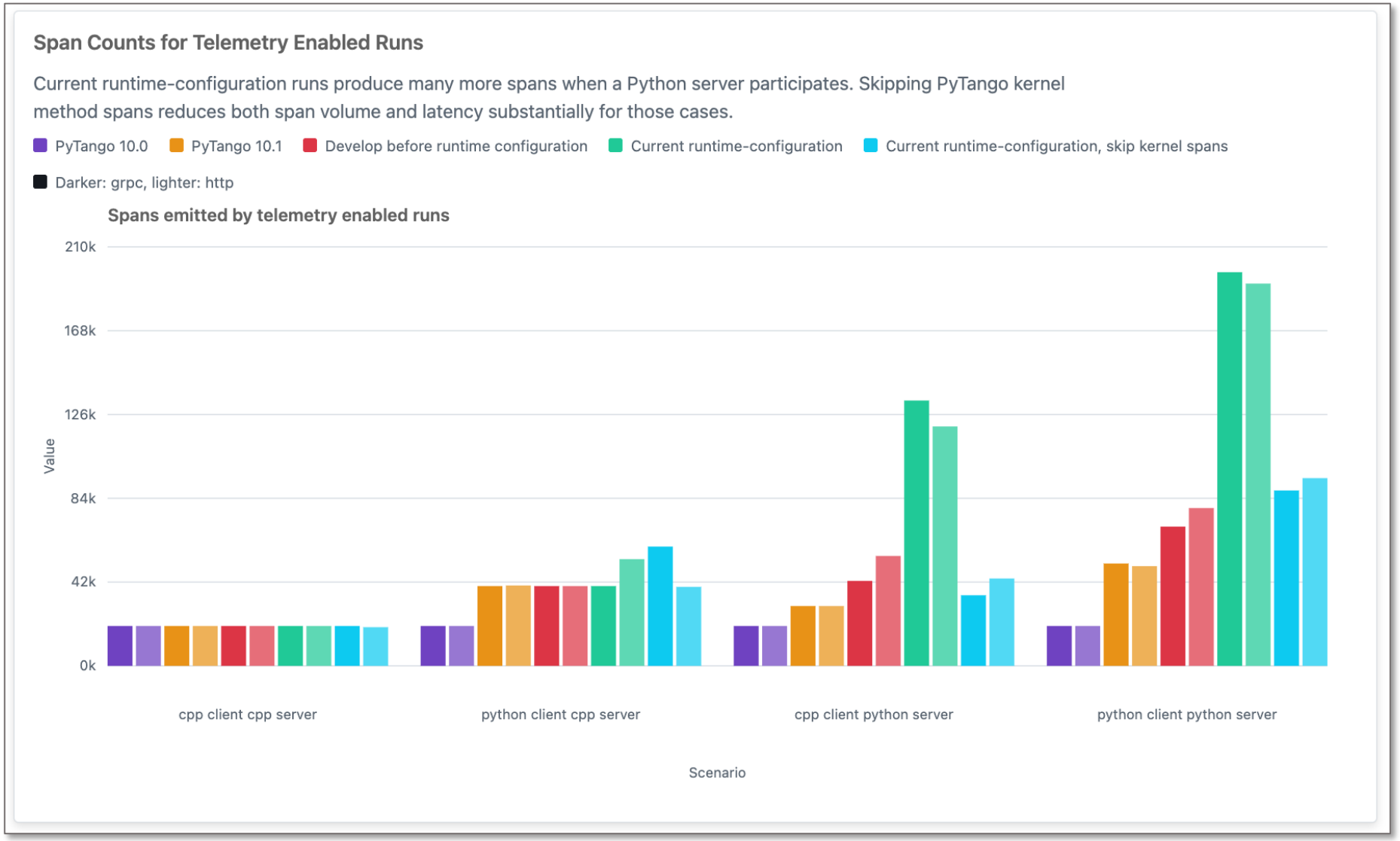


Benchmarks

PyTango Telemetry Benchmark Analysis - 2026-05-13

Results from 100 iterations with 300 reads per iteration, using `TOPICS=all`. Times are per read in microseconds. Span counts are rounded to thousands as reported by the runner. Run on a M1 MacBook Pro. Git hash: d84d6bb923. Extension built in RelWithDebInfo mode.





More details in [benchmark_analysis.html]() and [repo]()

Conclusion

Conclusion

Servers: runtime configuration is easy

Clients: runtime configuration unsolved

Topics: will be useful in v10.4

Performance: investigate regression in 10.3

Useful links

PyTango documentation

https://tango-controls.readthedocs.io/projects/pytango/en/latest/how-to/telemetry.html

Tools

https://grafana.com/docs/opentelemetry/docker-lgtm/

ICALEPCS paper

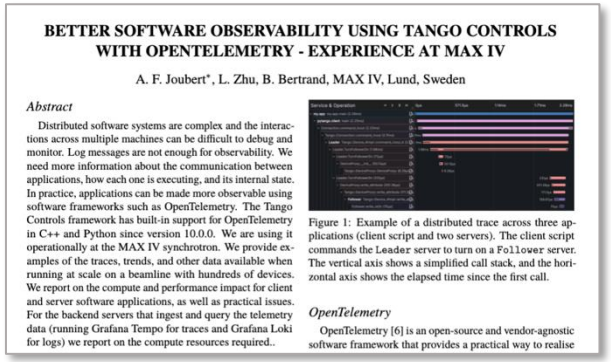
https://proceedings.jacow.org/icalepcs2025/pdf/WEAG006.pdf

Observability and OpenTelemetry

https://opentelemetry.io/docs/concepts/observability-primer/

https://opentelemetry.io/docs/what-is-opentelemetry/

https://opentelemetry.io/docs/concepts/sampling/



BETTER SOFTWARE OBSERVABILITY USING TANGO CONTROLS WITH OPENTELEMETRY - EXPERIENCE AT MAX IV

A. F. Joubert*, L. Zhu, B. Bertrand, MAX IV, Lund, Sweden

Abstract

Distributed software systems are complex and the interactions across multiple machines can be difficult to debug and monitor. Log messages are not enough for observability. We need more information about the communication between applications, how each one is executing, and its internal state. In practice, applications can be made more observable using software frameworks such as OpenTelemetry. The Tango Controls framework has built-in support for OpenTelemetry in C++ and Python since version 10.0.0. We are using it operationally at the MAX IV synchrotron. We provide examples of the traces, trends, and other data available when running at scale on a beamline with hundreds of devices. We report on the compute and performance impact for client and server software applications, as well as practical issues. For the backend servers that ingest and query the telemetry data (running Grafana Tempo for traces and Grafana Loki for logs) we report on the compute resources required..

Figure 1: Example of a distributed trace across three applications (client script and two servers). The client script commands the `Leader` server to turn on a `Follower` server. The vertical axis shows a simplified call stack, and the horizontal axis shows the elapsed time since the first call.

OpenTelemetry

OpenTelemetry [6] is an open-source and vendor-agnostic software framework that provides a practical way to realise

Thank you!

Anton Joubert
anton.joubert@maxiv.lu.se