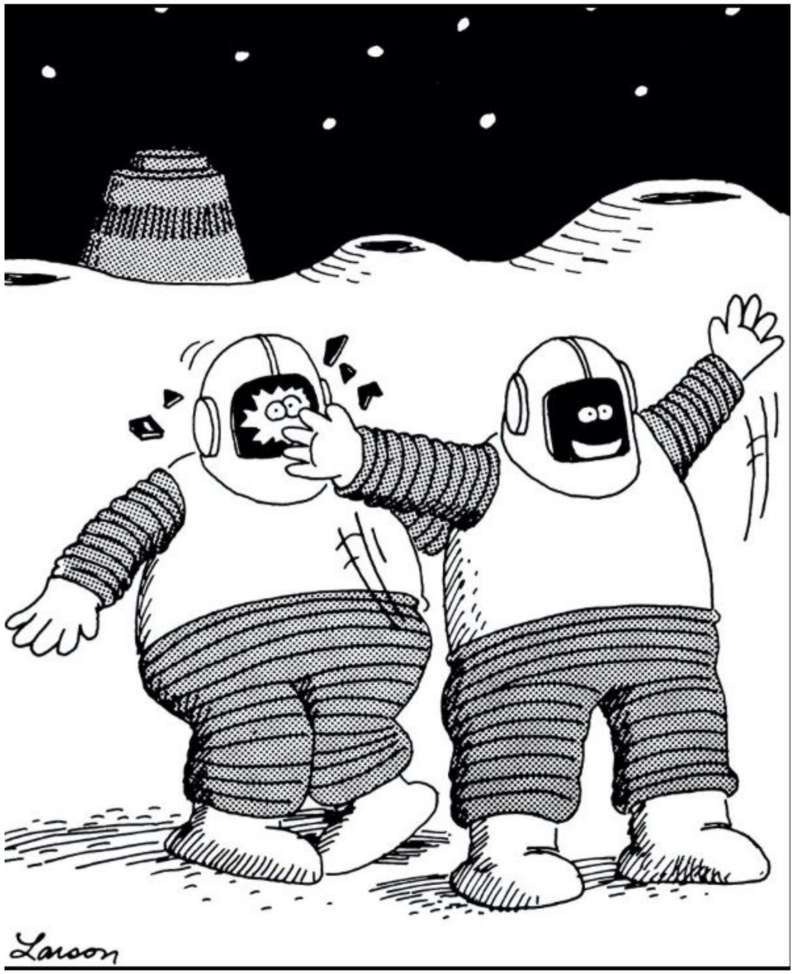
Tango Controls v10/IDLv6 (and what is next)

It has arrived -
Tango v10/IDLv6 is (almost) here!

Presenters:
Thomas Juerges (SKAO)
Anton Joubert (MAX IV)

Tango Controls IDL: What is it?



"We've made it, Community! IDLv6!

- Tango Controls IDL: Describes the generic inter-system interface for Tango Controls components

- Defines everything that is
  - allowed,
  - not allowed,
  - possible and
  - not possible

  in a Tango Controls System

- Decision to modify IDL from v6 on more often:
  - Smaller changes
  - Lower impact

Tango IDLv5

10 years ago, in a galaxy far, far away…

commit:

```
commit a249b067affb01e0ee2071503504cd661bf07f6e
Author: Emmanuel Taurel <taurel@users.noreply.github.com>
Date:   2014-08-27 15:08:08 +0000

    - Added set_pipe_config in Device_5
```

Tango Controls IDLv6: New features

- New alarm event
- New DevInfo_6 with version information
- Distributed tracing support and enhanced logging
- ~~Warning and alarm hysteresis~~

Tango Controls IDLv6: New alarm event

- What is it? A new event: ALARM

- Why do we need it?
  - Allows clients to just subscribe to events when attributes cross the ALARM quality factor threshold
  - Removes need to subscribe and subsequently filter out unwanted events on every(!) subscriber

- What is the catch?
  - If you want to disable pushing of alarm events when a user calls push change event.
    Set free property: *AutoAlarmOnChangeEvent* to false.

- When? 10.0.0 release

Tango Controls IDLv6: New DevInfo_6 with version information

- What is it? Extended DevInfo_5, carries version information
  - sequence<(string name, string value)> version_info
  - Mandatory: library version (tango), IDL version, implementation version (pytango, java)
  - Optional: source version, whatever you like

- Why do we need it?
  - Deployment and configuration traceability
  - Ensuring compatibility, Maintenance, Debugging
- What is the catch?
  - No catch

- When? 10.0.0 release

Tango Controls IDLv6: Distributed tracing support and enhanced logging

- What is it? Tracing and logging support for OpenTelemetry

- Why do we need it?
  - Allows tracing of remote calls through distributed system
  - Associate logs with remote calls
  - Profiling / monitoring

- What is the catch?
  - Tiny overhead
    - BUT: Can be disabled

- When? 10.0.0 release

Tango Controls IDLv6: Deprecating notifd

- What is it?
  - Very old way of getting events.
- Why don't we want it?
  - ZeroMQ is better
  - We don't have any tests for it, and it has very little usage
- Plan:
  - Kernel: Warn of deprecation in 10.0.0
  - Kernel: Remove in next major release, 11.0.0
  - You: Don't use it in any new development!

ROADMAP: time to unlock "gradual" change!!

Tango v10

- New alarm event
- Enhanced logging system and distributed tracing support
- New DevInfo_6 with version information

Tango v11

- Warning and alarm hysteresis
- New datatype DevDict (and deprecate DevPipes)

Tango v12

- Multi-parameter commands

Tango v13

- Multi-dimensional arrays

Is v10 ready?

cppTango

- Release candidate rc1 was made today

PyTango

- Release candidate rc1 “soon”

JTango

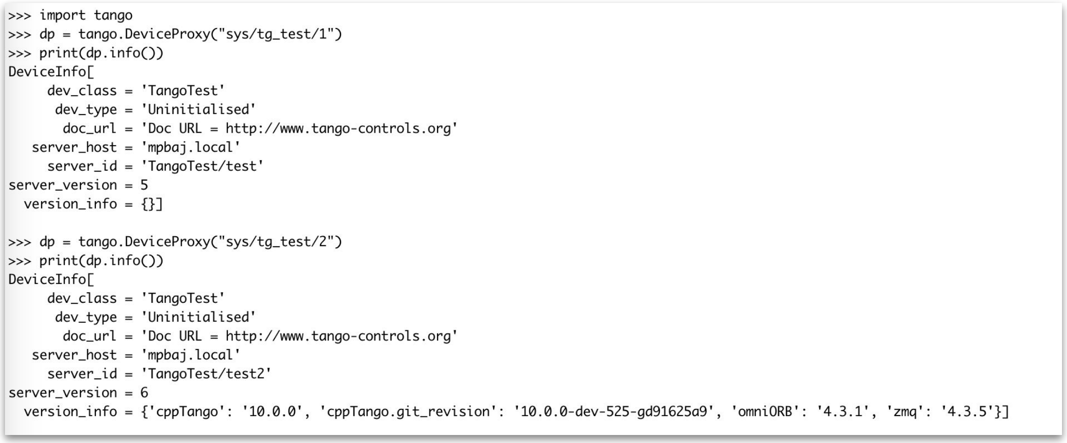
- Aiming for rc1 in September 2024

DEMO 1!

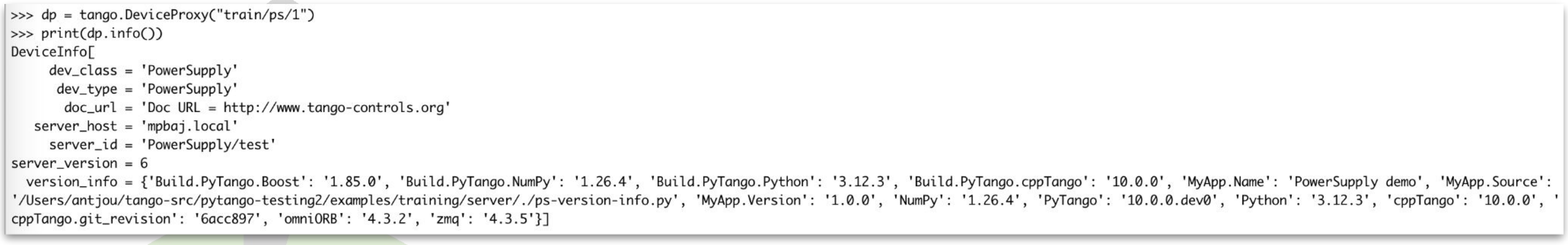
- Version Info
- Alarm events

Image credit: Generated with AI
(which can’t spell “demo” twice…)

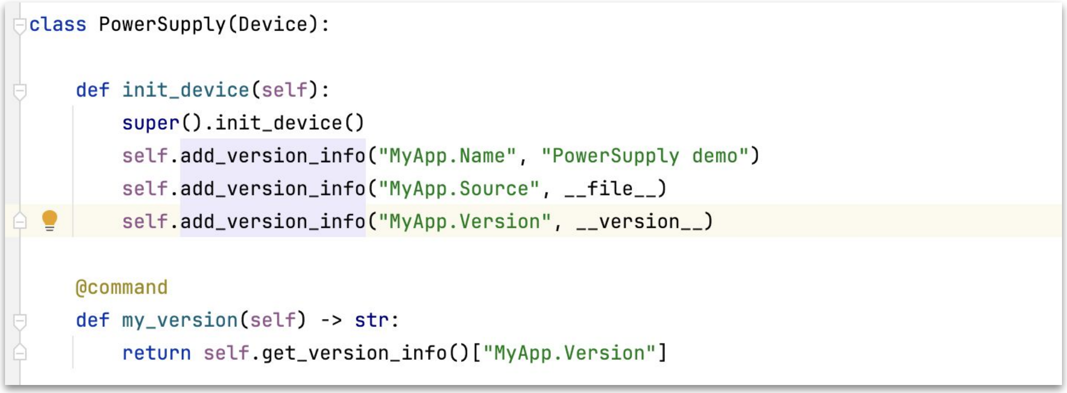
Device version info



```
>>> import tango
>>> dp = tango.DeviceProxy("sys/tg_test/1")
>>> print(dp.info())
DeviceInfo[
     dev_class = 'TangoTest'
      dev_type = 'Uninitialised'
       doc_url = 'Doc URL = http://www.tango-controls.org'
   server_host = 'mpbaj.local'
     server_id = 'TangoTest/test'
server_version = 5
  version_info = {}]

>>> dp = tango.DeviceProxy("sys/tg_test/2")
>>> print(dp.info())
DeviceInfo[
     dev_class = 'TangoTest'
      dev_type = 'Uninitialised'
       doc_url = 'Doc URL = http://www.tango-controls.org'
   server_host = 'mpbaj.local'
     server_id = 'TangoTest/test2'
server_version = 6
  version_info = {'cppTango': '10.0.0', 'cppTango.git_revision': '10.0.0-dev-525-gd91625a9', 'omniORB': '4.3.1', 'zmq': '4.3.5'}]
```

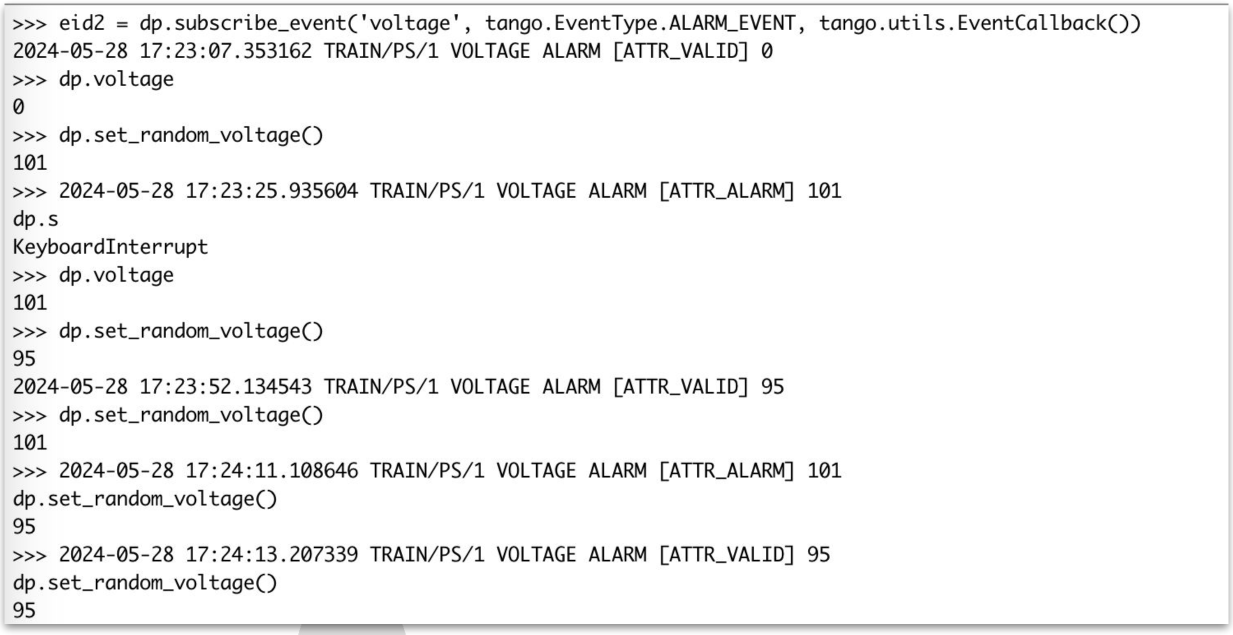


```
>>> dp = tango.DeviceProxy("train/ps/1")
>>> print(dp.info())
DeviceInfo[
     dev_class = 'PowerSupply'
      dev_type = 'PowerSupply'
       doc_url = 'Doc URL = http://www.tango-controls.org'
   server_host = 'mpbaj.local'
     server_id = 'PowerSupply/test'
server_version = 6
  version_info = {'Build.PyTango.Boost': '1.85.0', 'Build.PyTango.NumPy': '1.26.4', 'Build.PyTango.Python': '3.12.3', 'Build.PyTango.cppTango': '10.0.0', 'MyApp.Name': 'PowerSupply demo', 'MyApp.Source':
'/Users/antjou/tango-src/pytango-testing2/examples/training/server/./ps-version-info.py', 'MyApp.Version': '1.0.0', 'NumPy': '1.26.4', 'PyTango': '10.0.0.dev0', 'Python': '3.12.3', 'cppTango': '10.0.0', '
cppTango.git_revision': '6acc897', 'omniORB': '4.3.2', 'zmq': '4.3.5'}]
```

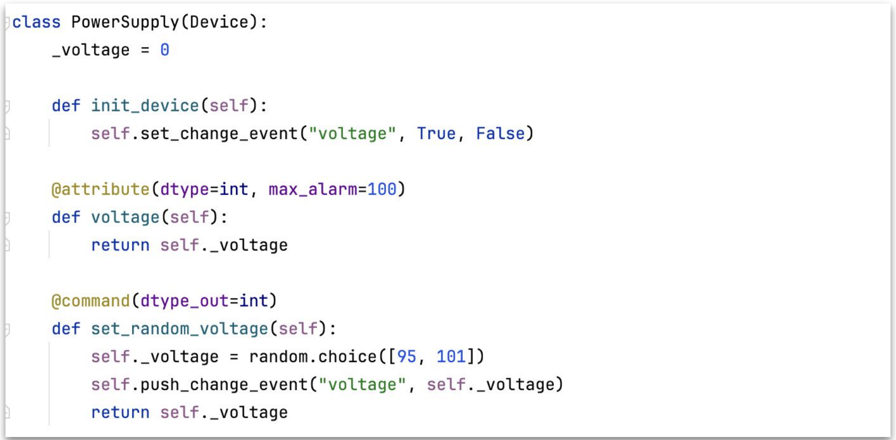


```python
class PowerSupply(Device):

    def init_device(self):
        super().init_device()
        self.add_version_info("MyApp.Name", "PowerSupply demo")
        self.add_version_info("MyApp.Source", __file__)
        self.add_version_info("MyApp.Version", __version__)

    @command
    def my_version(self) -> str:
        return self.get_version_info()["MyApp.Version"]
```

Alarm event from device pushing change event

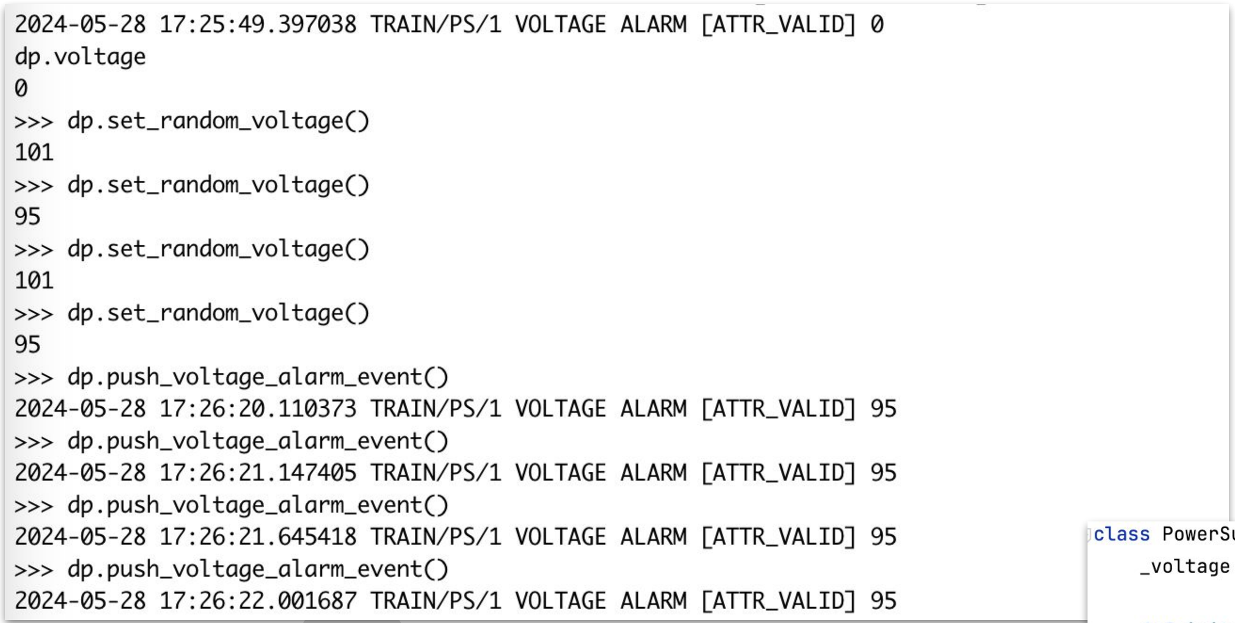


```
>>> eid2 = dp.subscribe_event('voltage', tango.EventType.ALARM_EVENT, tango.utils.EventCallback())
2024-05-28 17:23:07.353162 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 0
>>> dp.voltage
0
>>> dp.set_random_voltage()
101
>>> 2024-05-28 17:23:25.935604 TRAIN/PS/1 VOLTAGE ALARM [ATTR_ALARM] 101
dp.s
KeyboardInterrupt
>>> dp.voltage
101
>>> dp.set_random_voltage()
95
2024-05-28 17:23:52.134543 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 95
>>> dp.set_random_voltage()
101
>>> 2024-05-28 17:24:11.108646 TRAIN/PS/1 VOLTAGE ALARM [ATTR_ALARM] 101
dp.set_random_voltage()
95
>>> 2024-05-28 17:24:13.207339 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 95
dp.set_random_voltage()
95
```

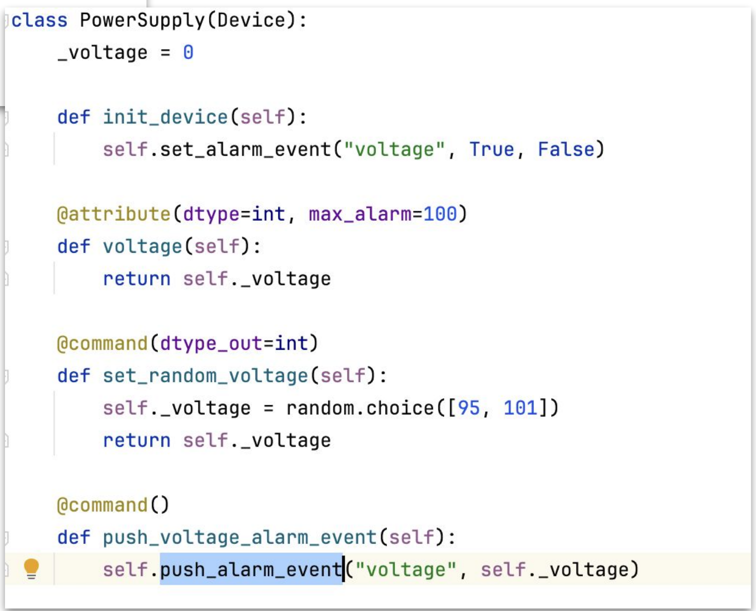


```python
class PowerSupply(Device):
    _voltage = 0

    def init_device(self):
        self.set_change_event("voltage", True, False)

    @attribute(dtype=int, max_alarm=100)
    def voltage(self):
        return self._voltage

    @command(dtype_out=int)
    def set_random_voltage(self):
        self._voltage = random.choice([95, 101])
        self.push_change_event("voltage", self._voltage)
        return self._voltage
```

Alarm event from device pushing alarm event



```
2024-05-28 17:25:49.397038 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 0
dp.voltage
0
>>> dp.set_random_voltage()
101
>>> dp.set_random_voltage()
95
>>> dp.set_random_voltage()
101
>>> dp.set_random_voltage()
95
>>> dp.push_voltage_alarm_event()
2024-05-28 17:26:20.110373 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 95
>>> dp.push_voltage_alarm_event()
2024-05-28 17:26:21.147405 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 95
>>> dp.push_voltage_alarm_event()
2024-05-28 17:26:21.645418 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 95
>>> dp.push_voltage_alarm_event()
2024-05-28 17:26:22.001687 TRAIN/PS/1 VOLTAGE ALARM [ATTR_VALID] 95
```



```python
class PowerSupply(Device):
    _voltage = 0

    def init_device(self):
        self.set_alarm_event("voltage", True, False)

    @attribute(dtype=int, max_alarm=100)
    def voltage(self):
        return self._voltage

    @command(dtype_out=int)
    def set_random_voltage(self):
        self._voltage = random.choice([95, 101])
        return self._voltage

    @command()
    def push_voltage_alarm_event(self):
        self.push_alarm_event("voltage", self._voltage)
```

Distributed tracing support

- Observability
- OpenTelemetry
- Viewing the data
- Changes in cppTango and PyTango
- Performance impact
- Sampling

Observability

- Understand a complex system from outside by examining outputs.

  Helps to troubleshoot novel problems:
  “why is this happening?”

- Application code must be *instrumented* to emit signals:
  - **Traces**

    The path taken to handle a request to the application.
  - **Metrics**

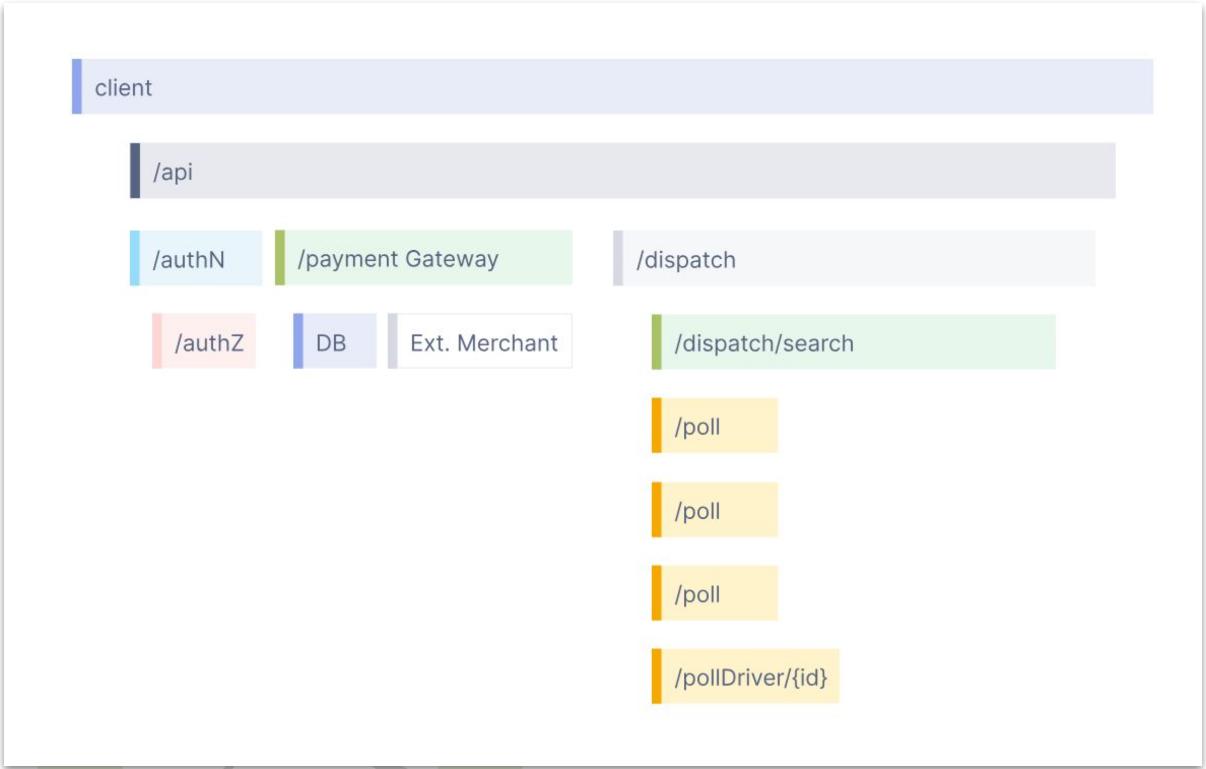
    Measurement captured at runtime (counter, gauge, histogram)
  - **Logs**

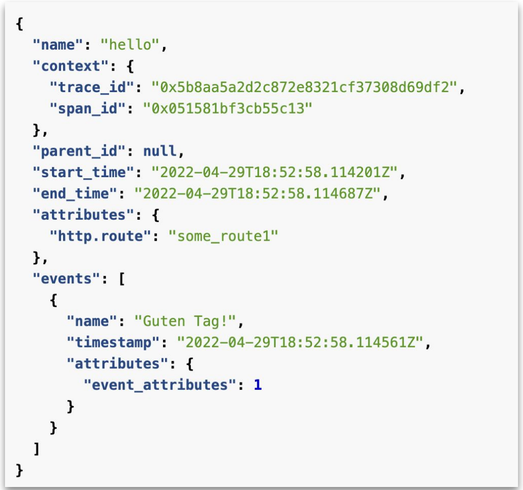
    Time-stamped text record

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

Distributed trace

- Made up of one of more spans
- Span: unit of work, tracking a specific operation





```
{
  "name": "hello",
  "context": {
    "trace_id": "0x5b8aa5a2d2c872e8321cf37308d69df2",
    "span_id": "0x051581bf3cb55c13"
  },
  "parent_id": null,
  "start_time": "2022-04-29T18:52:58.114201Z",
  "end_time": "2022-04-29T18:52:58.114687Z",
  "attributes": {
    "http.route": "some_route1"
  },
  "events": [
    {
      "name": "Guten Tag!",
      "timestamp": "2022-04-29T18:52:58.114561Z",
      "attributes": {
        "event_attributes": 1
      }
    }
  ]
}
```

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

OpenTelemetry

- A tool used to *instrument* software applications so that they are more observable:

  an observability framework

- Vendor and tool agnostic
- A Cloud Native Computing Foundation (CNCF) project
- Includes:
  - Specifications
  - Semantic conventions
  - Application Programmer Interfaces
  - Software Development Kits in many languages
- Integrated into many free and commercial tools.

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

Viewing data

- Signoz
- Jaeger
- Elasticsearch
- Grafana Tempo
- Zipkin
- More…

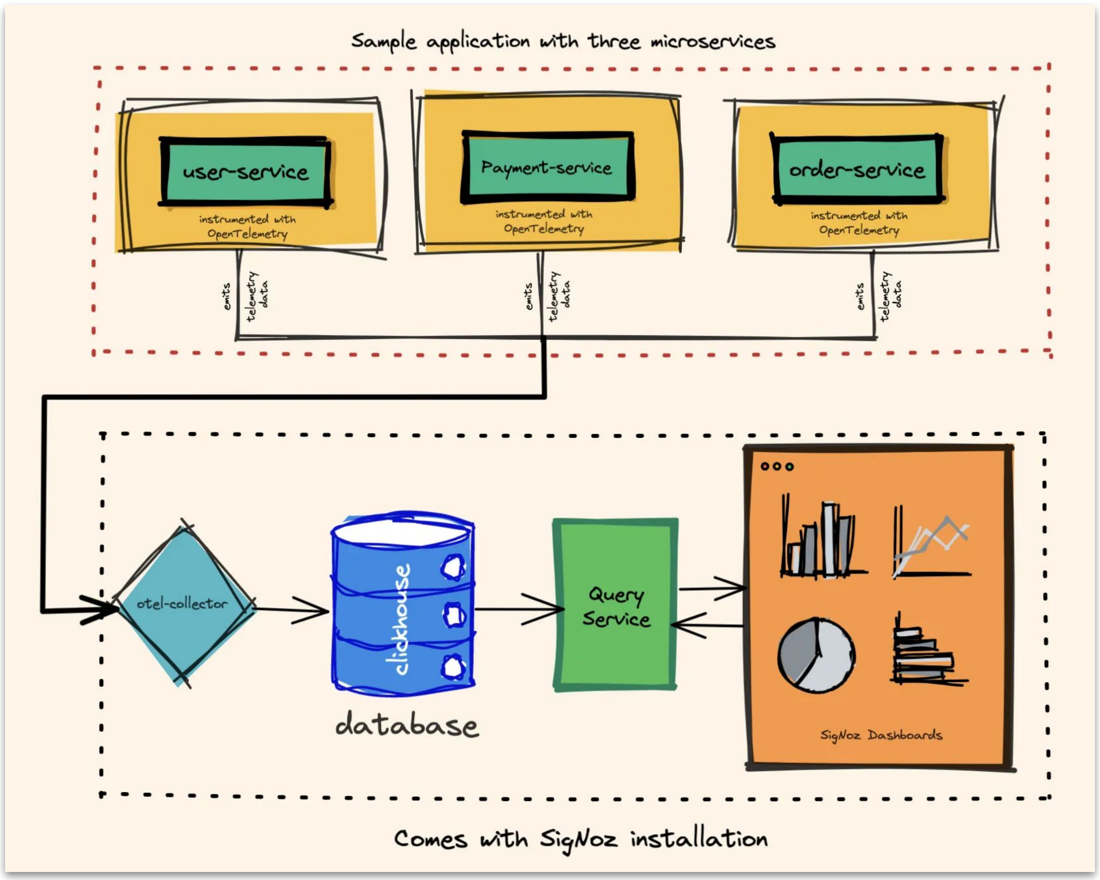

Sample application with three microservices
user-service
instrumented with OpenTelemetry
Payment-service
instrumented with OpenTelemetry
order-service
instrumented with OpenTelemetry
emits telemetry data
emits telemetry data
emits telemetry data
otel-collector
clickhouse
database
Query Service
SigNoz Dashboards
Comes with SigNoz installation

Image credit: https://signoz.io/blog/opentelemetry-backend/

Changes in cppTango and PyTango

- Opt-in, at 2 levels:
  - Compile-time (default is on)
  - Runtime via environment variable (default is off)
- Dependency on OpenTelemetry API and SDK, OpenSSL, gRPC, and more…
- Device servers and clients emit traces for most common operations. Standard Tango logs are also emitted. Note: only high-level API devices for PyTango.
- Some new environment variables (see docs)
- Not yet: option to toggle on/off after device server has process started

Performance impact*

		Telemetry OFF				*Telemetry ON*	
		(A) Compilation off		(B) Environment var off		(C) Environment var on	
Server	**Client**	**Time [µs]**	**Overhead [µs]**	**Time [µs]**	**Overhead [µs]**	**Time [µs]**	**Overhead [µs]**
C++	C++	10	0	11	1	26	16
C++	Python	20	0	21	1	115	95
Python	C++	43	0	42	-1	181	139
Python	Python	61	0	59	-2	313	252

- Time for client to read DevDouble attribute
- Single host running server, client, and database (i.e., loopback network)
- Averaged over 30k reads
- Traces and logs sent over gRPC
- Traces dropped if in-process buffer full

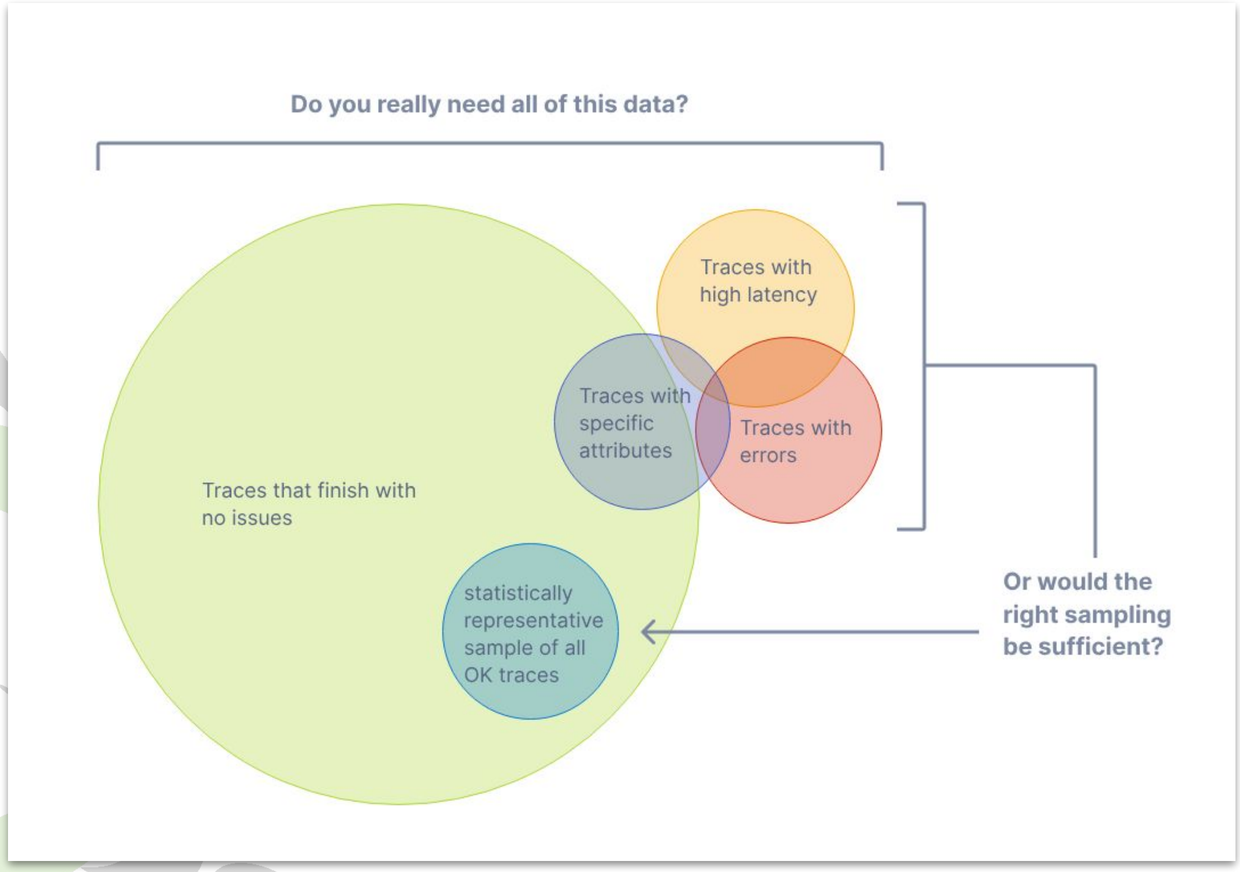
Standard deviation

(A)	3, 4, 13, 31 µs
(B)	4, 2, 11, 19 µs
(C)	11, 61, 85, 105 µs

* THIS WON'T MATCH YOUR SYSTEM!

Sampling

- Reduce the number of signals sent to the backend.
- Can be done by modifying the collector.
- E.g., keep 5% of spans


Do you really need all of this data?
Traces with high latency
Traces with specific attributes
Traces with errors
Traces that finish with no issues
statistically representative sample of all OK traces
Or would the right sampling be sufficient?

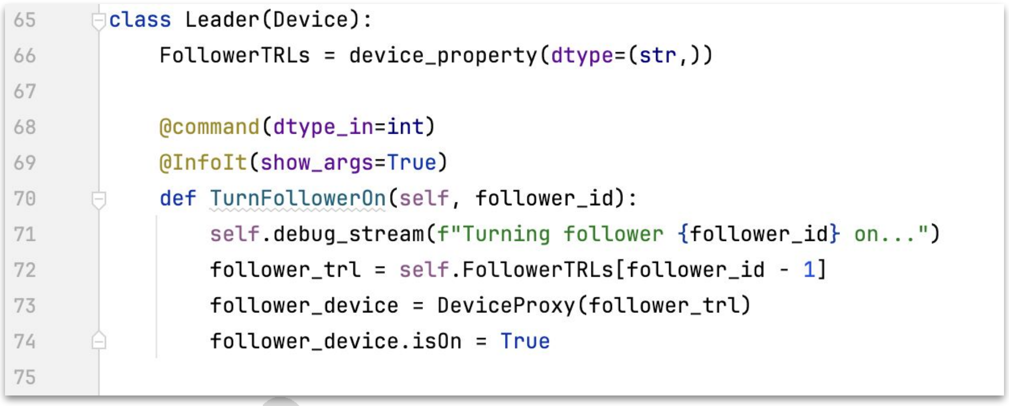
Source: https://opentelemetry.io/docs/concepts/sampling/

DEMO 2!

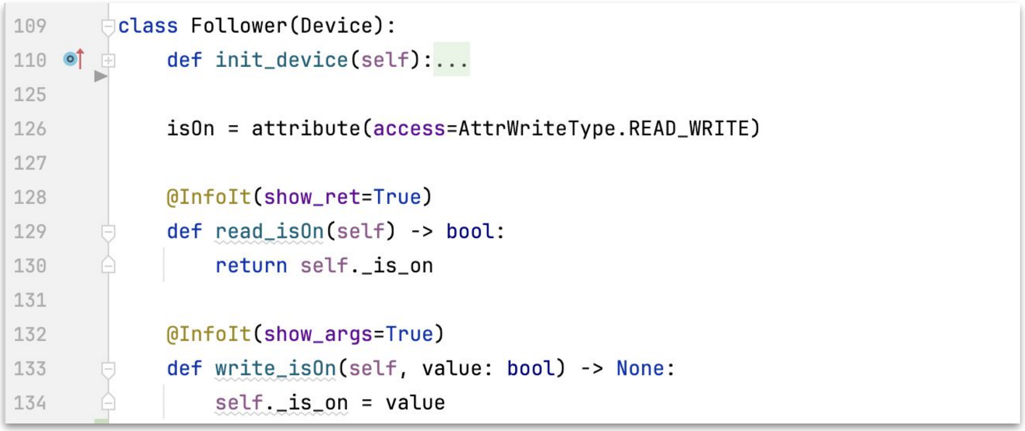
- Distributed tracing and logging

Image credit: Generated with AI

Telemetry demo devices

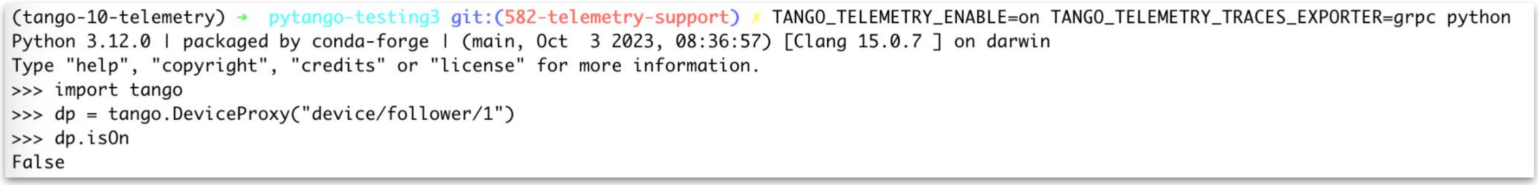


```python
class Leader(Device):
    FollowerTRLs = device_property(dtype=(str,))

    @command(dtype_in=int)
    @InfoIt(show_args=True)
    def TurnFollowerOn(self, follower_id):
        self.debug_stream(f"Turning follower {follower_id} on...")
        follower_trl = self.FollowerTRLs[follower_id - 1]
        follower_device = DeviceProxy(follower_trl)
        follower_device.isOn = True

```

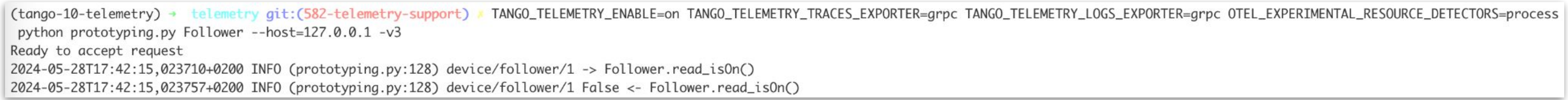


```python
class Follower(Device):
    def init_device(self):...

    isOn = attribute(access=AttrWriteType.READ_WRITE)

    @InfoIt(show_ret=True)
    def read_isOn(self) -> bool:
        return self._is_on

    @InfoIt(show_args=True)
    def write_isOn(self, value: bool) -> None:
        self._is_on = value
```

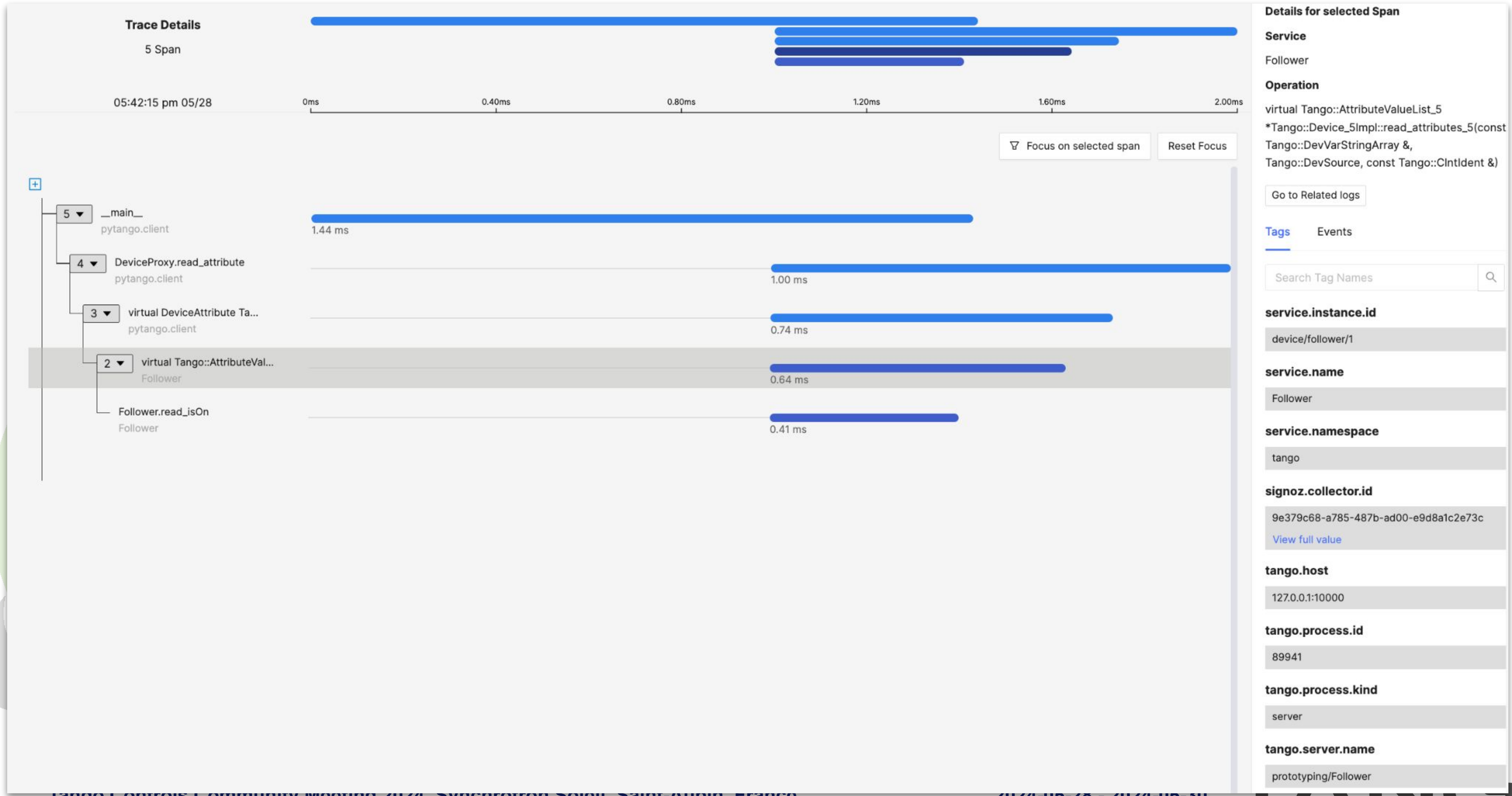
Telemetry demo - read attribute on Follower device



```
(tango-10-telemetry) ➜  pytango-testing3 git:(582-telemetry-support) ✗ TANGO_TELEMETRY_ENABLE=on TANGO_TELEMETRY_TRACES_EXPORTER=grpc python
Python 3.12.0 | packaged by conda-forge | (main, Oct  3 2023, 08:36:57) [Clang 15.0.7 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tango
>>> dp = tango.DeviceProxy("device/follower/1")
>>> dp.isOn
False
```



```
(tango-10-telemetry) ➜  telemetry git:(582-telemetry-support) ✗ TANGO_TELEMETRY_ENABLE=on TANGO_TELEMETRY_TRACES_EXPORTER=grpc TANGO_TELEMETRY_LOGS_EXPORTER=grpc OTEL_EXPERIMENTAL_RESOURCE_DETECTORS=process
 python prototyping.py Follower --host=127.0.0.1 -v3
Ready to accept request
2024-05-28T17:42:15,023710+0200 INFO (prototyping.py:128) device/follower/1 -> Follower.read_isOn()
2024-05-28T17:42:15,023757+0200 INFO (prototyping.py:128) device/follower/1 False <- Follower.read_isOn()
```



Trace Details

5 Span

05:42:15 pm 05/28

5 __main__ pytango.client 1.44 ms

4 DeviceProxy.read_attribute pytango.client 1.00 ms

3 virtual DeviceAttribute Ta... pytango.client 0.74 ms

2 virtual Tango::AttributeVal... Follower 0.64 ms

Follower.read_isOn Follower 0.41 ms

Details for selected Span

Service

Follower

Operation

virtual Tango::AttributeValueList_5 *Tango::Device_5Impl::read_attributes_5(const Tango::DevVarStringArray &, Tango::DevSource, const Tango::ClntIdent &)

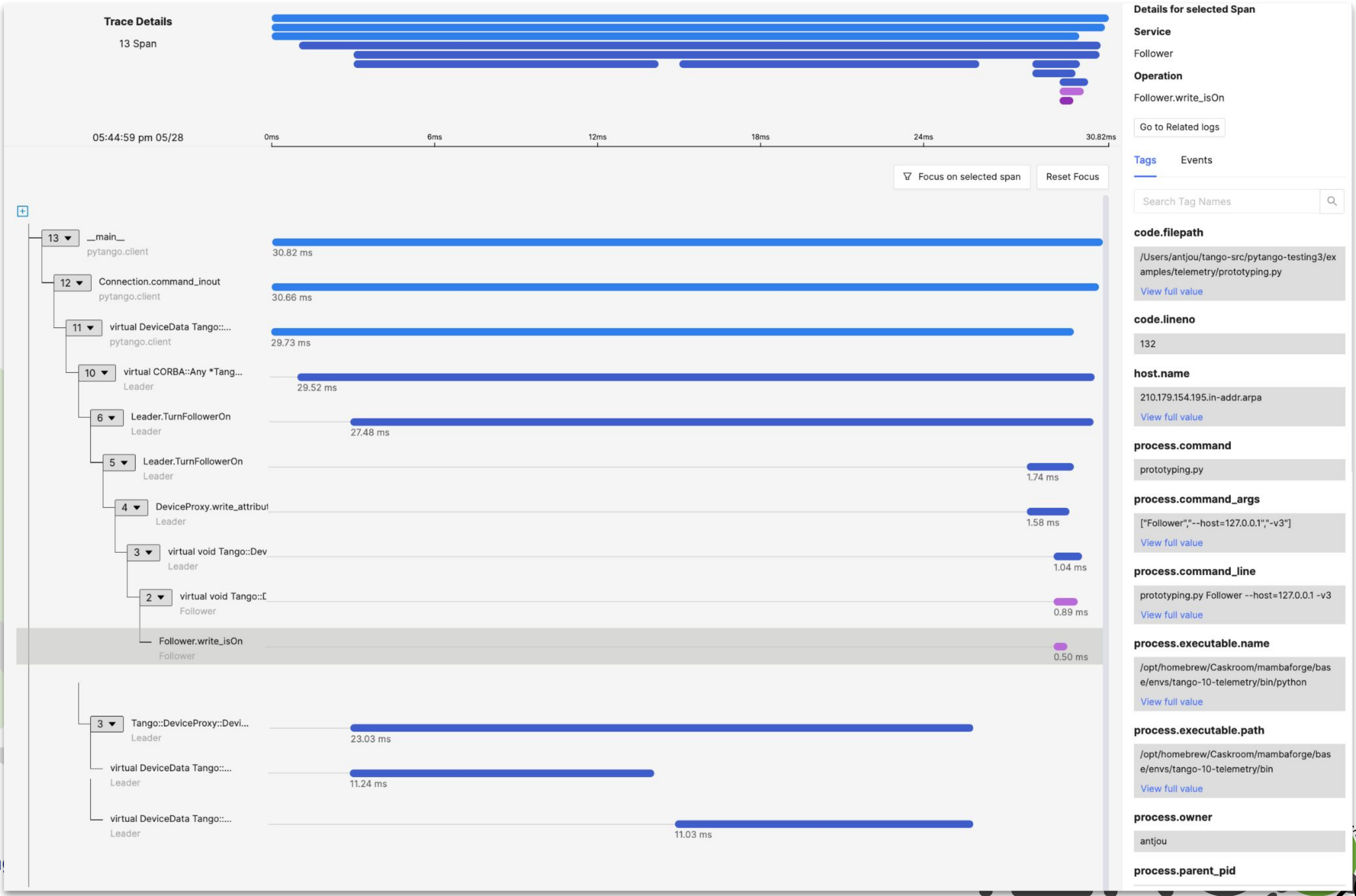
service.instance.id	device/follower/1
service.name	Follower
service.namespace	tango
signoz.collector.id	9e379c68-a785-487b-ad00-e9d8a1c2e73c View full value
tango.host	127.0.0.1:10000
tango.process.id	89941
tango.process.kind	server
tango.server.name	prototyping/Follower

Telemetry demo - Leader turning Follower on

```
>>> dp = tango.DeviceProxy("device/leader/1")
>>> dp.TurnFollowerOn(1)
```

```
(tango-10-telemetry) ➜  telemetry git:(582-telemetry-support) ✗ TANGO_TELEMETRY_ENABLE=on TANGO_TELEMETRY_TRACES_EXPORTER=grpc TANGO_TELEMETRY_LOGS_EXPORTER=grpc python prototyping.py Leader --host=127.0.0.1 -v3

Ready to accept request
2024-05-28T17:44:59,584418+0200 INFO (prototyping.py:68) device/leader/1 -> Leader.TurnFollowerOn(1)
2024-05-28T17:44:59,584519+0200 DEBUG (prototyping.py:71) device/leader/1 Turning follower 1 on...
2024-05-28T17:44:59,611571+0200 INFO (prototyping.py:68) device/leader/1 <- Leader.TurnFollowerOn()
```

```
2024-05-28T17:42:15,023710+0200 INFO (prototyping.py:128) device/follower/1 -> Follower.read_isOn()
2024-05-28T17:42:15,023757+0200 INFO (prototyping.py:128) device/follower/1 False <- Follower.read_isOn()
```



```
13 __main__                      pytango.client  30.82 ms
  12 Connection.command_inout    pytango.client  30.66 ms
    11 virtual DeviceData Tango::...  pytango.client  29.73 ms
      10 virtual CORBA::Any *Tang...  Leader  29.52 ms
        6 Leader.TurnFollowerOn       Leader  27.48 ms
          5 Leader.TurnFollowerOn     Leader  1.74 ms
            4 DeviceProxy.write_attribut  Leader  1.58 ms
              3 virtual void Tango::Dev   Leader  1.04 ms
                2 virtual void Tango::D   Follower  0.89 ms
                  Follower.write_isOn     Follower  0.50 ms
        3 Tango::DeviceProxy::Devi...  Leader  23.03 ms
          virtual DeviceData Tango::...  Leader  11.24 ms
          virtual DeviceData Tango::...  Leader  11.03 ms
```

Details for selected Span

Service
Follower

Operation
Follower.write_isOn

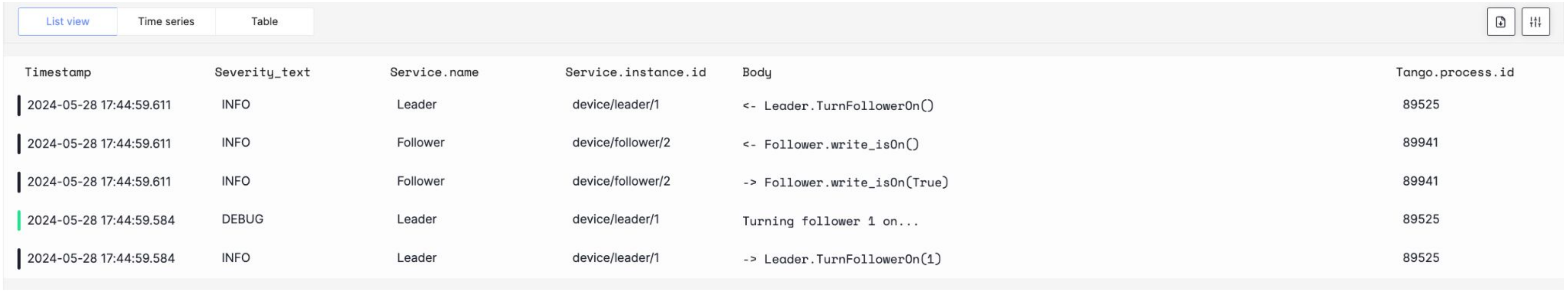
Go to Related logs

Tags Events

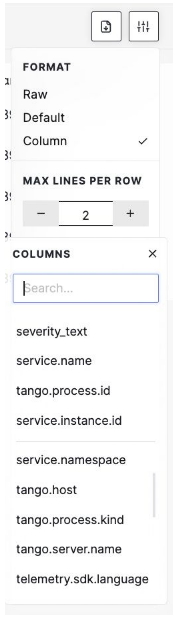
Search Tag Names

code.filepath	/Users/antjou/tango-src/pytango-testing3/examples/telemetry/prototyping.py
code.lineno	132
host.name	210.179.154.195.in-addr.arpa
process.command	prototyping.py
process.command_args	["Follower","--host=127.0.0.1","-v3"]
process.command_line	prototyping.py Follower --host=127.0.0.1 -v3
process.executable.name	/opt/homebrew/Caskroom/mambaforge/base/envs/tango-10-telemetry/bin/python
process.executable.path	/opt/homebrew/Caskroom/mambaforge/base/envs/tango-10-telemetry/bin
process.owner	antjou
process.parent_pid	

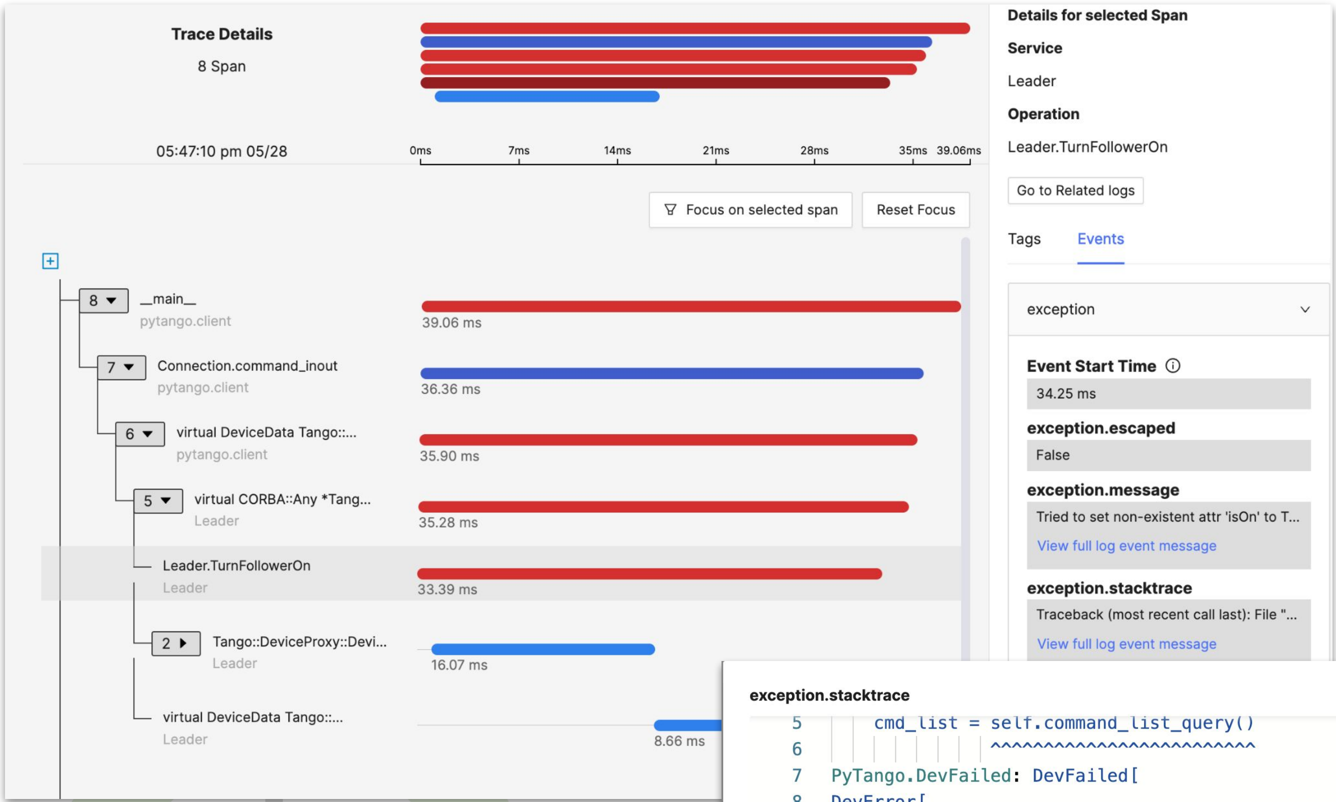
Telemetry demo - logs associated with trace



Timestamp	Severity_text	Service.name	Service.instance.id	Body	Tango.process.id
2024-05-28 17:44:59.611	INFO	Leader	device/leader/1	`<- Leader.TurnFollowerOn()`	89525
2024-05-28 17:44:59.611	INFO	Follower	device/follower/2	`<- Follower.write_isOn()`	89941
2024-05-28 17:44:59.611	INFO	Follower	device/follower/2	`-> Follower.write_isOn(True)`	89941
2024-05-28 17:44:59.584	DEBUG	Leader	device/leader/1	`Turning follower 1 on...`	89525
2024-05-28 17:44:59.584	INFO	Leader	device/leader/1	`-> Leader.TurnFollowerOn(1)`	89525



Telemetry demo - exception (Follower device server stopped)



Span	Service	Duration
8 __main__	pytango.client	39.06 ms
7 Connection.command_inout	pytango.client	36.36 ms
6 virtual DeviceData Tango::...	pytango.client	35.90 ms
5 virtual CORBA::Any *Tang...	Leader	35.28 ms
Leader.TurnFollowerOn	Leader	33.39 ms
2 Tango::DeviceProxy::Devi...	Leader	16.07 ms
virtual DeviceData Tango::...	Leader	8.66 ms

Service: Leader

Operation: Leader.TurnFollowerOn

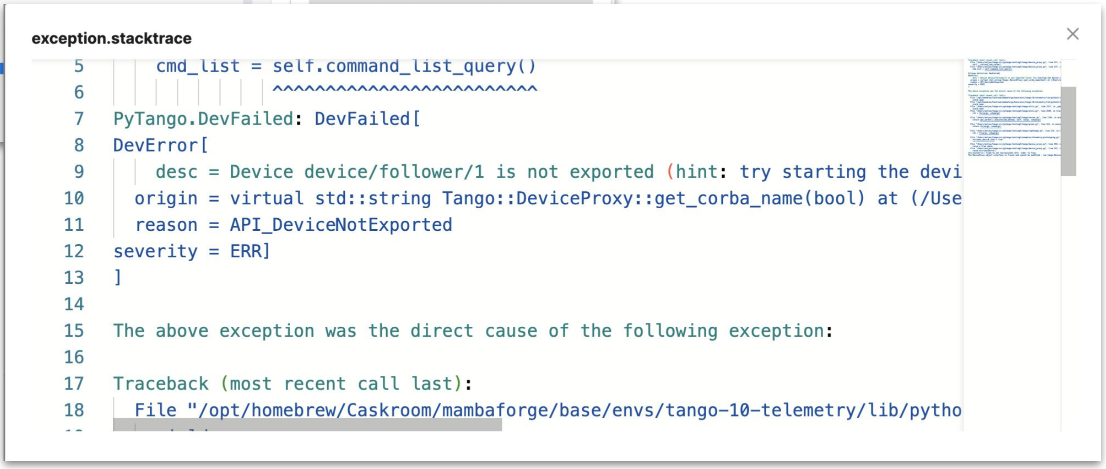
exception

Event Start Time: 34.25 ms

exception.escaped: False

exception.message: Tried to set non-existent attr 'isOn' to T...
View full log event message

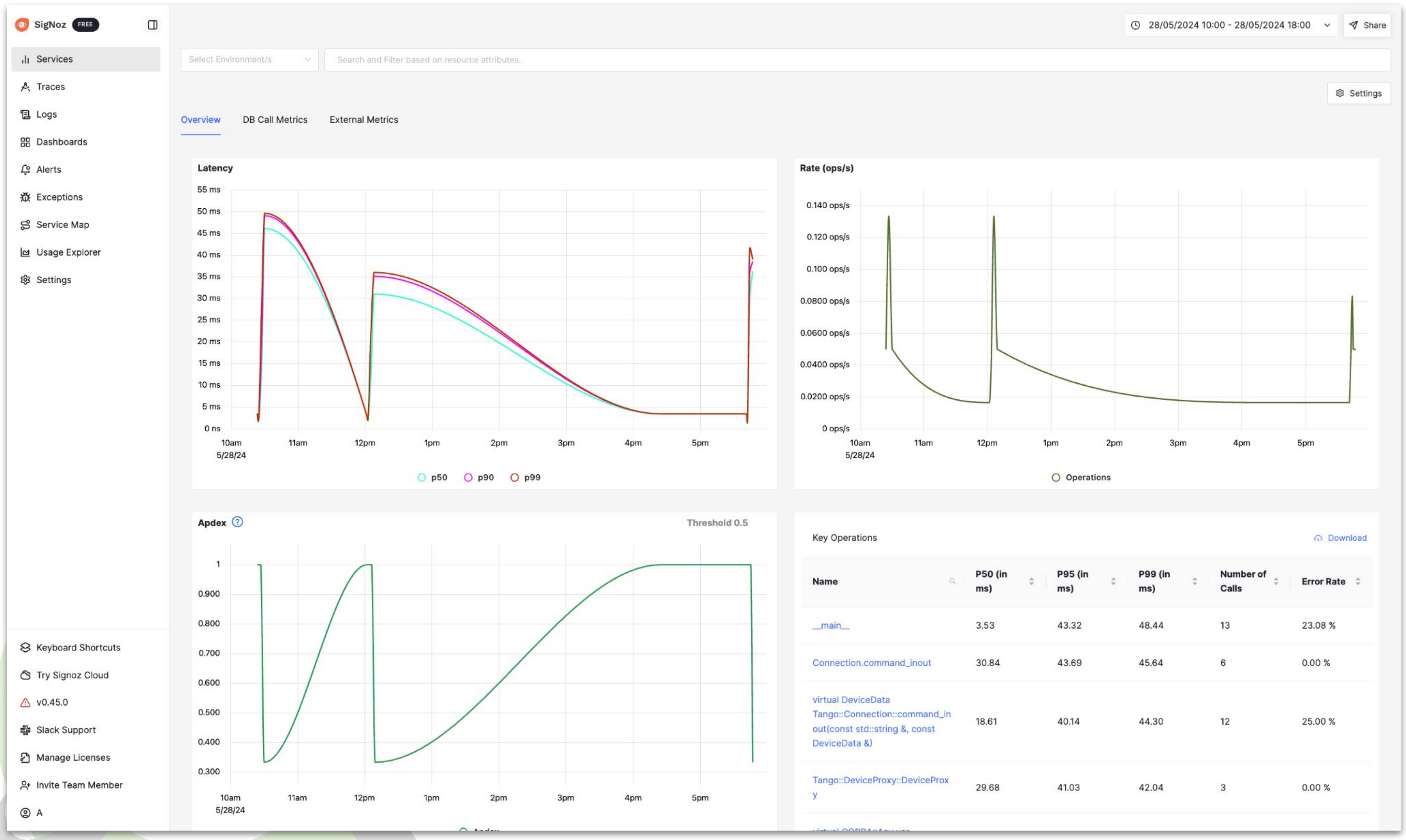
exception.stacktrace: Traceback (most recent call last): File "...
View full log event message



exception.stacktrace

```
    cmd_list = self.command_list_query()
                         ^^^^^^^^^^^^^^^^^^^^^^^^^
PyTango.DevFailed: DevFailed[
DevError[
    desc = Device device/follower/1 is not exported (hint: try starting the devi
  origin = virtual std::string Tango::DeviceProxy::get_corba_name(bool) at (/Use
  reason = API_DeviceNotExported
severity = ERR]
]

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/opt/homebrew/Caskroom/mambaforge/base/envs/tango-10-telemetry/lib/pytho
```

Telemetry - Example of stats for a service



Name	P50 (in ms)	P95 (in ms)	P99 (in ms)	Number of Calls	Error Rate
__main__	3.53	43.32	48.44	13	23.08 %
Connection.command_inout	30.84	43.69	45.64	6	0.00 %
virtual DeviceData Tango::Connection::command_inout(const std::string &, const DeviceData &)	18.61	40.14	44.30	12	25.00 %
Tango::DeviceProxy::DeviceProxy	29.68	41.03	42.04	3	0.00 %

And now…

Acknowledgement (in alphabetic order):

Anton Joubert
Chien Li
Gaetan Schneller
Graziano Scalamera
Grzergorz Kowalski
Guifre Cuni
Guillaume Pichon
Gwenaelle Abeille
Johan Forsberg
Jose Ramos
Lorenzo Pivetta
Maria Teresa Nuñez
Mateus Celary
Nicolas Leclercq
Piotr Goryl
Reynald Bourtembourg
Thomas Braun
Thomas Ives
Thomas Madej
Thomas Juerges
Sergi Rubio
Stephane Poirier
Vincent Hardion
Yury Matveyev
Zbigniew Reszela

Thank you!
Any questions?

Bonus slides

Tango Controls IDLv7: New datatype DevDict

- What is it? A complex datatype that resembles a Python dict
- Why do we need it?
  - Many hacks and workarounds can be avoided
  - The new multi-parameter command will be much easier to implement
- What is the catch?
  - You will fall in love with it 😍
- When? IDLv7, 11.0.0 release

Tango Controls IDLv7: Warning and alarm hysteresis

- What is it? Support for hysteresis in WARN and ALARM quality factor decision

- Why do we need it?
  - Avoid too frequent jittering of WARN or ALARM quality factors
  - Remove the need for workarounds

- What is the catch?
  - No catch

- When? IDLv7, 11.0.0 release

Tango Controls IDLv7: Deprecating of DevPipes

- Once DevDict is available, we would like to deprecate the DevPipe data type.

- Removal would be in the next major version after that, so 12.0.0.

Tango Controls IDLv8: Multi-parameter commands

- What is it? Commands with more than one parameter

- Why do we need it? Because a single parameter is often too limiting

- What is the catch?
  - Needs DevDict
  - Therefore not all data types will be supported

- When? IDLv8, 12.0.0 release

Tango Controls IDLv9: Multi-dimensional arrays

- What is it? Arrays with more dimensions than 1 or 2
- Why do we need it? Because DevSpectrum and DevImage are not enough any more to match our use-cases
- What is the catch? Not all basic data types will be supported
- When? IDLv9, 13.0.0 release
- What will happen to DevSpectrum and DevImage?
  - They will continue to exist
  - They will be implemented as 1d- or 2d-case of the multi-dimensional array