PyTango 8

…again?

Agenda

- Latest release 8.0.3
  - Bug fixes
  - Feature requests
  - Improvements
- Proposals
  - Proposal for proposals: TEP
  - HLAPI proposal
  - DatabaseDS proposal
  - numpy lazy import proposal
  - Multiple binding proposal

8.0.3

- Bug fixes (6 SourceForge + ~10 undescribed)
  - Attribute missing methods
  - DeviceClass methods not python GIL safe
  - DeviceProxy.__setattr__ breaks python descriptors
  - itango on 64bits exception
  - Import DLL failed on windows
  - PyTango build fails checking strange versions ex:1.2.3b1
- Feature requests (1 SourceForge)
  - `Util.server_set_event_loop()`
- Documentation updated with PyTEP
- PyTango on Windows:
  - (Python 2.6, 2.7, 3.2, 3.3) x (32, 64 bits)

PyTango 8.0.3 on Windows

Python & VC++		
	32bits	**64bits**
Py 2.6	9	9
Py 2.7	9	9
Py 3.2	9	9
Py 3.3	10	10

Tango 8 & VC++	
32bits	**64bits**
9	10

LEFT JOIN

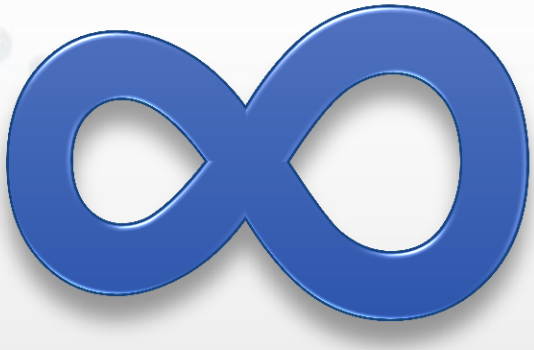
PyTango & VC++		
	32bits	**64bits**
Py 2.6	9	**9**
Py 2.7	9	**9**
Py 3.2	9	**9**
Py 3.3	**10**	10

python + tango + boost + VC++ =!!

- recipe:
  - pytango VC version = python VC version
  - home made boost-python lib
  - link statically with boost-python lib
  - link statically with tango lib

Proposal for proposals

- Today: New proposals in tango:
  - Mail to the mailing list
  - Wiki tango page
  - Ask for a new feature in SF



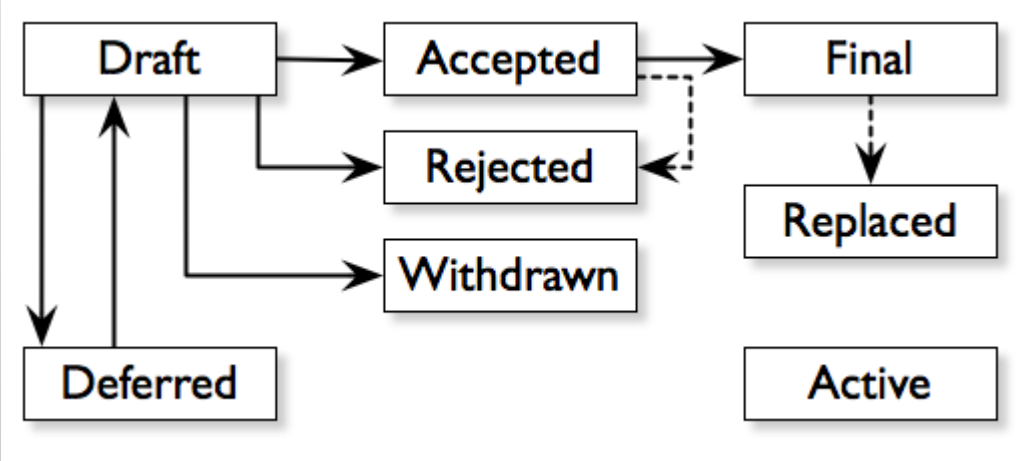
- Proposal:
  - Use a TEP: Tango Enhancement Proposal system
  - Like DEB: Debian
  - Like PEP: Python

TEP

- First step:

  TEP 0 – TEP Purpose and Guidelines

- What is a TEP?
- TEP workflow
- TEP Formats and templates
- Reporting TEP bugs and submitting updates
- …



TEP 1 - HLAPI

High Level tango device server API

*a **new** high level API for writing device servers*

- Maintain all features of the low-level API
- Automatic inheritance from the latest `DeviceImpl`
- No need to program `DeviceClass`
- Default implementation of `DeviceImpl` constructor
- Default implementation of `DeviceImpl.init_device()`
- Pythonic read/write attribute
- Pythonic command
- Pythonic property
- Simplify main()

www.tango-controls.org/static/PyTango/development/doc/html/tep/tep-0001.html

TEP 1 - HLAPI

```
import PyTango
import sys

class Motor (PyTango.Device_4Impl):

    def __init__(self,cl, name):
        PyTango.Device_4Impl.__init__(self,cl,name)
        self.debug_stream("In " + self.get_name() +
".__init__()")
        Motor.init_device(self)

    def delete_device(self):
        self.debug_stream("In " + self.get_name() +
".delete_device()")

    def init_device(self):
        self.debug_stream("In " + self.get_name() +
".init_device()")

self.get_device_properties(self.get_device_class())
        self.attr_Position_read = 0.0

    def always_executed_hook(self):
        self.debug_stream("In " + self.get_name() +
".always_excuted_hook()")

    def read_Position(self, attr):
        self.debug_stream("In " + self.get_name() +
".read_Position()")
        self.attr_Position_read = 1.0
        attr.set_value(self.attr_Position_read)

    def read_attr_hardware(self, data):
        self.debug_stream("In " + self.get_name() +
".read_attr_hardware()")
```

Page 1

```
class MotorClass(PyTango.DeviceClass):

    attr_list = {
        'Position':
            [[PyTango.DevDouble,
            PyTango.SCALAR,
            PyTango.READ]],
        }

    def __init__(self, name):
        PyTango.DeviceClass.__init__(self, name)
        self.set_type(name);


def main():
    try:
        py = PyTango.Util(sys.argv)
        py.add_class(MotorClass,Motor,'Motor')

        U = PyTango.Util.instance()
        U.server_init()
        U.server_run()

    except PyTango.DevFailed,e:
        print '-------> Received a DevFailed exception:',e
    except Exception,e:
        print '-------> An unforeseen exception occured....',e

if __name__ == '__main__':
    main()
```

Page 2

TEP 1 - HLAPI

use `hlpai`

```
from PyTango import DebugIt, server_run
from PyTango.hlapi import Device DeviceMeta, attr


class Motor(Device):
    __metaclass__ = DeviceMeta

    position = attr()

    @DebugIt()
    def read_position(self):
        return 1.0

def main():
    server_run((Motor,))

if __name__ == "__main__":
    main()
```

`Device` replaces `Device_4Impl`

`tg_attr` *descriptor* defines tango attributes

pythonic read attribute

simplified main

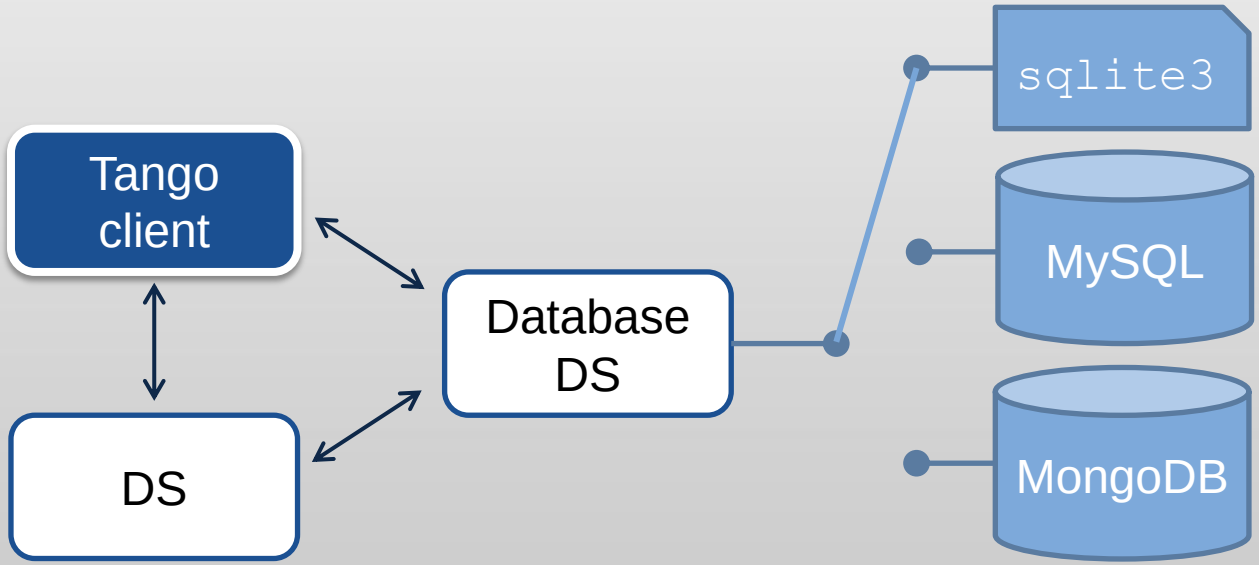
TEP 1 - HLAPI

```
class Motor(Device):
    __metaclass__ = DeviceMeta

    port = cls_prop(dtype=int, default=8765)
    step_per_unit = prop(dtype=float, default=1.0)

    velocity = attr(dtype=float,
        fread="get_speed", label="Speed",
        description="motor maximum speed",
        min_value=0, max_value=50)

    def get_speed(self):
        return 1.0

    @cmd(float, None):
    def move(self, new_pos):
        pass
```

TEP 2 - Tango DB serverless

Allow anyone to try/develop tango without the need to have a running SQL server

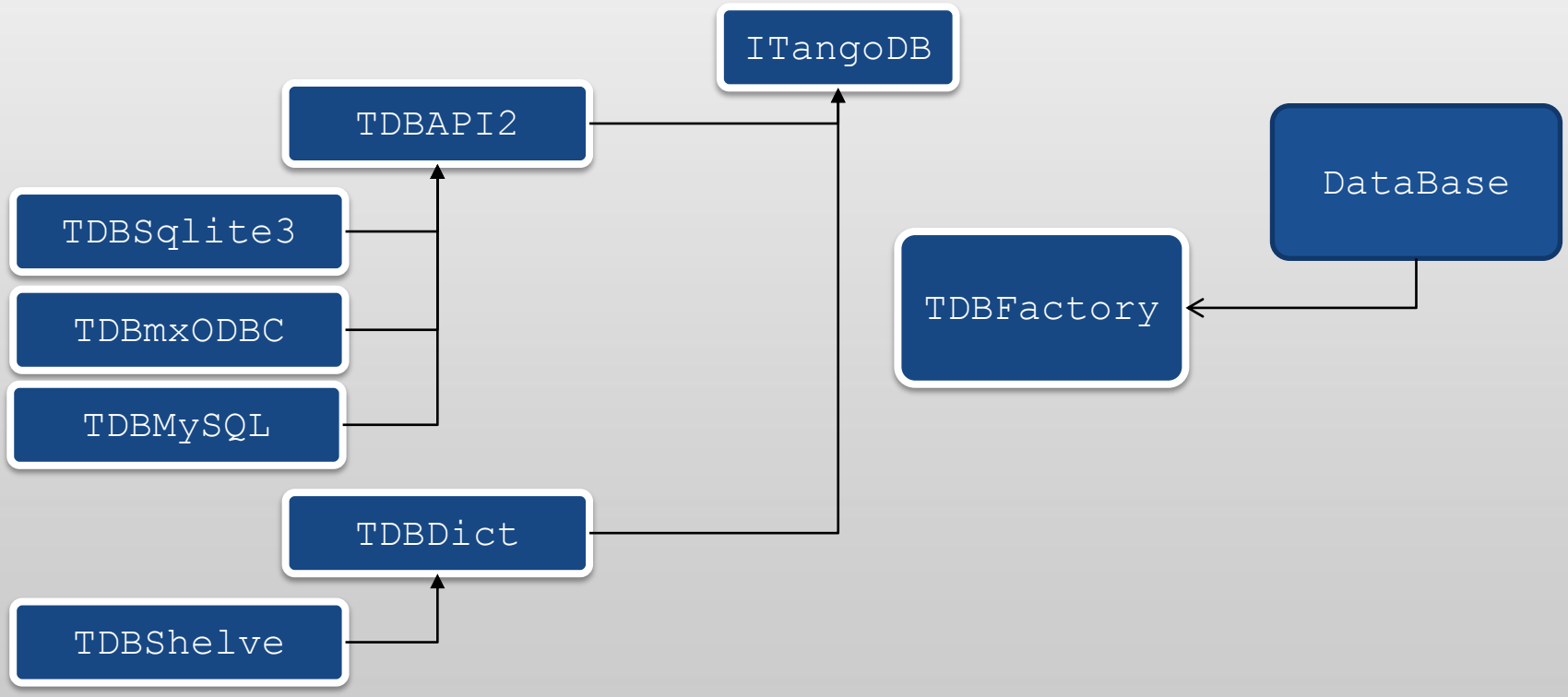
- A PyTango based DatabaseDS
- 100% Tango DatabaseDS C++ API compatible
- SQL server optional
- Any DB backend: dict, sqlite3 file, dbm, mysql, mongodb, …



www.tango-controls.org/static/PyTango/development/doc/html/tep/tep-0002.html

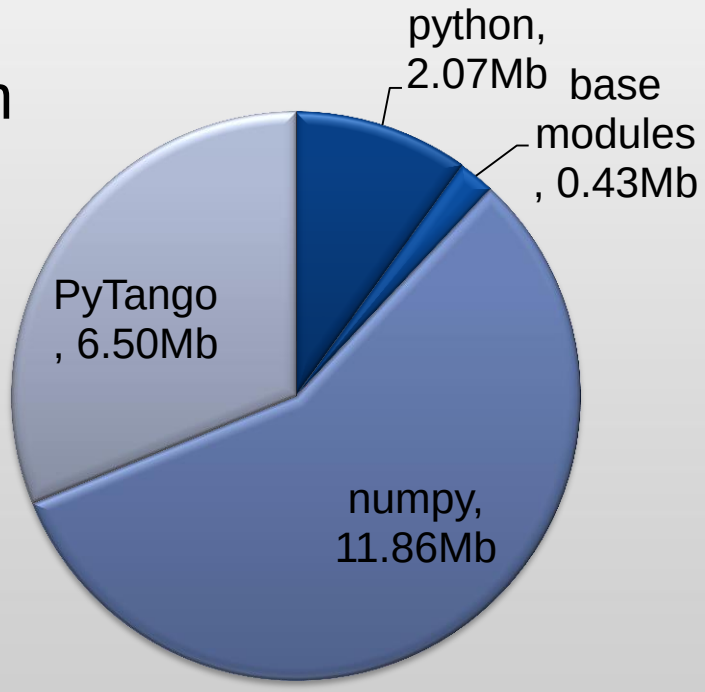
TEP 2 - Tango DB serverless

- Define a TangoDB interface: `ITangoDB`
  - `TangoDBSqlite3`, `TangoMySQLdb`, ...
- `TangoDBFactory` provides a proper specific `ITangoDB` on request
- Tango DatabaseDS asks `TangoDBFactory` for a specific `ITangoDB`
- Tango DatabaseDS talks to the given `ITangoDB`



numpy lazy import

- numpy is responsible for a big part of PyTango memory footprint
- Many DS don’t actually need numpy (SCALAR attributes and commands)
- Remove windows binary minimum dependency on numpy version
- Many DS running on RAM limited machines (ex.: IPC)



Python 2.6 32bits Windows 8 64bits

TEP [4]

Multiple binding proposal

Had capability do use different binding tools to make PyTango

- Currently boost-python:
  - Big stack trace in C++
  - Big memory footprint
  - Difficult to overwrite boost behavior for strings (UTF7 vs UTF8)
  - DLL hell on windows
  - Not prepared for python 3
  - Not prepared for C++ 0X
  - Heavy compilation
- Prepare SVN repository [DONE]
- Started proof of concept SIP binding => promising

Questions