

PyTango



Tiago Coutinho

■ What's new in 7.1.2

- General

- Client

- Server

■ Future changes

- Roadmap

- 7.1.3 highlights

- ❏ 7 bugs from SF
- ❏ 7 feature requests from SF
- ❏ Better online documentation

<http://www.tango-controls.org/static/PyTango/latest/doc/html/revision.html#version-history>

Attributes as members are back!

- Only if DS is running
- Auto-complete works!

```
>>> import PyTango
>>> power_supply = PyTango.DeviceProxy("B001/PC/CORH-01")
>>> print power_supply.read_attribute("voltage").value
12.354
>>> print power_supply.v<tab>
power_supply.version power_supply.voltage
>>> print power_supply.voltage
12.354
```

BTW: it also works for commands

```
>>> print power_supply.Cab<tab>
power_supply.CabinetOn power_supply.CabinetOff
>>> power_supply.CabinetOn()
>>> power_supply.command_inout("CabinetOn")
```

❏ Commands return numpy arrays

❏ Before (< 7.1.2)

```
>>> wave = power_supply.WaveForm()
>>> print wave
[1.32, 5.43, 6.4343, 124.43556, 5.343, ... 546.545]
>>> print type(wave)
<type 'list'>
```

❏ Now

```
>>> wave = power_supply.WaveForm()
>>> print wave
array([ 1.32,  5.43,  6.4343, ..., 453.4,
        454.43, 546.545])
>>> print type(wave)
<type 'numpy.ndarray'>
```

- ❏ Client objects are now 'Pickable'
 - ❏ Database
 - ❏ DeviceProxy
 - ❏ AttributeProxy
 - ❏ Group (not yet!)

```
>>> print power_supply
BruckerPS (B001/PC/CORH-01)
```

```
>>> import pickle
>>> f = file("hello.dat", "w")
>>> pickle.dump(power_supply, f)
>>> f.close()
>>> new_power_supply = pickle.load(file("hello.dat"))
>>> print new_power_supply
BruckerPS (B001/PC/CORH-01)
```

Spock: a PyTango profile to IPython

Highlights

Tab completion ¶

Spock exports many tango specific objects to the console namespace. These include:

- the PyTango module itself

```
Spock <homer:10000> [1]: PyTango
Result [1]: <module 'PyTango' from ...>
```

- The **DeviceProxy** (=Device), **AttributeProxy** (=Attribute), **Database** and **Group** classes

```
Spock <homer:10000> [1]: De<tab>
DeprecationWarning      Device      DeviceProxy

Spock <homer:10000> [2]: Device
Result [2]: <class 'PyTango._PyTango.DeviceProxy'>

Spock <homer:10000> [3]: Device("sys/tg_test/1")
Result [3]: DeviceProxy(sys/tg_test/1)

Spock <homer:10000> [4]: Datab<tab>

Spock <homer:10000> [4]: Database

Spock <homer:10000> [4]: Att<tab>
Attribute      AttributeError  AttributeProxy
```

- The tango **PyTango.Database** object to which the spock session is currently connected

```
Spock <homer:10000> [1]: db
Result [1]: Database(homer, 10000)
```

Device name completion

Spock knows the complete list of device names (including alias) for the current tango database. This means that when you try to create a new Device, by pressing <tab> you can see a context sensitive list of devices.

```
Spock <homer:10000> [1]: test = Device("<tab>
Display all 3654 possibilities? (y or n) n

Spock <homer:10000> [1]: test = Device("sys<tab>
sys/access_control/1  sys/database/2  sys/tautest/1  sys/tg_test/1

Spock <homer:10000> [2]: test = Device("sys/tg_test/1")
```

```
tcoutinho@pt060:~$ ipython -p spock
Spock 7.1.3 -- An interactive Tango client.
Running on top of Python 2.6.5, IPython 0.10 and PyTango 7.1.3dev
help      -> Spock's help system.
object?   -> Details about 'object'. ?object also works, ?? prints more.
hint: Try typing: mydev = Device("<tab>

Spock <pt060:10000> [1]: motor = Motor("BL98_M1")

Spock <pt060:10000> [2]: motor
Result [2]: Motor(motor/bl98_dummymotorctrl1/1)

Spock <pt060:10000> [3]: motor.position
Result [3]: 100.0

Spock <pt060:10000> [4]: switchdb controls01
Result [4]: Database(controls01, 10000)

Spock <controls01:10000> [5]:
```

<http://www.tango-controls.org/static/PyTango/latest/doc/html/spock/index.html>

Dynamic devices

Server

```
class MacroServer(PyTango.Device_4Impl):
    ...
    def CreateDoor(self, pars):
        dev_name, alias = pars
        util = PyTango.Util.instance()
        util.create_device("Door", dev_name, alias=alias, cb=None)

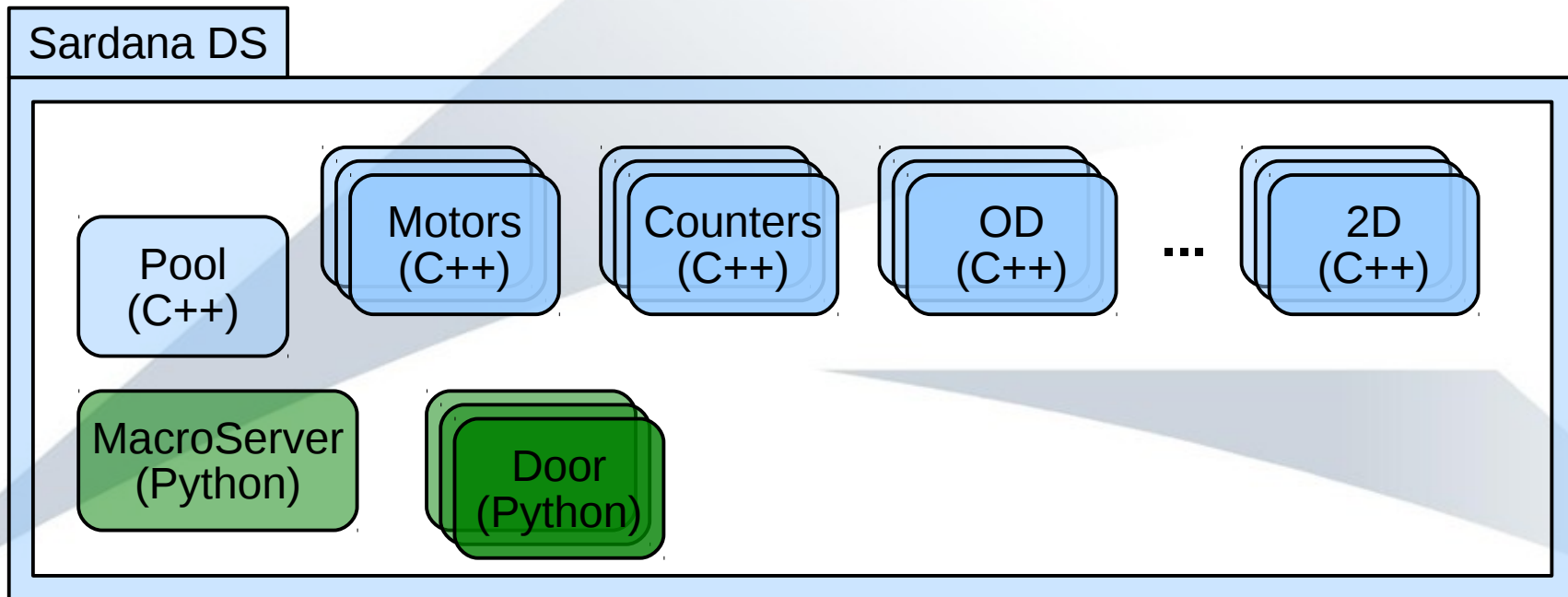
    def DeleteDoor(self, dev_name):
        util = PyTango.Util.instance()
        util.delete_device("Door", dev_name)
```

Client

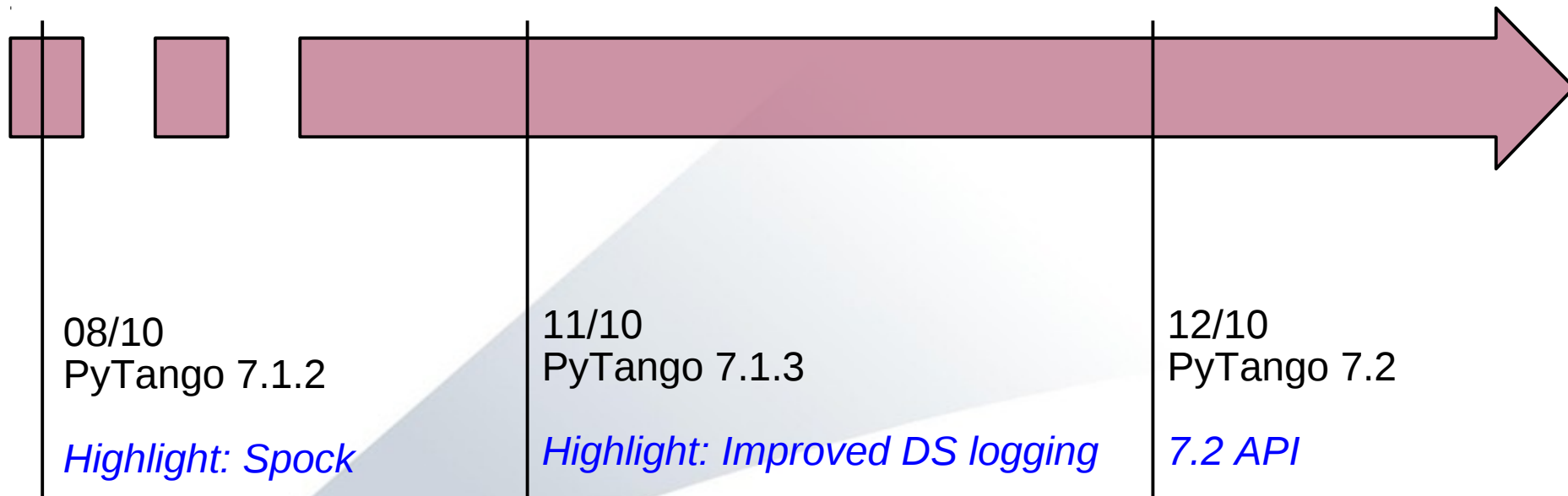
```
>>> macro_server = PyTango.DeviceProxy("BL99_MS")
>>> macro_server.CreateDoor("DOOR/BL99/1", "BL99_DOOR1")
>>> door = PyTango.DeviceProxy("BL99_DOOR1")
>>> door.RunMacro("ascan energy 20 600 20 0.1")
```

<http://www.tango-controls.org/static/PyTango/latest/doc/html/server/index.html#dynamic-devices>

- ❏ Mix C++ and Python in same DS
 - ❏ Feature exists since 3.0.3
 - ❏ Now is more robust
 - ❏ Tested with Sardana
 - Pool (C++) + MacroServer (Python)



Roadmap



- Fixed licence information to LGPL
- TLS (Tango Logging System) decorators & print

```
class MacroServer(PyTango.Device_4Impl):
```

```
...
```

```
@PyTango.DebugIt()
```

```
def read_MacroList(self, attr):
```

```
    self.info_stream("inside read_MacroList")
```

```
    attr.set_value(self._macro_list)
```

```
1287643409 [-1417680016] DEBUG MS/BL99/1 -> read_MacroList()
1287643409 [-1417680016] INFO MS/BL99/1 inside read_MacroList
1287643409 [-1417680016] DEBUG MS/BL99/1 <- read_MacroList()
```

```
@PyTango.DebugIt(show_args=True, show_ret=False)
```

```
def CreateDoor(self, pars):
```

```
    print >>self.log_info, "inside CreateDoor"
```

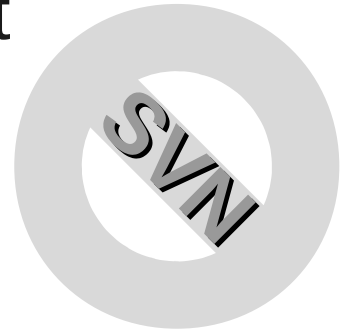
```
    dev_name, alias = pars
```

```
    util = PyTango.Util.instance()
```

```
    util.create_device("Door", dev_name)
```

```
1287643410 [-1417680016] DEBUG MS/BL99/1 -> CreateDoor(["DOOR/BL99/1", "BL99_DOOR1"])
1287643410 [-1417680016] INFO MS/BL99/1 inside CreateDoor
1287643410 [-1417680016] DEBUG MS/BL99/1 <- CreateDoor()
```

<http://www.tango-controls.org/static/PyTango/latest/doc/html/server/index.html#logging>



- ❏ Works with python 2.4
- ❏ Better spock
- ❏ Exported ApiUtil
- ❏ Bug read property bool[]



- ❑ DevEncoded JPEG support
- ❑ Automatic test procedure
- ❑ Python 3k – no excuse anymore!
 - ✓ Python
 - ✓ Boost-python
 - ✓ PyQt4
 - ✓ Numpy ≥ 1.5
- ❑ New pogo plugin
- ❑ Test SWIG/SIP/Shiboken against boost

