

Python Device server for SCPI instruments

S. Blanch-Torné¹ A. Milán² M. Broseta¹ C. Falcón¹
J. Andreu¹ D. Roldán¹ J. Moldes¹ G. Cuní¹

¹ALBA Synchrotron, CELLS
Cerdanyola del Vallès

²MAX IV Laboratory
Lund

Tango Meeting, 2019

- 1 What's SCPI?
- 2 Tango Device Servers
 - SkippyDS
 - Sardana Controller
- 3 Python module
 - python-skippy
 - python-scplib
- 4 Wish & ToDo lists

What's SCPI?

Standard Commands for Programmable Instruments



From the wikipedia's definition

The *Standard Commands for Programmable Instruments* (SCPI; often pronounced "skippy") defines a standard for syntax and commands to use in controlling programmable test and measurement devices, such as automatic test equipment and electronic test equipment.

What's SCPI?

Standard Commands for Programmable Instruments



From the wikipedia's definition

The *Standard Commands for Programmable Instruments* (SCPI; often pronounced "skippy") defines a standard for syntax and commands to use in controlling programmable test and measurement devices, such as automatic test equipment and electronic test equipment.

Standard definition

- SCPI-99
- IEEE 488.2-2004

What's SCPI?

Standard Commands for Programmable Instruments

From the wikipedia's definition

The *Standard Commands for Programmable Instruments* (SCPI; often pronounced "skippy") defines a standard for syntax and commands to use in controlling programmable test and measurement devices, such as automatic test equipment and electronic test equipment.

Standard definition

- SCPI-99
- IEEE 488.2-2004

How it looks like:

*IDN?,	SOURce:FREQuency:STARt?,
*RST,...	SYSTem:COMMunicate:SERial:BAUD 2400

Tango Device Servers



What we (all) did with SCPI, or at least what I've seen:

- At least 49 Device Servers identified in the Catalogue

Tango Device Servers

What we (all) did with SCPI, or at least what I've seen:

- At least 49 Device Servers identified in the Catalogue
- Represents $> 6\%$ of the current Device Servers in the inventory

Tango Device Servers

What we (all) did with SCPI, or at least what I've seen:

- At least 49 Device Servers identified in the Catalogue
- Represents $> 6\%$ of the current Device Servers in the inventory
- 40 are written in Cpp, 8 in Python, 1 in Java

What we (all) did with SCPI, or at least what I've seen:

- At least 49 Device Servers identified in the Catalogue
- Represents $> 6\%$ of the current Device Servers in the inventory
- 40 are written in Cpp, 8 in Python, 1 in Java

by Family

Communications	8
Instrumentation	19
Measurement Instruments	20
Other Instruments	1
Standard Interfaces	1

What we (all) did with SCPI, or at least what I've seen:

- At least 49 Device Servers identified in the Catalogue
- Represents $> 6\%$ of the current Device Servers in the inventory
- 40 are written in Cpp, 8 in Python, 1 in Java

by Family

Communications	8
Instrumentation	19
Measurement Instruments	20
Other Instruments	1
Standard Interfaces	1

by Institute

3control	1 ^a
alba	7
desy	22
esrf	8
nexeya	2
soleil	9

^aScpiDS multiple instruments

What we (all) did with SCPI, or at least what I've seen:

- At least 49 Device Servers identified in the Catalogue
- Represents $> 6\%$ of the current Device Servers in the inventory
- 40 are written in Cpp, 8 in Python, 1 in Java

by Family

Communications	8
Instrumentation	19
Measurement Instruments	20
Other Instruments	1
Standard Interfaces	1

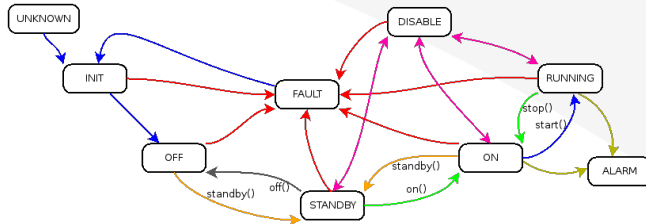
by Institute

3control	1 ^a
alba	7
desy	22
esrf	8
nexeya	2
soleil	9

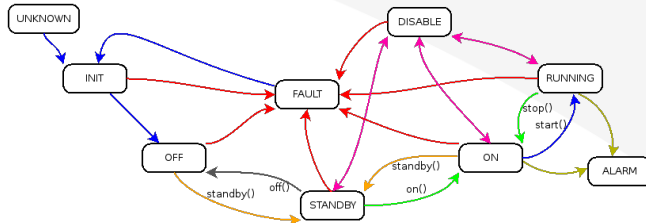
Skippy:
13 instruments
9 manufacturers
4 in progress

^aScpiDS multiple instruments

State machine and tango description



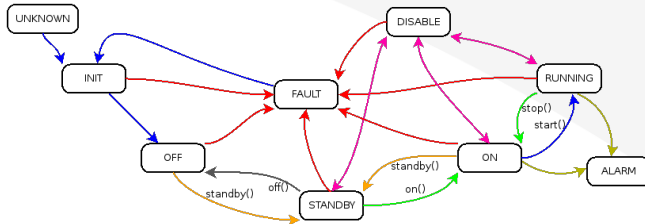
State machine and tango description



commands

- `IDN()`
- `Off()`, `Standby()`, `On()`
- `Start()`, `Stop()`
- `{Add,Remove}Monitoring()`
- `{Get,Set}MonitoringPeriod()`
- `CMD()`

State machine and tango description

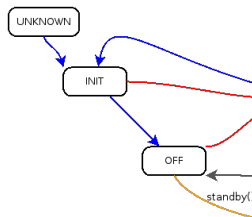


commands

- IDN()
- Off(), Standby(), On()
- Start(), Stop()
- {Add,Remove}Monitoring()
- {Get,Set}MonitoringPeriod()
- CMD()

attributes

- QueryWindow
- TimeStampsThreshold



Properties

- Instrument
- Port
- Serial{Baudrate,Bytesize,...}
- Num{Channels,Functions,Multiple}
- MonitoredAttributes
- Auto{Standby,On,Start}
- TxTerminator

commands

- IDN()
- Off(), Standby(), On()
- Start(), Stop()
- {Add,Remove}Monitoring()
- {Get,Set}MonitoringPeriod()
- CMD()

attributes

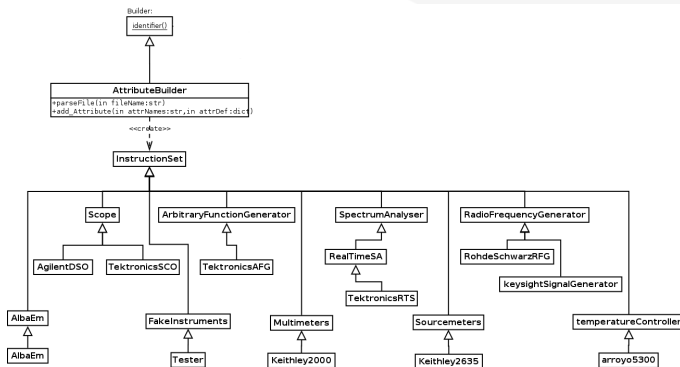
- QueryWindow
- TimeStampsThreshold

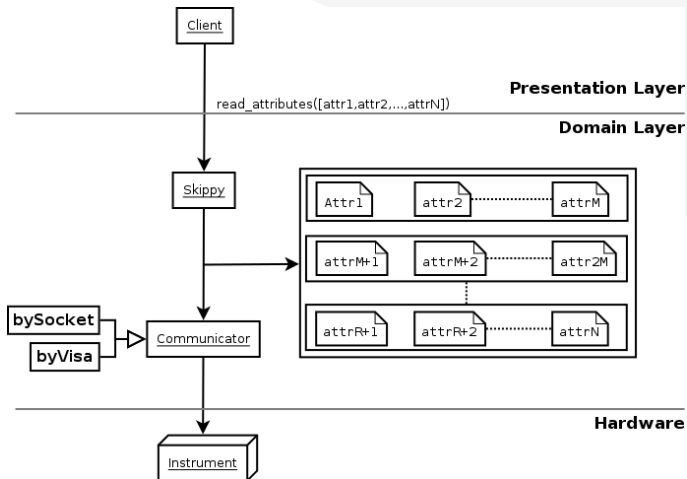
Attribute builder

```
Attribute('IO',
        {'type': PyTango.CmdArgType.DevString,
         'dim': [0],
         'readCmd': lambda mult, num: "%s%.2d:VALU?" % (mult, num),
         'writeCmd': lambda mult, num: (
             lambda value: "%s%.2d:VALU_%s" % (mult, num, value)),
         'multiple': {'scpiPrefix': 'IOPort', 'attrSuffix': 'Port'}}
    )
```

keywords

- type, dim
- label, description,
- format, unit,
- memorized
- min/max
- readCmd, writeCmd
- channels, functions, multiple
- delayAfterWrite
- readFormula





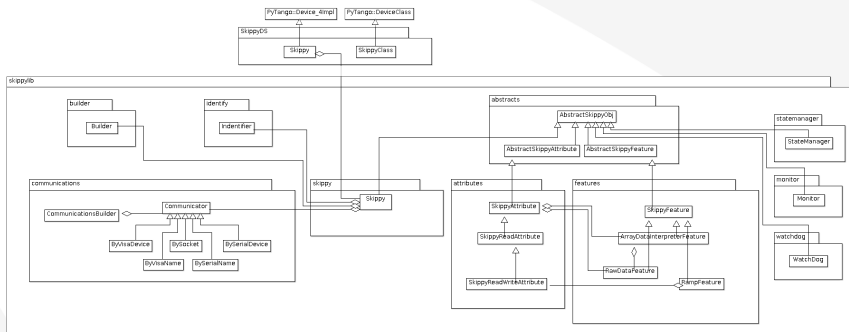
- 1 Use a proxy in the controller

- 1 Use a proxy in the controller
- 2 Reduce layers: Instead of use a tango device, implement a native access to the instruments in the controller

- ① Use a proxy in the controller
 - ② Reduce layers: Instead of use a tango device, implement a native access to the instruments in the controller
- **Again**, one specific controller per instrument?

- ① Use a proxy in the controller
 - ② Reduce layers: Instead of use a tango device, implement a native access to the instruments in the controller
- **Again**, one specific controller per instrument?
 - **Reimplement** generic features?

- ① Use a proxy in the controller
 - ② Reduce layers: Instead of use a tango device, implement a native access to the instruments in the controller
- **Again**, one specific controller per instrument?
 - **Reimplement** generic features?
 - **Encapsulate and share** the features: a python module



Software patterns

Strategy pattern on the communications
 Singleton on statemanager, monitor and watchdog
 Composite in the attributes

Python console example

```
>>> from skippylib import Skippy
>>> skippyObj = Skippy(name='scodilt0401', port=5025, nChannels=4)
>>> skippyObj.idn
'KEYSIGHT_TECHNOLOGIES,DSOS204A,MY58150181,06.30.00701'
>>> stateCh1 = skippyObj.attributes['StateCh1']
>>> print("{!r}".format(StateCh1))
StateCh1 (SkippyReadWriteAttribute):
  rvalue: True
  wvalue: None
  timestamp: 1559207397.3
  quality: ATTR_VALID
  type: DevBoolean
  dim: 0
  readCmd: ':CHAN1:DISPlay?'
  readFormula: None
  writeCmd: ':%s%d:DISPlay %s'
>>> stateCh1.isRampeable()
False
```

Software patterns

Strategy
Singleton
Composite in the attributes

- Master oscillator
 - Radiofrequency generator

- Master oscillator
 - Radiofrequency generator
- Measured Filling Pattern:
 - Oscilloscope with the filling pattern
 - Many Oscilloscopes in the accelerator with we can cross different source signals

- Master oscillator
 - Radiofrequency generator
- Measured Filling Pattern:
 - Oscilloscope with the filling pattern
 - Many Oscilloscopes in the accelerator with we can cross different source signals
- Tune excitation:
 - Arbitrary Function Generator
 - Spectrum analyzer

- Master oscillator
 - Radiofrequency generator
- Measured Filling Pattern:
 - Oscilloscope with the filling pattern
 - Many Oscilloscopes in the accelerator with we can cross different source signals
- Tune excitation:
 - Arbitrary Function Generator
 - Spectrum analyzer
- Beamline & Laboratory instruments
 - Source & Multimeters
 - Pump controller ¹
 - Temperature controller ²

¹LN2 pump where J.Andreu has had to write the server side scpi in C#

²It doesn't work as scpi but string-like protocol

- Master oscillator
 - Radiofrequency generator
- Measured Filling Pattern:
 - Oscilloscope with the filling pattern
 - Many Oscilloscopes in the accelerator with we can cross different source signals
- Tune excitation:
 - Arbitrary Function Generator
 - Spectrum analyzer
- Beamline & Laboratory instruments
 - Source & Multimeters
 - Pump controller ¹
 - Temperature controller ²
- Alba #Em & Music SiPM
 - We close the circle with the scplib

¹LN2 pump where J.Andreu has had to write the server side scpi in C#

²It doesn't work as scpi but string-like protocol



skippylib

- Improve new instrument insertion
- Improve the watchdog
- Dynamic attributes as property
- Dynamic commands
- Generalize *TxTerminator*
- Different ramp strategies
- WriteFormula
- Input validation
- Dependencies with state-like

skippylib

- Improve new instrument insertion
- Improve the watchdog
- Dynamic attributes as property
- Dynamic commands
- Generalize *TxTerminator*
- Different ramp strategies
- WriteFormula
- Input validation
- Dependencies with state-like

skippylib

- Improve new instrument insertion
- Improve the watchdog
- Dynamic attributes as property
- Dynamic commands
- Generalize *TxTerminator*
- Different ramp strategies
- WriteFormula
- Input validation
- Dependencies with state-like

scplib

- *autodoc* scpi tree
- python3
- Set of *minimal* commands
- Write lock (current is RW)
- Report locker owner
- Extend *lock* feature to subtrees
- Listen more channels than network
- SSL and ACLs
- Event subscription
- Multidimensional data

skippylib

- Improve new instrument insertion
- Improve the watchdog
- Dynamic attributes as property
- Dynamic commands
- Generalize *TxTerminator*
- Different ramp strategies
- WriteFormula
- Input validation
- Dependencies with state-like

scplib

- *autodoc* scpi tree
- python3
- Set of *minimal* commands
- Write lock (current is RW)
- Report locker owner
- Extend *lock* feature to subtrees
- Listen more channels than network
- SSL and ACLs
- Event subscription
- Multidimensional data

skippylib

- Improve new instrument insertion
- Improve the watchdog
- Dynamic attributes as property
- Dynamic commands
- Generalize *TxTerminator*
- Different ramp strategies
- WriteFormula
- Input validation
- Dependencies with state-like

scplib

- *autodoc* scpi tree
- **python3**
- Set of *minimal* commands
- Write lock (current is RW)
- Report locker owner
- Extend *lock* feature to subtrees
- Listen more channels than network
- SSL and ACLs
- Event subscription
- Multidimensional data

skippylib

- Improve new instrument insertion
- Improve the watchdog
- Dynamic attributes as property
- Dynamic commands
- Generalize *TxTerminator*
- Different ramp strategies
- WriteFormula
- Input validation
- Dependencies with state-like

scplib

- *autodoc* scpi tree
- python3
- Set of *minimal* commands
- Write lock (current is RW)
- Report locker owner
- Extend *lock* feature to subtrees
- Listen more channels than network
- SSL and ACLs
- Event subscription

gui

- Generic taurus gui for any of the instruments

dimensional data