

MAXIN

The image shows the word "MAXIN" in a bold, grey, sans-serif font. A vibrant yellow swoosh, resembling a stylized 'C' or a dynamic underline, curves over the letters 'A', 'X', and 'I'. The swoosh starts above the 'A', loops around the top of the 'X', and ends above the 'I', creating a sense of motion and energy.



# **“dsconfig”, a Tango configuration tool**

Johan Forsberg

Tango meeting 22-06-30

# What is dsconfig?

A command line tool that reads a configuration from a file and applies it to an existing Tango control system.

Depends on **PyTango**

Defines a **JSON** file format

<https://gitlab.com/MaxIV/lib-maxiv-dsconfig>

# JSON file format

```
"servers": {
  "some-server/instance": {
    "SomeDeviceClass": {
      "some/device/1": {
        "properties": {
          "someImportantProperty": [
            "foo",
            "bar"
          ],
          "otherProperty": ["7"]
        },
        "attribute_properties": {
          "anAttribute": {
            "min_value": ["-5"],
            "unit": ["mV"]
          }
        }
      }
    }
  }
  "some/device/2": { ...
```

# How it works

1. Read and validate the JSON file
2. “Dump” the relevant parts of the Tango database
3. Compare the two configurations
  - If there is no difference, we are already done!
4. Display the “diff” to the user
  - If the “--write” flag was not added, stop here.
5. Add/remove/change in the database what is needed for the user supplied configuration to be “true”.

# What is the point?

Tango DB is a “moving target”; automated configuration may **overwrite** important manual changes

Applying configuration with **deployment tools** like Ansible (see Benjamin’s talk yesterday).

**Developers** can generate and reproduce testing setups. Also useful for automated testing, e.g. in CI.

Convenient way to get a “**snapshot**” of the database for backup or further analysis.

# Example session

```
$ conda create -n dsconfig -c conda-forge python=3.10 dsconfig
```

```
$ conda activate
```

```
$ json2tango --help
```

```
Usage: json2tango [options] JSONFILE
```

Options:

-h, --help	show this help message and exit
-w, --write	write to the Tango DB
-u, --update	don't remove things, only add/update
-c, --case-sensitive	Don't ignore the case of server, device,

...

# Example session

```
johfor@w-johfor-pc-0 ~/kits/meetings/tango_meeting_2022
$ cat dsconfig_example.json
{
  "servers": {
    "TangoTest/test": {
      "TangoTest": {
        "sys/tg_test/1": {}
      }
    }
  }
}
```

```
johfor@w-johfor-pc-0 ~/kits/meetings/tango_meeting_2022
$ json2tango dsconfig_example.json
```

```
*** No changes needed in Tango DB ***
```



# Example session

```
$ cat dsconfig_example2.json
{
  "servers": {
    "TangoTest/test": {
      "TangoTest": {
        "sys/tg_test/1": {
          "properties": {
            "message": ["Hello tangoers!"]
          }
        },
        "sys/tg_test/2": {}
      }
    }
  }
}
```

# Example session

```
$ json2tango dsconfig_example2.json
= Device: sys/tg_test/1
  Properties:
    + message
      Hello tangoers!
+ Device: sys/tg_test/2
  Server: TangoTest/test
  Class: TangoTest

Summary:
Add 1 devices to 1 servers.
Add/change 1 device properties in 1 devices.

*** Nothing was written to the Tango DB (use -w) ***
```

# Example session

```
johfor@w-johfor-pc-0 ~/kits/meetings/tango_meeting_2022
$ json2tango dsconfig_example2.json -w
= Device: sys/tg_test/1
  Properties:
    + message
      Hello tangoers!
+ Device: sys/tg_test/2
  Server: TangoTest/test
  Class: TangoTest
Progress: [#####] 100%

Summary:
Add 1 devices to 1 servers.
Add/change 1 device properties in 1 devices.

*** Data was written to the Tango DB ***
The previous DB data was saved to /tmp/dsconfig-kfcm6ke3.json

johfor@w-johfor-pc-0 ~/kits/meetings/tango_meeting_2022
$ json2tango dsconfig_example2.json

*** No changes needed in Tango DB ***
```

-w flag required  
for writing to DB

# Use case: plc2tango

At MAX IV, PLC handles vacuum interlocks, cooling flows, temperatures, etc

Need to make this info available via Tango for data acquisition, GUIs, archiving...

**Facade** devices, a "low code" solution to create devices combining info from various parts of the control system, configured via properties

→ Large numbers of different devices that need frequent updates.

Fortunately: **naming convention!**

# Use case: plc2tango

Example: A thermocouple

**B316A-001-DIA-TCO-04**

**B316A** = **System**: beamline on R3, number 16, branch A

**001** = **Location**: optics area 1

**DIA** = **Subsystem**: diagnostics

**TCO** = **Equipment type**: thermocouple

**04** = Fourth in order

PLC tags:

**B316A\_001\_DIA\_TC004\_AS** Temperature value

**B316A\_001\_DIA\_TC004\_BP** Bypassed

**B316A\_001\_DIA\_TC004\_A01\_AA\_HHInAlarm** High alarm

...

Tango device name: **B316A-O01/DIA/TCO-04**

# Use case: plc2tango

Tag list (CSV file) is  
parsed according to  
the naming  
convention and  
turned into a dsconfig  
file ->

```
{
  "servers": {
    "Thermocouple": {
      "B316A-DIA": {
        "Thermocouple": {
          "B316A-O01/DIA/TCO-04": {
            "properties": {
              "AlarmsDesc": [
                "B316A_O01_DIA_TCO04_A01_AD__InAlarm:TCO channel fault"
              ],
              "AlarmsList": [
                "B316A/VAC/PLC-01/B316A_O01_DIA_TCO04_A01_AD__InAlarm"
              ],
              "AlarmsReset": [
                "False"
              ],
              "ByPassedAttribute": [
                "B316A/VAC/PLC-01/B316A_O01_DIA_TCO04_BP"
              ],
              "HighAlarmAttribute": [
                "B316A/VAC/PLC-01/B316A_O01_DIA_TCO04_A01_AA__HHInAlarm"
              ],
              "HighAlarmLevelAttribute": [
                "B316A/VAC/PLC-01/B316A_O01_DIA_TCO04_A01_AA__HHLimit"
              ],
              "HighWarningAttribute": [
                "B316A/VAC/PLC-01/B316A_O01_DIA_TCO04_A01_AA__HInAlarm"
              ],
              ...
            }
          }
        }
      }
    }
  }
}
```

# Use case: plc2tango

## PLC2TANGO

PLC2Tango: v3.6.1  
Object Tag Definition: v3.6

### Upload your CSV file

System

B108A - PLC-01 (Species)

Equipment classes (High Level Devices) - [All](#) / [None](#)

<input checked="" type="checkbox"/> Beamshutter	<input checked="" type="checkbox"/> Circulation Fan	<input checked="" type="checkbox"/> Compressor
<input checked="" type="checkbox"/> Conductivity Sensor	<input checked="" type="checkbox"/> Control Valve	<input checked="" type="checkbox"/> Differential Pressure Sensor
<input checked="" type="checkbox"/> Fast Acting Valve	<input checked="" type="checkbox"/> Filter Actuator	<input checked="" type="checkbox"/> Flow Gauge

Simple web service allows PLC engineers to upload and apply new tag lists without specific knowledge about Tango

Runs dsconfig in the background

**Simulate deploy**

Runs a simulated deploy to the Tango Database (non destructive). The result describes the diff in configuration.

**Deploy**

Deploys the configuration to the Tango Database. The result describes the diff in configuration.

You must login to deploy. [Log In](#)

## Result

```
CONFIG_NOT_APPLIED

-----

- Device: B108A-D100830CAB60/VAC/PRC-01
  Server: ForeVacuumPump/B108A-VAC
  Class: ForeVacuumPump
+ Device: B108A-D100830CAB60/VAC/PRC-03
  Server: ForeVacuumPump/B108A-VAC
  Class: ForeVacuumPump
Properties:
+ AlarmsDesc
  B108A_D100830CAB60_VAC_PRC03_A01_AD__InAlarm:Pump is not At Speed, not pumping
B108A_D100830CAB60_VAC_PRC03_A02_AD__InAlarm:communication Fault to ACP Pump$N
+ AlarmsList
  B108A/VAC/PLC-01/B108A_D100830CAB60_VAC_PRC03_A01_AD__InAlarm
  B108A/VAC/PLC-01/B108A_D100830CAB60_VAC_PRC03_A02_AD__InAlarm
+ AlarmsReset
  False
  False
+ ByPassedAttribute
  B108A/VAC/PLC-01/B108A_D100830CAB60_VAC_PRC03_BP
+ ConstantSpeedAttribute
  B108A/VAC/PLC-01/B108A_D100830CAB60_VAC_PRC03_PAS
+ DisableStandbyCommand
  B108A/VAC/PLC-01/B108A_D100830CAB60_VAC_PRC03_SBSTOC
+ EnableStandbyCommand
  B108A/VAC/PLC-01/B108A_D100830CAB60_VAC_PRC03_SBSTAC
+ EnabledAttribute
  B108A/VAC/PLC-01/B108A_D100830CAB60_VAC_PRC03_ENAS
```

# Use case: Ansible

“**declarative**”: describe the config you want, and let the tool figure out how to get there.

“**Idempotent**”: do only what is needed to get to the desired state.

→ good fit for **Ansible**!

Our Ansible role for deploying Tango devices uses dsconfig as a python module.

(For more information about how we use Ansible at MAX IV, see Benjamin’s talk yesterday.)



# How to get it

Code:

<https://gitlab.com/MaxIV/lib-maxiv-dsconfig>

PyPI:

```
$ pip install dsconfig
```

Conda:

“dsconfig” on conda-forge

RPM:

<https://gitlab.com/tango-controls/RPM/dsconfig-spec>

**Thanks!**