The European Synchrotron

Matlab: Tango object layer

L. Farvacque / ESRF

MOTIVATION

Existing Tango-Matlab binding

- Almost complete image of the C++ interface
- Already used for long
- Functional interface

```
info = tango_command_inout('sys/database/2', 'DbGetServerNameList', 'A*');
%- always check error
if (tango_error == -1)
    %- handle error
    tango_print_error_stack;
end
```

Matlab has a mostly unexploited object support

⇒ Build a Tango Object interface on top of the existing tango binding

- Advantages:
  - Use existing binding
  - Pure Matlab code, very little code to write
  - Easy to use, closer to Matlab style

```
try
    info=tango.Device('sys/database/1').DbGetServerNameList('A*')
catch err
    disp(char(err))
end
```

MATLAB CLASS HIERARCHY

Main classes:

cs.Device (abstract)	
tango.Base (abstract)	
tango.Device	Base class for generic Tango devices
tango.Group	Group access to commands and attributes of several similar devices
taco…	…
cs.DevState (abstract)	
tango.DevState	Enumeration class representing device states
taco…	…
cs.DevError (abstract)	
tango.Error	Exception thrown by a Tango access error
taco…	…

Helper classes:

tango.Type	Enumeration class representing Tango data types
tango.AttrQuality	Enumeration class representing Tango attribute quality
tango.AttrWriteType	Enumeration class representing Tango attribute writable property
tango.AttrDataFormat	Enumeration class representing Tango attribute format

CLASS “TANGO.DEVICE”

Constructor: dev=tango.Device(device_name)

```
>> devtest=tango.Device('mcs/tango/test’)
devtest =
tango.Device('mcs/tango/test')
>>
```

Array of devices:

```
>> bpms=tango.Device('SR/D-BPMLIBERA/C4-1’,'SR/D-BPMLIBERA/C4-2’,'SR/D-BPMLIBERA/C4-3');
>> bpms.State
ans =
    On
ans =
    On
ans =
    On
>> bpmstates=cat(1,bpms.State)
bpmstates =
    On
    On
    On
>> whos bpm*
  Name           Size              Bytes  Class              Attributes

  bpms           3x1                 128  tango.Device
  bpmstates      3x1                 110  tango.DevState
>>
```

Information on attributes:

```
>> devtest.attributes()
Attributes for device mcs/tango/test
                    ampli  (   SCALAR,DEV_DOUBLE  )
            boolean_scalar  (   SCALAR,DEV_BOOLEAN )
             double_scalar  (   SCALAR,DEV_DOUBLE  )
         double_scalar_rww  (   SCALAR,DEV_DOUBLE  )
           double_scalar_w  (   SCALAR,DEV_DOUBLE  )
              float_scalar  (   SCALAR,DEV_FLOAT   )
             long64_scalar  (   SCALAR,DEV_LONG64  )
                       ...
           double_spectrum  ( SPECTRUM,DEV_DOUBLE  )
        double_spectrum_ro  ( SPECTRUM,DEV_DOUBLE  )
            float_spectrum  ( SPECTRUM,DEV_FLOAT   )
         float_spectrum_ro  ( SPECTRUM,DEV_FLOAT   )
                       ...
              double_image  (    IMAGE,DEV_DOUBLE  )
            double_image_ro  (    IMAGE,DEV_DOUBLE  )
               float_image  (    IMAGE,DEV_FLOAT   )
>>
```

TANGO ATTRIBUTES

```
>> devtest.attrinfo('double_scalar')
ans =
                    name: 'double_scalar'
                   max_m: 1
                   max_n: 1
             description: 'No description'
                   label: 'double_scalar'
                    unit: 'No unit'
           standard_unit: 1
            display_unit: 1
                  format: '%6.2f'
               min_value: 'Not specified'
               max_value: 'Not specified'
               min_alarm: 'Not specified'
               max_alarm: 'Not specified'
    writable_attr_name: 'double_scalar'
              extensions: []
              disp_level: 0
                writable: [1x1 tango.AttrWriteType]
             data_format: [1x1 tango.AttrDataFormat]
               data_type: [1x1 tango.Type]
>>
```

ACCESS TO TANGO ATTRIBUTES – SYNTAX 1

dev.set_attribute('attrname',value)

```
>> devtest.set_attribute('double_scalar',33)
```

v = dev.get_attribute('attrname1','attrname2,…)

```
>> v=devtest.get_attribute('double_scalar','short_scalar')
v =
1x2 struct array with fields:
    set
    read
    time
    quality
    error

>> cat(1,v.set)
ans =
     33
      0

>> datestr(cat(1,v.time))
ans =
15-May-2014 14:17:05
15-May-2014 14:17:05

>> cat(1,v.quality)
ans =
    VALID
    VALID
>>
```

ACCESS TO TANGO ATTRIBUTES – SYNTAX 2

dev. attrname=value

v = dev.attrname

```
>> devtest.double_spectrum=1:10;
>> devtest.double_spectrum
ans =
        set: [1 2 3 4 5 6 7 8 9 10]
       read: [1 2 3 4 5 6 7 8 9 10]
       time: 7.3574e+05
    quality: [1x1 tango.AttrQuality]
      error: 0
```

The attribute name may be a Matlab string variable:

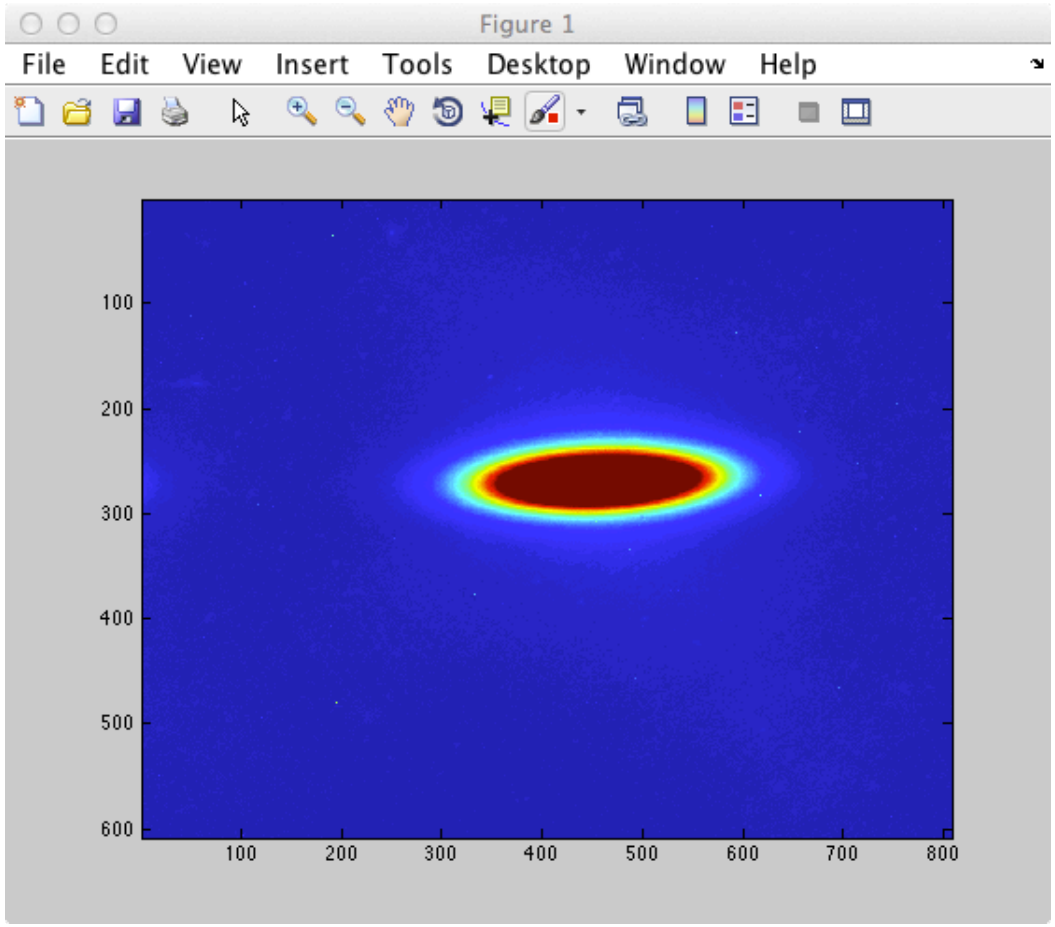
```
>> attr='string_scalar';
>> devtest.(attr)='sample string';
>> devtest.(attr)
ans =
        set: ''
       read: 'sample string'
       time: 7.3574e+05
    quality: [1x1 tango.AttrQuality]
      error: 0
```

Caution:

In that syntax, the attribute name is case sensitive !

Example:

```
>> image(tango.Device('sr/d-emit/d9-a').Image.read)
```



Information on commands:

```
>> devtest=tango.Device('mcs/tango/test')
devtest =
tango.Device('mcs/tango/test')

>> devtest.commands()
Commands for device mcs/tango/test
              DevBoolean  (DEV_BOOLEAN,DEV_BOOLEAN)
               DevDouble  (DEV_DOUBLE,DEV_DOUBLE)
                DevFloat  (DEV_FLOAT,DEV_FLOAT)
                     ...
               DevString  (DEV_STRING,DEV_STRING)
                    Init  (DEV_VOID,DEV_VOID)
                   State  (DEV_VOID,DEV_STATE)
                  Status  (DEV_VOID,DEV_STRING)
            SwitchStates  (DEV_VOID,DEV_VOID)

>> devtest.cmdinfo('DevDouble')
ans =
          cmd_name: 'DevDouble'
           cmd_tag: 0
      in_type_desc: 'Any DevDouble value'
     out_type_desc: 'Echo of the argin value'
        disp_level: 0
    disp_level_str: 'Operator'
           in_type: [1x1 tango.Type]
          out_type: [1x1 tango.Type]
>>
```

EXECUTION OF TANGO COMMANDS

output_arg = dev.cmd('CommandName',input_arg)

```
>> a1=devtest.cmd('DevDouble',42)
a1 =
    42
>>
```

output_arg = dev.CommandName(input_arg)

```
>> a2=devtest.DevShort(int16(3))
a2 =
     3
>> cmdname='DevString';
>> a3=devtest.(cmdname)('abcd')
a3 =
abcd
>> whos a*
  Name      Size            Bytes  Class     Attributes

  a1        1x1                 8  double
  a2        1x1                 2  int16
  a3        1x4                 8  char
```

DERIVATION

```
classdef (ConstructOnLoad) TestDevice < tango.Device

    properties(Dependent=true)   % New property <=> pseudo-attribute
        twice_double_scalar
    end

    methods
        % Constructor
        function dev=TestDevice(varargin)
            dev=dev@tango.Device(varargin{:});
        end

        % Access to the new property
        function v=get.twice_double_scalar(dev)
            v=2*dev.double_scalar.set;
        end
        function set.twice_double_scalar(dev,value)
            dev.double_scalar=2*value;
        end

        % New method
        function [v1,v2]=doublecmd(dev,cmd1,cmd2,arg1,arg2)
            v1=dev.cmd(cmd1,arg1); % Syntax 1
            v2=dev.(cmd2)(arg2);   % Syntax 2
        end
    end
end
```

EXAMPLE OF DERIVATION

```
>> tg=tango.TestDevice('mcs/tango/test')
tg =
tango.TestDevice('mcs/tango/test')

>> [a,b]=tg.doublecmd('DevDouble','DevShort',45,int16(4))
a =
    45
b =
     4

>> tg.double_scalar=2;
>> tg.twice_double_scalar
ans =
     4

>> tg.twice_double_scalar=8;
>> tg.double_scalar.set
ans =
    16
>>
```

CLASS “TANGO.DEVSTATE”

Human readable state information:

```
>> state=devtest.State
state =
    Run

>> state==tango.DevState.On
ans =
     0
```

Standard conversions are implemented:

```
>> [char(devtest),': ', char(devtest.State)]
ans =
mcs/tango/test: Run

>> double(tango.DevState.Off)
ans =
     1

>> tango.DevState(3)
ans =
    Open
>>
```

Synonyms are allowed:

```
>> tango.DevState.Run==tango.DevState.Running
ans =
     1
```

Convenience methods:

```
>> state.color()
ans =
         0    0.5000         0

>>
```

Notion of “severity”: “Fault” is worse (larger) than “Alarm”

Ordering operators act on the severity:

```
>> tango.DevState.On > tango.DevState.Alarm
ans =
     0
```

Combining states is easy:

```
>> states=enumeration('tango.DevState')'
states =
  Columns 1 through 6
    On           Off          Closed       Open         Inserted     Extracted
  Columns 7 through 12
    Moving       Standby      Fault        Init         Run          Alarm
  Columns 13 through 14
    Disabled     Unknown

>> globalstate=max(states(1:10))
globalstate =
    Fault
```

Standard Matlab function work as expected:

```
>> sort(states)
ans =
  Columns 1 through 6
    On           Open         Extracted    Run          Standby      Off
  Columns 7 through 12
    Moving       Init         Closed       Inserted     Alarm        Disabled
  Columns 13 through 14
    Fault        Unknown
>>
```

CLASS “TANGO.GROUP”

A tango.Group allows a group access to commands and attributes of several similar devices:

```
>> bpms=tango.Group('sr/d-bpmlibera/c6-1','sr/d-bpmlibera/c6-2','sr/d-bpmlibera/c6-3')
bpms =
|- GROUP: group1 [3:3]
          |- DEVICE: sr/d-bpmlibera/c6-1
          |- DEVICE: sr/d-bpmlibera/c6-2
          |- DEVICE: sr/d-bpmlibera/c6-3
```

```
>> bpms.State
ans =
    On           On           On
```

```
>> bpms.XPosSA
ans =
        set: [NaN NaN NaN]
       read: [-0.0053 0.0296 -0.1760]
       time: [7.3574e+05 7.3574e+05 7.3574e+05]
    quality: [1x3 tango.AttrQuality]
      error: {[0]  [0]  [0]}
```

OTHER FEATURES

- **Asynchronous access**

  Both Device and Group have asynchronous commands and attribute access

- **Access to properties**

- **Online documentation**

```
>> doc tango.Device
>> help tango.Device.get_attribute
```

- **Object persistence (for Devices, not yet for Groups)**

  A tango.Device object can be saved in a .mat file and is working immediately when loaded

```
>> ct=tango.Device('sr/d-ct/1');
>> save ctfile ct

>> load ctfile
>> ct.Current.read
ans =
   79.0678
>>
```

```
>> ct2=tango.Device('sr/d-ct/2')
Error using tango.Base/set.tangoname (line 84)
Error evaluating tango_open_device('sr/d-ct/2'):
*************************************************************
*                     TANGO ERROR STACK                     *
*************************************************************
- ERROR 1
    |-reason.....DB_DeviceNotDefined
    |-desc.......device sr/d-ct/2 not defined in the database !
    |-origin.....DataBase::ImportDevice()
    |-severity...Error (1)
- ERROR 2
    |-reason.....API_CommandFailed
    |-desc.......Failed to execute command_inout on device sys/database/2, command DbImportDevice
    |-origin.....Connection::command_inout()
    |-severity...Error (1)
- ERROR 3
    |-reason.....API_DeviceNotDefined
    |-desc.......Can't connect to device sr/d-ct/2
    |-origin.....DeviceProxy::DeviceProxy
    |-severity...Error (1)
- ERROR 4
    |-reason.....Tango::DeviceProxy instanciation failed
    |-desc.......failed to create proxy for device sr/d-ct/2
    |-origin.....DevRepository::device_desc
    |-severity...Error (1)
- ERROR 5
    |-reason.....failed to open device sr/d-ct/2
    |-desc.......could not obtain a valid device reference
    |-origin.....TangoBinding::open_device
    |-severity...Error (1)
*************************************************************

Error in tango.Device (line 82)
                    dev(i).tangoname=allnames{i}; %#ok<AGROW>
>>
```