TANGO Device Server

MicrocodeDataViewer User's Guide

MicrocodeDataViewer Class

Revision: release_1_0_0 - Author: dhaussy
Implemented in C++

Introduction:

This device allows to read/write microcode variables of a GalliBox using dynamic attributes defined in properties

Class Inheritance:

- Tango::Device_3Impl
  - MicrocodeDataViewer

Properties:

Device Properties		
Property name	**Property type**	**Description**
MicrocodeName	Tango::DEV_STRING	The GalliBox microcode name
ProcessId	Tango::DEV_STRING	The GalliBox microcode process id
GalilBoxDeviceName	Tango::DEV_STRING	The complete name of the GalilBox device
Data	Array of string	Describe the data that should be read/write on the GalilBox. For each line the device will create an attribute with the specified name and type. Format ::: : the name of the attribute that will be created : BOOLEAN, LONG, DOUBLE : READ or READ_WRITE : the name of the microcode var
ControlboxCommunicationCommand	Tango::DEV_STRING	The tango command to use to communicate with the control box. Microcode V1.0 : WriteRead Microcode V2.0 : ExecLowLevelCmd Default : ExecLowLevelCmd

Device Properties Default Values:

Property Name	**Default Values**
MicrocodeName	No default value
ProcessId	No default value
GalilBoxDeviceName	Test/GalilBox/dev1
Data	No default value
ControlboxCommunicationCommand	ExecLowLevelCmd

There is no Class properties.

States:

States	
Names	**Descriptions**
RUNNING	The microcode is RUNNING
STANDBY	The microcode is NOT RUNNING

Commands:

More Details on commands....

Device Commands for Operator Level		
Command name	**Argument In**	**Argument Out**
Init	DEV_VOID	DEV_VOID
State	DEV_VOID	DEV_STATE
Status	DEV_VOID	CONST_DEV_STRING
Start	DEV_VOID	DEV_VOID

Device Commands for Expert Level Only		
Command name	**Argument In**	**Argument Out**
Abort	DEV_VOID	DEV_VOID

1 - Init

- **Description:** This commands re-initialise a device keeping the same network connection.
  After an Init command executed on a device, it is not necessary for client to re-connect to the device.
  This command first calls the device *delete_device()* method and then execute its *init_device()* method.
  For C++ device server, all the memory allocated in the *nit_device()* method must be freed in the *delete_device()* method.
  The language device desctructor automatically calls the *delete_device()* method.

- **Argin:**
  DEV_VOID : none.

- **Argout:**
  DEV_VOID : none.

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

2 - State

- **Description:** This command gets the device state (stored in its *device_state* data member) and returns it to the caller.

- **Argin:**
  DEV_VOID : none.

- **Argout:**
  DEV_STATE : State Code

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

3 - Status

- **Description:** This command gets the device status (stored in its *device_status* data member) and returns it to the caller.

- **Argin:**
  DEV_VOID : none.

- **Argout:**
  CONST_DEV_STRING : Status description

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

4 - Start

- **Description:** Start the microcode

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

5 - Abort (for expert only)

- **Description:** Abort the microcode execution. CAUTION : This function kills the running process, which is not the proper way to stop. USE WITH CARE

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

TANGO Device Server

MicrocodeDataViewer User's Guide

MicrocodeDataViewer Class

Revision: release_1_0_0 - Author: dhaussy
Implemented in C++

Introduction:

This device allows to read/write microcode variables of a GalliBox using dynamic attributes defined in properties

Class Inheritance:

- Tango::Device_3Impl
  - MicrocodeDataViewer

Properties:

Device Properties

Property name	Property type	Description
MicrocodeName	Tango::DEV_STRING	The GalliBox microcode name
ProcessId	Tango::DEV_STRING	The GalliBox microcode process id
GalilBoxDeviceName	Tango::DEV_STRING	The complete name of the GalilBox device
Data	Array of string	Describe the data that should be read/write on the GalilBox. For each line the device will create an attribute with the specified name and type. Format ::: : the name of the attribute that will be created : BOOLEAN, LONG, DOUBLE : READ or READ_WRITE : the name of the microcode var
ControlboxCommunicationCommand	Tango::DEV_STRING	The tango command to use to communicate with the control box. Microcode V1.0 : WriteRead Microcode V2.0 : ExecLowLevelCmd Default : ExecLowLevelCmd

Device Properties Default Values:

Property Name	**Default Values**
MicrocodeName	No default value
ProcessId	No default value
GalilBoxDeviceName	Test/GalilBox/dev1
Data	No default value
ControlboxCommunicationCommand	ExecLowLevelCmd

There is no Class properties.

States:

States

Names	Descriptions
RUNNING	The microcode is RUNNING
STANDBY	The microcode is NOT RUNNING

Commands:

More Details on commands....

Device Commands for Operator Level		
Command name	**Argument In**	**Argument Out**
Init	DEV_VOID	DEV_VOID
State	DEV_VOID	DEV_STATE
Status	DEV_VOID	CONST_DEV_STRING
Start	DEV_VOID	DEV_VOID

Device Commands for Expert Level Only		
Command name	**Argument In**	**Argument Out**
Abort	DEV_VOID	DEV_VOID

1 - Init

- **Description:** This commands re-initialise a device keeping the same network connection.
After an Init command executed on a device, it is not necessary for client to re-connect to the device.
This command first calls the device *delete_device()* method and then execute its *init_device()* method.
For C++ device server, all the memory allocated in the *nit_device()* method must be freed in the *delete_device()* method.
The language device desctructor automatically calls the *delete_device()* method.

- **Argin:**
DEV_VOID : none.

- **Argout:**
DEV_VOID : none.

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

2 - State

- **Description:** This command gets the device state (stored in its *device_state* data member) and returns it to the caller.
- **Argin:**
  DEV_VOID : none.
- **Argout:**
  DEV_STATE : State Code
- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

3 - Status

- **Description:** This command gets the device status (stored in its *device_status* data member) and returns it to the caller.
- **Argin:**
  DEV_VOID : none.
- **Argout:**
  CONST_DEV_STRING : Status description
- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

4 - Start

- **Description:** Start the microcode
- **Argin:**
  DEV_VOID :
- **Argout:**
  DEV_VOID :
- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

5 - Abort (for expert only)

- **Description:** Abort the microcode execution. CAUTION : This function kills the running process, which is not the proper way to stop. USE WITH CARE

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

TANGO Device Server

MicrocodeDataViewer
Device Commands Description
MicrocodeDataViewer Class

Revision: release_1_0_0 - Author: dhaussy

1 - Init

- **Description:** This commands re-initialise a device keeping the same network connection.
  After an Init command executed on a device, it is not necessary for client to re-connect to the device.
  This command first calls the device *delete_device()* method and then execute its *init_device()* method.
  For C++ device server, all the memory allocated in the *nit_device()* method must be freed in the *delete_device()* method.
  The language device desctructor automatically calls the *delete_device()* method.

- **Argin:**
  DEV_VOID : none.

- **Argout:**
  DEV_VOID : none.

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

2 - State

- **Description:** This command gets the device state (stored in its *device_state* data member) and returns it to the caller.

- **Argin:**
  DEV_VOID : none.

- **Argout:**
  DEV_STATE : State Code

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

3 - Status

- **Description:** This command gets the device status (stored in its *device_status* data member) and returns it to the caller.

- **Argin:**
  DEV_VOID : none.

- **Argout:**
  CONST_DEV_STRING : Status description

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

4 - Start

- **Description:** Start the microcode

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**
  - Tango::RUNNING
  - Tango::STANDBY

5 - Abort (for expert only)

- **Description:** Abort the microcode execution. CAUTION : This function kills the running process, which is not the proper way to stop. USE WITH CARE

- **Argin:**
  DEV_VOID :

- **Argout:**
  DEV_VOID :

- **Command allowed for:**

- Tango::RUNNING
- Tango::STANDBY

ESRF - Software Engineering Group