

Reference Manual

3.0.2

Generated by Doxygen 1.5.3

Wed Nov 5 16:05:39 2014

Contents

1	The Tango C Language Binding	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	Module Documentation	7
4.1	Tango C Binding Enumerations	7
4.2	Tango Data Type Related Definitions	11
4.3	Tango C Binding Data Structures	15
4.4	Tango Proxy Related Functions	18
4.5	Tango Command Related Functions	23
4.6	Tango Attribute Related Functions	26
4.7	Error Handling Related Functions	30
4.8	Tango Property and Database Related Functions	31
5	Data Structure Documentation	37
5.1	AttributeData Struct Reference	37
5.2	AttributeDataList Struct Reference	38
5.3	AttributeInfo Struct Reference	39
5.4	AttributeInfoList Struct Reference	41
5.5	CommandData Struct Reference	42
5.6	CommandInfo Struct Reference	43
5.7	CommandInfoList Struct Reference	44
5.8	DbData Struct Reference	45
5.9	DbDatum Struct Reference	46
5.10	DevFailed Struct Reference	47
5.11	ErrorStack Struct Reference	48

5.12 TangoAttributeData Union Reference	49
5.13 TangoCommandData Union Reference	50
5.14 TangoDevEncoded Struct Reference	51
5.15 TangoPropertyData Union Reference	52
5.16 VarBoolArray Struct Reference	53
5.17 VarCharArray Struct Reference	54
5.18 VarDoubleArray Struct Reference	55
5.19 VarDoubleStringArray Struct Reference	56
5.20 VarEncodedArray Struct Reference	57
5.21 VarFloatArray Struct Reference	58
5.22 VarLong64Array Struct Reference	59
5.23 VarLongArray Struct Reference	60
5.24 VarLongStringArray Struct Reference	61
5.25 VarShortArray Struct Reference	62
5.26 VarStateArray Struct Reference	63
5.27 VarStringArray Struct Reference	64
5.28 VarULong64Array Struct Reference	65
5.29 VarULongArray Struct Reference	66
5.30 VarUShortArray Struct Reference	67

Chapter 1

The Tango C Language Binding

The Tango C language binding is a reduced C interface which wraps the Tango C++ API. The actual binding only contains the basic query functionality and the basic synchronous reading and writing of commands and attributes.

The API is structured in proxy related functions, command related functions and attribute related functions.

Tango Proxy Related Functions (p.18)

Tango Command Related Functions (p. 23)

Tango Attribute Related Functions (p. 26)

Tango Property and Database Related Functions (p.31)

Error Handling Related Functions (p. 30)

Tango C Binding Enumerations (p. 7)

Tango Data Type Related Definitions (p. 11)

Tango C Binding Data Structures (p. 15)

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Tango C Binding Enumerations	7
Tango Data Type Related Definitions	11
Tango C Binding Data Structures	15
Tango Proxy Related Functions	18
Tango Command Related Functions	23
Tango Attribute Related Functions	26
Error Handling Related Functions	30
Tango Property and Database Related Functions	31

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

AttributeData (A structure containing the scalar Tango data type and the attribute data union to transfer attribute data to and from a server)	37
AttributeDataList (A structure containing a pointer to a sequence of attribute data structures and the number of elements in the sequence)	38
AttributeInfo (The attribute info structure contains descriptive attribute properties)	39
AttributeInfoList (A structure containing a pointer to a sequence of attribute info structures and the number of elements in the sequence)	41
CommandData (A structure containing the Tango data type and the command data union to transfer command data to and from a server)	42
CommandInfo (The command info structure contains descriptive command properties)	43
CommandInfoList (A structure containing a pointer to a sequence of command info structures and the number of elements in the sequence)	44
DbData (A structure containing a pointer to a sequence of DbDatum (p. 46) structures and the number of elements in the sequence)	45
DbDatum (A container structure for the Tango database access)	46
DevFailed (A structure that maps all fields of the Tango::DevFailed exception)	47
ErrorStack (A structure containing a pointer to a sequence of error structures and the number of elements in the sequence)	48
TangoAttributeData (An union of all Tango array data types used for attribute reading and writing)	49
TangoCommandData (An union of all Tango scalar and array data types used for command data reading and writing)	50
TangoDevEncoded (A structure containing a data description string and a pointer to the data buffer)	51
TangoPropertyData (An union of all Tango scalar and array data types used for property reading and writing)	52
VarBoolArray (A structure containing a pointer to a sequence of boolean values and the number of elements in the sequence)	53
VarCharArray (A structure containing a pointer to a sequence of char values and the number of elements in the sequence)	54
VarDoubleArray (A structure containing a pointer to a sequence of double values and the number of elements in the sequence)	55

VarDoubleStringArray (A structure containing a pointer to a sequence of double and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence)	56
VarEncodedArray (A structure containing a pointer to a sequence of TangoDevEncoded (p. 51) values and the number of elements in the sequence)	57
VarFloatArray (A structure containing a pointer to a sequence of float values and the number of elements in the sequence)	58
VarLong64Array (A structure containing a pointer to a sequence of 64 bit long values and the number of elements in the sequence)	59
VarLongArray (A structure containing a pointer to a sequence of 32 bit long values and the number of elements in the sequence)	60
VarLongStringArray (A structure containing a pointer to a sequence of long and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence)	61
VarShortArray (A structure containing a pointer to a sequence of short values and the number of elements in the sequence)	62
VarStateArray (A structure containing a pointer to a sequence of TangoDevState values and the number of elements in the sequence)	63
VarStringArray (A structure containing a pointer to a sequence of strings and the number of elements in the sequence)	64
VarULong64Array (A structure containing a pointer to a sequence of 64 bit unsigned long values and the number of elements in the sequence)	65
VarULongArray (A structure containing a pointer to a sequence of 32 bit unsigned long values and the number of elements in the sequence)	66
VarUShortArray (A structure containing a pointer to a sequence of unsigned short values and the number of elements in the sequence)	67

Chapter 4

Module Documentation

4.1 Tango C Binding Enumerations

4.1.1 Detailed Description

All enumerations used in the Tango C binding.

Enumerations

- enum **TangoDataType** {
DEV_VOID = 0, DEV_BOOLEAN, DEV_SHORT, DEV_LONG,
DEV_FLOAT, DEV_DOUBLE, DEV_USHORT, DEV_ULONG,
DEV_STRING, DEVVAR_CHARARRAY, DEVVAR_SHORTARRAY,
DEVVAR_LONGARRAY,
DEVVAR_FLOATARRAY, DEVVAR_DOUBLEARRAY, DEVVAR_-
USHORTARRAY, DEVVAR_ULONGARRAY,
DEVVAR_STRINGARRAY, DEVVAR_LONGSTRINGARRAY, DEVVAR_-
DOUBLESTRINGARRAY, DEV_STATE,
CONST_DEV_STRING, DEVVAR_BOOLEANARRAY, DEV_UCHAR,
DEV_LONG64,
DEV_ULONG64, DEVVAR_LONG64ARRAY, DEVVAR_-
ULONG64ARRAY, DEV_INT,
DEV_ENCODED }
• enum **TangoDevState** {
ON, OFF, CLOSE, OPEN,
INSERT, EXTRACT, MOVING, STANDBY,
FAULT, INIT, RUNNING, ALARM,
DISABLE, UNKNOWN }
• enum **AttrQuality** {
ATTR_VALID, ATTR_INVALID, ATTR_ALARM, ATTR_CHANGING,
ATTR_WARNING }

- enum **AttrWriteType** { **READ**, **READ_WITH_WRITE**, **WRITE**, **READ_WRITE** }
- enum **AttrDataFormat** { **SCALAR**, **SPECTRUM**, **IMAGE** }
- enum **DispLevel** { **OPERATOR**, **EXPERT** }
- enum **ErrSeverity** { **WARN**, **ERR**, **PANIC** }
- enum **DevSource** { **DEV**, **CACHE**, **CACHE_DEV** }

4.1.2 Enumeration Type Documentation

4.1.2.1 enum TangoDataType

All available Tango data types.

The list of all available Tango data types. Scalar types and array types.

Enumerator:

DEV_VOID void
DEV_BOOLEAN bool
DEV_SHORT short
DEV_LONG int (32bits)
DEV_FLOAT float
DEV_DOUBLE double
DEV_USHORT unsigned short
DEV_ULONG unsigned long
DEV_STRING char *
DEVVAR_CHARARRAY array of unsigned char
DEVVAR_SHORTARRAY array of short
DEVVAR_LONGARRAY array of int (32bits)
DEVVAR_FLOATARRAY array of float
DEVVAR_DOUBLEARRAY array of double
DEVVAR_USHORTARRAY array of unsigned short
DEVVAR_ULONGARRAY array of unsigned int (32bits)
DEVVAR_STRINGARRAY array of char *
DEVVAR_LONGSTRINGARRAY array of unsigned int (32bits) followed by an array of char *
DEVVAR_DOUBLESTRINGARRAY array of double followed by an array of char *
DEV_STATE TangoDevState enumeration.
CONST_DEV_STRING const char *
DEVVAR_BOOLEANARRAY array of bool
DEV_UCHAR unsigned char
DEV_LONG64 long or long long (64bits)
DEV_ULONG64 unsigned long or unsigned long long (64bits)
DEVVAR_LONG64ARRAY array of long or long long (64bits)
DEVVAR_ULONG64ARRAY array of unsigned long or unsigned long long (64bits)
DEV_INT int (32bits)
DEV_ENCODED Endoed data, description + buffer.

4.1.2.2 enum TangoDevState

The Tango Device States.

The list of all possible states for Tango devices. Every state is represented with a fixed color on the application level.

Enumerator:

- ON** The device is switched ON (green).
- OFF** The device is switched OFF (white).
- CLOSE** The device is CLOSED (white).
- OPEN** The device is OPEN (green).
- INSERT** The device is INSERTED to the beam (white).
- EXTRACT** The device is EXTRACTED from the beam (green).
- MOVING** The device is MOVING or in a state transition (blue).
- STANDBY** The device is STANDBY (yellow).
- FAULT** The device has detected a FAULT (red).
- INIT** The device is INITIALising (beige).
- RUNNING** The device is RUNNING and doing some work (blue).
- ALARM** The device indicates an ALARM (orange).
- DISABLE** The device is DISABLED by an interlock (magenta).
- UNKNOWN** The device lost its connection, the state is UNKNOWN (gray).

4.1.2.3 enum AttrQuality

The attribute quality factor.

The list of all possible attribute data quality factors. Every read attribute data has an assigned quality value to indicate the data validity.

Enumerator:

- ATTR_VALID** The attribute data is VALID.
- ATTR_INVALID** The attribute data is INVALID.
- ATTR_ALARM** The attribute indicates an ALARM on the data.
- ATTR_CHANGING** The attribute value is CHANGING and not stable.
- ATTR_WARNING** The attribute indicates an WARNING on the data.

4.1.2.4 enum AttrWriteType

The attribute write type.

The list of all possible attribute types. An attribute can be read only, write only or read/write.

Enumerator:

- READ** The attribute is read only.
- READ_WITH_WRITE** The attribute is of type read with a second write attribute associated.
- WRITE** The attribute is write only.
- READ_WRITE** The attribute is of type read/write.

4.1.2.5 enum AttrDataFormat

The attribute data format.

The data format of an attribute can be a scalar attribute, a spectrum (1D array) attribute or an image (2D array) attribute.

Enumerator:

SCALAR The attribute handles scalar values.

SPECTRUM The attribute handles a spectrum (1D array).

IMAGE The attribute handles an image (2D array).

4.1.2.6 enum DispLevel

The attribute display level.

The attribute might be displayed for expert users only.

Enumerator:

OPERATOR Attribute display all users.

EXPERT Attribute display only for expert users.

4.1.2.7 enum ErrSeverity

The error severity level.

A Tango error can indicate three different severity levels.

Enumerator:

WARN Warning level.

ERR Error level.

PANIC Real severe Panic level.

4.1.2.8 enum DevSource

The Tango data source.

Data can be read directly from the device or from the polling cache. In the case of `CACHE_DEV`, the data is read from the cache when it is available, otherwise from the device. This is the default setting.

Enumerator:

DEV Direct device reading.

CACHE Reading only from polling cache.

CACHE_DEV Reading from chache or device.

4.2 Tango Data Type Related Definitions

4.2.1 Detailed Description

Tango data type definitions for the encoded data type, array data types and for long data types. The long data types should be used to avoid the 32/64 bit problem.

Data Structures

- struct **TangoDevEncoded**
A structure containing a data description string and a pointer to the data buffer.
- struct **VarBoolArray**
A structure containing a pointer to a sequence of boolean values and the number of elements in the sequence.
- struct **VarCharArray**
A structure containing a pointer to a sequence of char values and the number of elements in the sequence.
- struct **VarShortArray**
A structure containing a pointer to a sequence of short values and the number of elements in the sequence.
- struct **VarUShortArray**
A structure containing a pointer to a sequence of unsigned short values and the number of elements in the sequence.
- struct **VarLongArray**
A structure containing a pointer to a sequence of 32 bit long values and the number of elements in the sequence.
- struct **VarULongArray**
A structure containing a pointer to a sequence of 32 bit unsigned long values and the number of elements in the sequence.
- struct **VarLong64Array**
A structure containing a pointer to a sequence of 64 bit long values and the number of elements in the sequence.
- struct **VarULong64Array**
A structure containing a pointer to a sequence of 64 bit unsigned long values and the number of elements in the sequence.
- struct **VarFloatArray**
A structure containing a pointer to a sequence of float values and the number of elements in the sequence.
- struct **VarDoubleArray**
A structure containing a pointer to a sequence of double values and the number of elements in the sequence.

- struct **VarStringArray**

A structure containing a pointer to a sequence of strings and the number of elements in the sequence.

- struct **VarStateArray**

A structure containing a pointer to a sequence of `TangoDevState` values and the number of elements in the sequence.

- struct **VarEncodedArray**

*A structure containing a pointer to a sequence of **`TangoDevEncoded`** (p. 51) values and the number of elements in the sequence.*

- struct **VarLongStringArray**

A structure containing a pointer to a sequence of long and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence.

- struct **VarDoubleStringArray**

A structure containing a pointer to a sequence of double and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence.

- struct **TangoDevEncoded**

A structure containing a data description string and a pointer to the data buffer.

- struct **VarBoolArray**

A structure containing a pointer to a sequence of boolean values and the number of elements in the sequence.

- struct **VarCharArray**

A structure containing a pointer to a sequence of char values and the number of elements in the sequence.

- struct **VarShortArray**

A structure containing a pointer to a sequence of short values and the number of elements in the sequence.

- struct **VarUShortArray**

A structure containing a pointer to a sequence of unsigned short values and the number of elements in the sequence.

- struct **VarLongArray**

A structure containing a pointer to a sequence of 32 bit long values and the number of elements in the sequence.

- struct **VarULongArray**

A structure containing a pointer to a sequence of 32 bit unsigned long values and the number of elements in the sequence.

- struct **VarLong64Array**

A structure containing a pointer to a sequence of 64 bit long values and the number of elements in the sequence.

- struct **VarULong64Array**

A structure containing a pointer to a sequence of 64 bit unsigned long values and the number of elements in the sequence.

- struct **VarFloatArray**

A structure containing a pointer to a sequence of float values and the number of elements in the sequence.

- struct **VarDoubleArray**

A structure containing a pointer to a sequence of double values and the number of elements in the sequence.

- struct **VarStringArray**

A structure containing a pointer to a sequence of strings and the number of elements in the sequence.

- struct **VarStateArray**

*A structure containing a pointer to a sequence of *TangoDevState* values and the number of elements in the sequence.*

- struct **VarEncodedArray**

*A structure containing a pointer to a sequence of *TangoDevEncoded* (p. 51) values and the number of elements in the sequence.*

- struct **VarLongStringArray**

A structure containing a pointer to a sequence of long and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence.

- struct **VarDoubleStringArray**

A structure containing a pointer to a sequence of double and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence.

Typedefs

- typedef int **TangoDevLong**

A 32 bit long value.

- typedef unsigned int **TangoDevULong**

A 32 bit unsigned long value.

- typedef long long **TangoDevLong64**

A 64 bit long value.

- typedef unsigned long long **TangoDevULong64**

A 64 bit unsigned long value.

- typedef struct **TangoDevEncoded TangoDevEncoded**

- typedef struct **VarBoolArray VarBoolArray**

- typedef struct **VarCharArray VarCharArray**

- typedef struct **VarShortArray VarShortArray**

- typedef struct **VarUShortArray** VarUShortArray
- typedef struct **VarLongArray** VarLongArray
- typedef struct **VarULongArray** VarULongArray
- typedef struct **VarLong64Array** VarLong64Array
- typedef struct **VarULong64Array** VarULong64Array
- typedef struct **VarFloatArray** VarFloatArray
- typedef struct **VarDoubleArray** VarDoubleArray
- typedef struct **VarStringArray** VarStringArray
- typedef struct **VarStateArray** VarStateArray
- typedef struct **VarEncodedArray** VarEncodedArray
- typedef struct **VarLongStringArray** VarLongStringArray
- typedef struct **VarDoubleStringArray** VarDoubleStringArray

4.3 Tango C Binding Data Structures

4.3.1 Detailed Description

Data structures used in the Tango C binding.

Data Structures

- union **TangoAttributeData**
An union of all Tango array data types used for attribute reading and writing.
- union **TangoCommandData**
An union of all Tango scalar and array data types used for command data reading and writing.
- union **TangoPropertyData**
An union of all Tango scalar and array data types used for property reading and writing.
- struct **CommandData**
A structure containing the Tango data type and the command data union to transfer command data to and from a server.
- struct **AttributeData**
A structure containing the scalar Tango data type and the attribute data union to transfer attribute data to and from a server.
- struct **AttributeDataList**
A structure containing a pointer to a sequence of attribute data structures and the number of elements in the sequence.
- struct **DevFailed**
A structure that maps all fields of the Tango::DevFailed exception.
- struct **ErrorStack**
A structure containing a pointer to a sequence of error structures and the number of elements in the sequence.
- struct **CommandInfo**
The command info structure contains descriptive command properties.
- struct **CommandInfoList**
A structure containing a pointer to a sequence of command info structures and the number of elements in the sequence.
- struct **AttributeInfo**
The attribute info structure contains descriptive attribute properties.
- struct **AttributeInfoList**
A structure containing a pointer to a sequence of attribute info structures and the number of elements in the sequence.

- struct **DbDatum**
A container structure for the Tango database access.
- struct **DbData**
*A structure containing a pointer to a sequence of **DbDatum** (p. 46) structures and the number of elements in the sequence.*
- union **TangoAttributeData**
An union of all Tango array data types used for attribute reading and writing.
- union **TangoCommandData**
An union of all Tango scalar and array data types used for command data reading and writing.
- union **TangoPropertyData**
An union of all Tango scalar and array data types used for property reading and writing.
- struct **CommandData**
A structure containing the Tango data type and the command data union to transfer command data to and from a server.
- struct **AttributeData**
A structure containing the scalar Tango data type and the attribute data union to transfer attribute data to and from a server.
- struct **AttributeDataList**
A structure containing a pointer to a sequence of attribute data structures and the number of elements in the sequence.
- struct **DevFailed**
A structure that maps all fields of the `Tango::DevFailed` exception.
- struct **ErrorStack**
A structure containing a pointer to a sequence of error structures and the number of elements in the sequence.
- struct **CommandInfo**
The command info structure contains descriptive command properties.
- struct **CommandInfoList**
A structure containing a pointer to a sequence of command info structures and the number of elements in the sequence.
- struct **AttributeInfo**
The attribute info structure contains descriptive attribute properties.
- struct **AttributeInfoList**
A structure containing a pointer to a sequence of attribute info structures and the number of elements in the sequence.
- struct **DbDatum**
A container structure for the Tango database access.

- struct **DbData**

*A structure containing a pointer to a sequence of **DbDatum** (p. 46) structures and the number of elements in the sequence.*

Typedefs

- typedef union **TangoAttributeData** **TangoAttributeData**
- typedef union **TangoCommandData** **TangoCommandData**
- typedef union **TangoPropertyData** **TangoPropertyData**
- typedef struct **CommandData** **CommandData**
- typedef struct **AttributeData** **AttributeData**
- typedef struct **AttributeDataList** **AttributeDataList**
- typedef struct **DevFailed** **DevFailed**
- typedef struct **ErrorStack** **ErrorStack**
- typedef struct **CommandInfo** **CommandInfo**
- typedef struct **CommandInfoList** **CommandInfoList**
- typedef struct **AttributeInfo** **AttributeInfo**
- typedef struct **AttributeInfoList** **AttributeInfoList**
- typedef struct **DbDatum** **DbDatum**
- typedef struct **DbData** **DbData**

4.4 Tango Proxy Related Functions

4.4.1 Detailed Description

Functions to manipulate a device connection.

Functions

- **bool tango_create_device_proxy** (char *dev_name, void **proxy, **ErrorStack** *error)
Create the access to a Tango device.
- **bool tango_delete_device_proxy** (void **proxy, **ErrorStack** *error)
Delete the access to a Tango device.
- **bool tango_set_timeout_millis** (void *proxy, int millis, **ErrorStack** *error)
Set the timeout of a device connection.
- **bool tango_get_timeout_millis** (void *proxy, int *millis, **ErrorStack** *error)
Get the timeout of a device connection.
- **bool tango_set_source** (void *proxy, **DevSource** source, **ErrorStack** *error)
Set the source for data reading.
- **bool tango_get_source** (void *proxy, **DevSource** *source, **ErrorStack** *error)
Get the actual source for data reading.
- **bool tango_lock** (void *proxy, **ErrorStack** *error)
Lock a device.
- **bool tango_unlock** (void *proxy, **ErrorStack** *error)
Unlock a device.
- **bool tango_is_locked** (void *proxy, bool *is_locked, **ErrorStack** *error)
Checks the device lock.
- **bool tango_is_locked_by_me** (void *proxy, bool *is_locked_by_me, **ErrorStack** *error)
Checks whether the device lock is already taken by the caller.
- **bool tango_locking_status** (void *proxy, char **lock_status, **ErrorStack** *error)
Get a locking status string.

4.4.2 Function Documentation

- #### 4.4.2.1 **bool tango_create_device_proxy** (char * dev_name, void ** proxy, **ErrorStack** * error)

Create the access to a Tango device.

Parameters:

- ← *dev_name* The name of the device to connect to.
- *proxy* The pointer to the device handle.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.2 bool tango_delete_device_proxy (void ** *proxy*, ErrorStack * *error*)

Delete the access to a Tango device.

Parameters:

- ← *proxy* The pointer to the device handle.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.3 bool tango_set_timeout_millis (void * *proxy*, int *millis*, ErrorStack * *error*)

Set the timeout of a device connection.

The timeout value is given in milliseconds.

Parameters:

- ← *proxy* The pointer to the device handle.
- ← *millis* The timeout value.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.4 bool tango_get_timeout_millis (void * *proxy*, int * *millis*, ErrorStack * *error*)

Get the timeout of a device connection.

The timeout value is given in milliseconds.

Parameters:

- ← *proxy* The pointer to the device handle.
- *millis* The timeout value.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.5 **bool tango_set_source (void * *proxy*, DevSource *source*, ErrorStack * *error*)**

Set the source for data reading.

Data can be read from the device or from the polling cache.

Parameters:

- ← ***proxy*** The pointer to the device handle.
- ← ***source*** The data source to use.
- ***error*** The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.6 **bool tango_get_source (void * *proxy*, DevSource * *source*, ErrorStack * *error*)**

Get the actual source for data reading.

Data can be read from the device or from the polling cache.

Parameters:

- ← ***proxy*** The pointer to the device handle.
- ***source*** The actual data source.
- ***error*** The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.7 **bool tango_lock (void * *proxy*, ErrorStack * *error*)**

Lock a device.

A locked device is protected against the following calls when executed by another client: Command_inout call, except for device state and status requested via command and for the set of commands defined as allowed following the definition of allowed command in the Tango control access schema, write_attribute call, write_read_attribute call and set_attribute_config call

Parameters:

- ← ***proxy*** The pointer to the device handle.
- ***error*** The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.8 bool tango_unlock (void * *proxy*, ErrorStack * *error*)

Unlock a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.9 bool tango_is_locked (void * *proxy*, bool * *is_locked*, ErrorStack * *error*)

Checks the device lock.

Parameters:

- ← *proxy* The pointer to the device handle.
- *is_locked* True when locked, otherwise false.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.10 bool tango_is_locked_by_me (void * *proxy*, bool * *is_locked_by_me*, ErrorStack * *error*)

Checks whether the device lock is already taken by the caller.

Parameters:

- ← *proxy* The pointer to the device handle.
- *is_locked_by_me* True when locked by the caller, otherwise false.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.4.2.11 bool tango_locking_status (void * *proxy*, char ** *lock_status*, ErrorStack * *error*)

Get a locking status string.

The status contains detailed information on the process which has taken the lock. Memory for the status string will be allocated and need to be freed.

Parameters:

- ← *proxy* The pointer to the device handle.

- *lock_status* Lock status string.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.5 Tango Command Related Functions

4.5.1 Detailed Description

Functions to query and execute Tango commands.

Functions

- **bool tango_command_query** (void *proxy, char *cmd_name, **CommandInfo** *cmd_info, **ErrorStack** *error)
Query the descriptive command properties for a given command.
- **bool tango_command_list_query** (void *proxy, **CommandInfoList** *cmd_info_list, **ErrorStack** *error)
Query the descriptive command properties for all commands of a device.
- **bool tango_command_inout** (void *proxy, char *cmd_name, **CommandData** *argin, **CommandData** *argout, **ErrorStack** *error)
Execute Tango commands with input and output parameters.
- **void tango_free_CommandData** (**CommandData** *command_data)
Free the allocated command output data.
- **void tango_free_CommandInfo** (**CommandInfo** *command_info)
Free allocated command information.
- **void tango_free_CommandInfoList** (**CommandInfoList** *command_info_list)
Free the list of all allocated command information structures.

4.5.2 Function Documentation

4.5.2.1 **bool tango_command_query** (void * *proxy*, char * *cmd_name*, **CommandInfo** * *cmd_info*, **ErrorStack** * *error*)

Query the descriptive command properties for a given command.

Parameters:

- ← **proxy** The pointer to the device handle.
- ← **cmd_name** The name of the command.
- **cmd_info** Structure for descriptive command properties.
- **error** The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.5.2.2 `bool tango_command_list_query (void * proxy, CommandInfoList * cmd_info_list, ErrorStack * error)`

Query the descriptive command properties for all commands of a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- *cmd_info_list* A sequence of structures for descriptive command properties.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.5.2.3 `bool tango_command_inout (void * proxy, char * cmd_name, CommandData * argin, CommandData * argout, ErrorStack * error)`

Ececute Tango commands with input and output parameters.

Parameters:

- ← *proxy* The pointer to the device handle.
- ← *cmd_name* The name of the command.
- ← *argin* The input parameters.
- *argout* The output parameters.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.5.2.4 `void tango_free_CommandData (CommandData * command_data)`

Free the allocated command output data.

Parameters:

- ← *command_data* The command data structure with allocated fields.

4.5.2.5 `void tango_free_CommandInfo (CommandInfo * command_info)`

Free allocated command information.

Parameters:

- ← *command_info* The command info structure with allocated fields.

4.5.2.6 `void tango_free_CommandInfoList (CommandInfoList *
command_info_list)`

Free the list of all allocated command information structures.

Parameters:

← *command_info_list* The sequence of command info structures with allocated fields.

4.6 Tango Attribute Related Functions

4.6.1 Detailed Description

Functions to query, read and write Tango attributes.

Reading of attribute properties is also possible.

Functions

- **bool tango_get_attribute_list** (void *proxy, **VarStringArray** *attr_names, **ErrorStack** *error)
Get the names off all attributes of a device.
- **bool tango_get_attribute_config** (void *proxy, **VarStringArray** *attr_names, **AttributeInfoList** *attr_info_list, **ErrorStack** *error)
Query the descriptive attribute properties for a list of attributes.
- **bool tango_attribute_list_query** (void *proxy, **AttributeInfoList** *attr_info_list, **ErrorStack** *error)
Query the descriptive attribute properties for all attributes of a device.
- **bool tango_read_attribute** (void *proxy, char *attr_name, **AttributeData** *argout, **ErrorStack** *error)
Read data from one attribute of a device.
- **bool tango_write_attribute** (void *proxy, **AttributeData** *argin, **ErrorStack** *error)
Write data to one attribute of a device.
- **bool tango_read_attributes** (void *proxy, **VarStringArray** *attr_names, **AttributeDataList** *argout, **ErrorStack** *error)
Read data from a list of attributes of a device.
- **bool tango_write_attributes** (void *proxy, **AttributeDataList** *argin, **ErrorStack** *error)
Write data to a list of attributes of a device.
- **void tango_free_AttributeData** (**AttributeData** *attribute_data)
Free the allocated attribute output data.
- **void tango_free_AttributeDataList** (**AttributeDataList** *attribute_data_list)
Free the list of allocated attribute data structures.
- **void tango_free_VarStringArray** (**VarStringArray** *string_arr)
Free the an allocated string array.
- **void tango_free_AttributeInfoList** (**AttributeInfoList** *attribute_info_list)
Free the list of all allocated attribute information structures.

4.6.2 Function Documentation

4.6.2.1 `bool tango_get_attribute_list (void * proxy, VarStringArray * attr_names, ErrorStack * error)`

Get the names off all attributes of a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- *attr_names* A string array with the attribute names.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.6.2.2 `bool tango_get_attribute_config (void * proxy, VarStringArray * attr_names, AttributeInfoList * attr_info_list, ErrorStack * error)`

Query the descriptive attribute properties for a list of attributes.

Parameters:

- ← *proxy* The pointer to the device handle.
- ← *attr_names* The string array with the attribute names.
- *attr_info_list* A sequence of structures for descriptive attribute properties.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.6.2.3 `bool tango_attribute_list_query (void * proxy, AttributeInfoList * attr_info_list, ErrorStack * error)`

Query the descriptive attribute properties for all attributes of a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- *attr_info_list* A sequence of structures for descriptive attribute properties.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.6.2.4 `bool tango_read_attribute (void * proxy, char * attr_name, AttributeData * argout, ErrorStack * error)`

Read data from one attribute of a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- ← *attr_name* The attribute name.
- *argout* The read attribute data.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.6.2.5 `bool tango_write_attribute (void * proxy, AttributeData * argin, ErrorStack * error)`

Write data to one attribute of a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- ← *argin* The attribute data to be written.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.6.2.6 `bool tango_read_attributes (void * proxy, VarStringArray * attr_names, AttributeDataList * argout, ErrorStack * error)`

Read data from a list of attributes of a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- ← *attr_names* The string array with the attribute names.
- *argout* A sequence of attribute data structures.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.6.2.7 `bool tango_write_attributes (void * proxy, AttributeDataList * argIn, ErrorStack * error)`

Write data to a list of attributes of a device.

Parameters:

- ← *proxy* The pointer to the device handle.
- ← *argIn* A sequence of attribute data structures to be written.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.6.2.8 `void tango_free_AttributeData (AttributeData * attribute_data)`

Free the allocated attribute output data.

Parameters:

- ← *attribute_data* The attribute data structure with allocated fields.

4.6.2.9 `void tango_free_AttributeDataList (AttributeDataList * attribute_data_list)`

Free the list of allocated attribute data structures.

Parameters:

- ← *attribute_data_list* The sequence of attribute data structures with allocated fields.

4.6.2.10 `void tango_free_VarStringArray (VarStringArray * string_arr)`

Free the an allocated string array.

Parameters:

- ← *string_arr* The allocated string array.

4.6.2.11 `void tango_free_AttributeInfoList (AttributeInfoList * attribute_info_list)`

Free the list of all allocated attribute information structures.

Parameters:

- ← *attribute_info_list* The sequence of attribute info structures with allocated fields.

4.7 Error Handling Related Functions

4.7.1 Detailed Description

Functions to print and free a Tango error stack.

Functions

- void **tango_print_ErrorStack** (**ErrorStack** *error_stack)
Print an error stack to stdout.
- void **tango_free_ErrorStack** (**ErrorStack** *error_stack)
Free the data allocated for an error stack in case of a failure.

4.7.2 Function Documentation

4.7.2.1 void tango_print_ErrorStack (**ErrorStack** * *error_stack*)

Print an error stack to stdout.

Parameters:

← *error_stack* The error stack of a Tango exception in case of failure.

4.7.2.2 void tango_free_ErrorStack (**ErrorStack** * *error_stack*)

Free the data allocated for an error stack in case of a failure.

Parameters:

← *error_stack* The error stack of a Tango exception in case of failure.

4.8 Tango Property and Database Related Functions

4.8.1 Detailed Description

Functions to query, read and write Tango attributes.

Reading of attribute properties is also possible.

Functions

- **bool tango_create_database_proxy** (void **db_proxy, **ErrorStack** *error)
Create the access to the Tango database.
- **bool tango_delete_database_proxy** (void **db_proxy, **ErrorStack** *error)
Delete the access to the Tango database.
- **bool tango_get_device_exported** (void *db_proxy, char *name_filter, **DbDatum** *dev_list, **ErrorStack** *error)
Get a list of exported devices using a name filter.
- **bool tango_get_device_exported_for_class** (void *db_proxy, char *class_name, **DbDatum** *dev_list, **ErrorStack** *error)
Get a list of exported devices for a given Tango class.
- **bool tango_get_object_list** (void *db_proxy, char *name_filter, **DbDatum** *obj_list, **ErrorStack** *error)
Get a list of free property objects from the Tango database using a name filter.
- **bool tango_get_object_property_list** (void *db_proxy, char *obj_name, char *name_filter, **DbDatum** *prop_list, **ErrorStack** *error)
Get a list of property names for a given free property object, using a name filter.
- **bool tango_get_property** (void *db_proxy, char *obj_name, **DbData** *prop_list, **ErrorStack** *error)
Get a list of properties for a given free property object.
- **bool tango_put_property** (void *db_proxy, char *obj_name, **DbData** *prop_list, **ErrorStack** *error)
Put a list of properties for a given free property object.
- **bool tango_delete_property** (void *db_proxy, char *obj_name, **DbData** *prop_list, **ErrorStack** *error)
Delete a list of properties for a given free property object.
- **bool tango_get_device_property** (void *dev_proxy, **DbData** *prop_list, **ErrorStack** *error)
Get a list of device properties.
- **bool tango_put_device_property** (void *dev_proxy, **DbData** *prop_list, **ErrorStack** *error)
Put a list of device properties.

- **bool tango_delete_device_property** (void *dev_proxy, DbData *prop_list, **ErrorStack** *error)
Delete a list of device properties.
- **void tango_free_DbDatum** (DbDatum *db_datum)
Free the allocated database data structure.
- **void tango_free_DbData** (DbData *db_data)
Free the list of all allocated database data structures.

4.8.2 Function Documentation

4.8.2.1 **bool tango_create_database_proxy** (void ** db_proxy, **ErrorStack** * error)

Create the access to the Tango database.

The function uses the environment variable “TANGO_HOST” to determine which instance of the TANGO database to connect to.

Parameters:

- **db_proxy** The pointer to the database handle.
- **error** The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.2 **bool tango_delete_database_proxy** (void ** db_proxy, **ErrorStack** * error)

Delete the access to the Tango database.

Parameters:

- ← **db_proxy** The pointer to the database handle.
- **error** The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.3 **bool tango_get_device_exported** (void * db_proxy, char * name_filter, DbDatum * dev_list, **ErrorStack** * error)

Get a list of exported devices using a name filter.

The name filter can contain one or more wilcards (*). Example: sr/ *-pen/ *

Parameters:

- ← **db_proxy** The pointer to the database handle.

- ← *name_filter* The filter string
- *dev_list* **DbDatum** (p. 46) structure containing a string array with the list of exported devices.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.4 `bool tango_get_device_exported_for_class (void * db_proxy, char * class_name, DbDatum * dev_list, ErrorStack * error)`

Get a list of exported devices for a given Tango class.

Parameters:

- ← *db_proxy* The pointer to the database handle.
- ← *class_name* The name of the Tango class.
- *dev_list* **DbDatum** (p. 46) structure containing a string array with the list of exported devices.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.5 `bool tango_get_object_list (void * db_proxy, char * name_filter, DbDatum * obj_list, ErrorStack * error)`

Get a list of free property objects from the Tango database using a name filter.

The name filter can contain one or more wilcards (*). Example: my*prop/ *

Parameters:

- ← *db_proxy* The pointer to the database handle.
- ← *name_filter* The filter string
- *obj_list* **DbDatum** (p. 46) structure containing a string array with the list of free property objects.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.6 `bool tango_get_object_property_list (void * db_proxy, char * obj_name, char * name_filter, DbDatum * prop_list, ErrorStack * error)`

Get a list of property names for a given free property object, using a name filter.

Parameters:

- ← *db_proxy* The pointer to the database handle.
- ← *obj_name* The name of the free property object.
- ← *name_filter* The property name filter string
- *prop_list* **DbDatum** (p. 46) structure containing a string array with the list of property names.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.7 **bool tango_get_property (void * *db_proxy*, char * *obj_name*, DbData * *prop_list*, ErrorStack * *error*)**

Get a list of properties for a given free property object.

Parameters:

- ← *db_proxy* The pointer to the database handle.
- ← *obj_name* The name of the free property object.
- *prop_list* A sequence of **DbDatum** (p. 46) structures containing the property names and the returned values.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.8 **bool tango_put_property (void * *db_proxy*, char * *obj_name*, DbData * *prop_list*, ErrorStack * *error*)**

Put a list of properties for a given free property object.

Parameters:

- ← *db_proxy* The pointer to the database handle.
- ← *obj_name* The name of the free property object.
- ← *prop_list* A sequence of **DbDatum** (p. 46) structures containing the properties to write.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.9 `bool tango_delete_property (void * db_proxy, char * obj_name, DbData * prop_list, ErrorStack * error)`

Delete a list of properties for a given free property object.

Parameters:

- ← *db_proxy* The pointer to the database handle.
- ← *obj_name* The name of the free property object.
- ← *prop_list* A sequence of **DbDatum** (p.46) structures containing the property names to delete.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.10 `bool tango_get_device_property (void * dev_proxy, DbData * prop_list, ErrorStack * error)`

Get a list of device properties.

The function uses the device handle and not the database handle.

Parameters:

- ← *dproxy* The pointer to the device handle.
- *prop_list* A sequence of **DbDatum** (p.46) structures containing the property names and the returned values.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.11 `bool tango_put_device_property (void * dev_proxy, DbData * prop_list, ErrorStack * error)`

Put a list of device properties.

The function uses the device handle and not the database handle.

Parameters:

- ← *dproxy* The pointer to the device handle.
- ← *prop_list* A sequence of **DbDatum** (p.46) structures containing the properties to write.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.12 `bool tango_delete_device_property (void * dev_proxy, DbData * prop_list, ErrorStack * error)`

Delete a list of device properties.

The function uses the device handle and not the database handle.

Parameters:

- ← *dproxy* The pointer to the device handle.
- ← *prop_list* A sequence of **DbDatum** (p.46) structures containing the property names to delete.
- *error* The error stack of a Tango exception in case of failure.

Returns:

false when an failure was detected otherwise true.

4.8.2.13 `void tango_free_DbDatum (DbDatum * db_datum)`

Free the allocated database data structure.

Parameters:

- ← *db_datum* The returned database data structure with allocated fields.

4.8.2.14 `void tango_free_DbData (DbData * db_data)`

Free the list of all allocated database data structures.

Parameters:

- ← *db_data* The sequence of returned database data structures with allocated fields.

Chapter 5

Data Structure Documentation

5.1 AttributeData Struct Reference

5.1.1 Detailed Description

A structure containing the scalar Tango data type and the attribute data union to transfer attribute data to and from a server.

The structure also contains the data dimension, the data quality and a time stamp when the data was acquired.

Data Fields

- **TangoDataType data_type**
Tango scalar data type.
- **TangoAttributeData attr_data**
Union for attribute data.
- **AttrQuality quality**
Data quality factor.
- **char * name**
Attribute name.
- **int dim_x**
Data dimension X.
- **int dim_y**
Data dimension Y.
- **struct timeval time_stamp**
Time stanp in seconds and milliseconds since epoch.

5.2 AttributeDataList Struct Reference

5.2.1 Detailed Description

A structure containing a pointer to a sequence of attribute data structures and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **AttributeData** * **sequence**

5.3 AttributeInfo Struct Reference

5.3.1 Detailed Description

The attribute info structure contains descriptive attribute properties.

Data Fields

- **char * name**
Attribute name string.
- **AttrWriteType writable**
Attribute type READ, WRITE, READ and WRITE.
- **AttrDataFormat data_format**
scalar, 1D or 2D data
- **TangoDataType data_type**
The scalar Tango data type.
- **int max_dim_x**
Maximum data size X.
- **int max_dim_y**
Maximum data size Y.
- **char * description**
Attribute description text.
- **char * label**
Attribute GUI label.
- **char * unit**
Attribute unit.
- **char * standard_unit**
Conversion factor to MKS unit.
- **char * display_unit**
Conversion factor to GUI display unit.
- **char * format**
Attribute display format (printf format).
- **char * min_value**
Min value, checked when writing to an attribute.
- **char * max_value**
Max value, checked when writing to an attribute.

- **char * min_alarm**
Min alarm value, checked during state reading.
- **char * max_alarm**
Max alarm value, checked during state reading.
- **char * writable_attr_name**
Used only for READ_ WTH_ WRITE attributes.
- **DispLevel disp_level**
operator or expert display

5.4 AttributeInfoList Struct Reference

5.4.1 Detailed Description

A structure containing a pointer to a sequence of attribute info structures and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **AttributeInfo * sequence**

5.5 CommandData Struct Reference

5.5.1 Detailed Description

A structure containing the Tango data type and the command data union to transfer command data to and from a server.

Data Fields

- **TangoDataType** `arg_type`
Tango data type.
- **TangoCommandData** `cmd_data`
Union for command data.

5.6 CommandInfo Struct Reference

5.6.1 Detailed Description

The command info structure contains descriptive command properties.

Data Fields

- **char * cmd_name**
Command name string.
- **int cmd_tag**
Command as binary value (for TACO).
- **int in_type**
in type as binary value
- **int out_type**
out type as binary value
- **char * in_type_desc**
description of in type (optional)
- **char * out_type_desc**
description of out type (optional)
- **DispLevel disp_level**
Command display level.

5.7 CommandInfoList Struct Reference

5.7.1 Detailed Description

A structure containing a pointer to a sequence of command info structures and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **CommandInfo** * **sequence**

5.8 DbData Struct Reference

5.8.1 Detailed Description

A structure containing a pointer to a sequence of **DbDatum** (p. 46) structures and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **DbDatum** * **sequence**

5.9 DbDatum Struct Reference

5.9.1 Detailed Description

A container structure for the Tango database access.

All Tango query data and property related data is passed with this structure.

Data Fields

- **char * property_name**
Name of the property.
- **TangoDataType data_type**
Tango data type.
- **TangoPropertyData prop_data**
Union for property data.
- **bool is_empty**
set when no properties available
- **bool wrong_data_type**
set when the property value cannot be converted to the given data type

5.10 DevFailed Struct Reference

5.10.1 Detailed Description

A structure that maps all fields of the Tango::DevFailed exception.

Data Fields

- **char * desc**
Error description.
- **char * reason**
Error reason.
- **char * origin**
Error origin (class and method).
- **ErrSeverity severity**
Error severity.

5.11 ErrorStack Struct Reference

5.11.1 Detailed Description

A structure containing a pointer to a sequence of error structures and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **DevFailed** * **sequence**

5.12 TangoAttributeData Union Reference

5.12.1 Detailed Description

An union of all Tango array data types used for attribute reading and writing.

Data Fields

- VarBoolArray bool_arr
- VarCharArray char_arr
- VarShortArray short_arr
- VarUShortArray ushort_arr
- VarLongArray long_arr
- VarULongArray ulong_arr
- VarLong64Array long64_arr
- VarULong64Array ulong64_arr
- VarFloatArray float_arr
- VarDoubleArray double_arr
- VarStringArray string_arr
- VarStateArray state_arr
- VarEncodedArray encoded_arr

5.13 TangoCommandData Union Reference

5.13.1 Detailed Description

An union of all Tango scalar and array data types used for command data reading and writing.

Data Fields

- bool **bool_val**
- short **short_val**
- unsigned short **ushort_val**
- int **long_val**
- unsigned int **ulong_val**
- float **float_val**
- double **double_val**
- char * **string_val**
- TangoDevState **state_val**
- TangoDevLong64 **long64_val**
- TangoDevULong64 **ulong64_val**
- VarBoolArray **bool_arr**
- VarCharArray **char_arr**
- VarShortArray **short_arr**
- VarUShortArray **ushort_arr**
- VarLongArray **long_arr**
- VarULongArray **ulong_arr**
- VarLong64Array **long64_arr**
- VarULong64Array **ulong64_arr**
- VarFloatArray **float_arr**
- VarDoubleArray **double_arr**
- VarStringArray **string_arr**
- VarStateArray **state_arr**
- TangoDevEncoded **encoded_val**
- VarLongStringArray **long_string_arr**
- VarDoubleStringArray **double_string_arr**

5.14 TangoDevEncoded Struct Reference

5.14.1 Detailed Description

A structure containing a data description string and a pointer to the data buffer.

Data Fields

- char * **encoded_format**
- unsigned int **encoded_length**
- unsigned char * **encoded_data**

5.15 TangoPropertyData Union Reference

5.15.1 Detailed Description

An union of all Tango scalar and array data types used for property reading and writing.

Data Fields

- bool **bool_val**
- unsigned char **char_val**
- short **short_val**
- unsigned short **ushort_val**
- int **long_val**
- unsigned int **ulong_val**
- float **float_val**
- double **double_val**
- char * **string_val**
- TangoDevLong64 **long64_val**
- TangoDevULong64 **ulong64_val**
- VarShortArray **short_arr**
- VarUShortArray **ushort_arr**
- VarLongArray **long_arr**
- VarULongArray **ulong_arr**
- VarLong64Array **long64_arr**
- VarULong64Array **ulong64_arr**
- VarFloatArray **float_arr**
- VarDoubleArray **double_arr**
- VarStringArray **string_arr**

5.16 VarBoolArray Struct Reference

5.16.1 Detailed Description

A structure containing a pointer to a sequence of boolean values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- bool * **sequence**

5.17 VarCharArray Struct Reference

5.17.1 Detailed Description

A structure containing a pointer to a sequence of char values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- unsigned char * **sequence**

5.18 VarDoubleArray Struct Reference

5.18.1 Detailed Description

A structure containing a pointer to a sequence of double values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- double * **sequence**

5.19 VarDoubleStringArray Struct Reference

5.19.1 Detailed Description

A structure containing a pointer to a sequence of double and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence.

Data Fields

- unsigned int **double_length**
- double * **double_sequence**
- unsigned int **string_length**
- char ** **string_sequence**

5.20 VarEncodedArray Struct Reference

5.20.1 Detailed Description

A structure containing a pointer to a sequence of **TangoDevEncoded** (p. 51) values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **TangoDevEncoded** * **sequence**

5.21 VarFloatArray Struct Reference

5.21.1 Detailed Description

A structure containing a pointer to a sequence of float values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- float * **sequence**

5.22 VarLong64Array Struct Reference

5.22.1 Detailed Description

A structure containing a pointer to a sequence of 64 bit long values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **TangoDevLong64** * **sequence**

5.23 VarLongArray Struct Reference

5.23.1 Detailed Description

A structure containing a pointer to a sequence of 32 bit long values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **TangoDevLong** * **sequence**

5.24 VarLongStringArray Struct Reference

5.24.1 Detailed Description

A structure containing a pointer to a sequence of long and the number of elements in the sequence as well as a pointer to a sequence of strings and the number of elements in the sequence.

Data Fields

- unsigned int **long_length**
- **TangoDevLong * long_sequence**
- unsigned int **string_length**
- char ** **string_sequence**

5.25 VarShortArray Struct Reference

5.25.1 Detailed Description

A structure containing a pointer to a sequence of short values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- short * **sequence**

5.26 VarStateArray Struct Reference

5.26.1 Detailed Description

A structure containing a pointer to a sequence of TangoDevState values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **TangoDevState * sequence**

5.27 VarStringArray Struct Reference

5.27.1 Detailed Description

A structure containing a pointer to a sequence of strings and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- char ** **sequence**

5.28 VarULong64Array Struct Reference

5.28.1 Detailed Description

A structure containing a pointer to a sequence of 64 bit unsigned long values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **TangoDevULong64** * **sequence**

5.29 VarULongArray Struct Reference

5.29.1 Detailed Description

A structure containing a pointer to a sequence of 32 bit unsigned long values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- **TangoDevULong * sequence**

5.30 VarUShortArray Struct Reference

5.30.1 Detailed Description

A structure containing a pointer to a sequence of unsigned short values and the number of elements in the sequence.

Data Fields

- unsigned int **length**
- unsigned short * **sequence**

Index

ALARM

Enum, 9

ATTR_ALARM

Enum, 9

ATTR_CHANGING

Enum, 9

ATTR_INVALID

Enum, 9

ATTR_VALID

Enum, 9

ATTR_WARNING

Enum, 9

AttrDataFormat

Enum, 9

Attribute

tango_attribute_list_query, 27
tango_free_AttributeData, 29
tango_free_AttributeDataList, 29
tango_free_AttributeInfoList, 29
tango_free_VarStringArray, 29
tango_get_attribute_config, 27
tango_get_attribute_list, 27
tango_read_attribute, 27
tango_read_attributes, 28
tango_write_attribute, 28
tango_write_attributes, 28

AttributeData, 37

AttributeDataList, 38

AttributeInfo, 39

AttributeInfoList, 41

AttrQuality

Enum, 9

AttrWriteType

Enum, 9

CACHE

Enum, 10

CACHE_DEV

Enum, 10

CLOSE

Enum, 9

Command

tango_command_inout, 24
tango_command_list_query, 23
tango_command_query, 23

tango_free_CommandData, 24

tango_free_CommandInfo, 24

tango_free_CommandInfoList, 24

CommandData, 42

CommandInfo, 43

CommandInfoList, 44

CONST_DEV_STRING

Enum, 8

DbData, 45

DbDatum, 46

DEV

Enum, 10

DEV_BOOLEAN

Enum, 8

DEV_DOUBLE

Enum, 8

DEV_ENCODED

Enum, 8

DEV_FLOAT

Enum, 8

DEV_INT

Enum, 8

DEV_LONG

Enum, 8

DEV_LONG64

Enum, 8

DEV_SHORT

Enum, 8

DEV_STATE

Enum, 8

DEV_STRING

Enum, 8

DEV_UCHAR

Enum, 8

DEV_ULONG

Enum, 8

DEV_ULONG64

Enum, 8

DEV_USHORT

Enum, 8

DEV_VOID

Enum, 8

DevFailed, 47

DevSource

- Enum, 10
- DEVVAR_BOOLEANARRAY
 - Enum, 8
- DEVVAR_CHARARRAY
 - Enum, 8
- DEVVAR_DOUBLEARRAY
 - Enum, 8
- DEVVAR_DOUBLESTRINGARRAY
 - Enum, 8
- DEVVAR_FLOATARRAY
 - Enum, 8
- DEVVAR_LONG64ARRAY
 - Enum, 8
- DEVVAR_LONGARRAY
 - Enum, 8
- DEVVAR_LONGSTRINGARRAY
 - Enum, 8
- DEVVAR_SHORTARRAY
 - Enum, 8
- DEVVAR_STRINGARRAY
 - Enum, 8
- DEVVAR_ULONG64ARRAY
 - Enum, 8
- DEVVAR_ULONGARRAY
 - Enum, 8
- DEVVAR_USHORTARRAY
 - Enum, 8
- DISABLE
 - Enum, 9
- DispLevel
 - Enum, 10
- Enum
 - ALARM, 9
 - ATTR_ALARM, 9
 - ATTR_CHANGING, 9
 - ATTR_INVALID, 9
 - ATTR_VALID, 9
 - ATTR_WARNING, 9
 - AttrDataFormat, 9
 - AttrQuality, 9
 - AttrWriteType, 9
 - CACHE, 10
 - CACHE_DEV, 10
 - CLOSE, 9
 - CONST_DEV_STRING, 8
 - DEV, 10
 - DEV_BOOLEAN, 8
 - DEV_DOUBLE, 8
 - DEV_ENCODED, 8
 - DEV_FLOAT, 8
 - DEV_INT, 8
 - DEV_LONG, 8
 - DEV_LONG64, 8
 - DEV_SHORT, 8
 - DEV_STATE, 8
 - DEV_STRING, 8
 - DEV_UCHAR, 8
 - DEV_ULONG, 8
 - DEV_ULONG64, 8
 - DEV_USHORT, 8
 - DEV_VOID, 8
 - DevSource, 10
 - DEVVAR_BOOLEANARRAY, 8
 - DEVVAR_CHARARRAY, 8
 - DEVVAR_DOUBLEARRAY, 8
 - DEVVAR_DOUBLESTRINGARRAY, 8
 - DEVVAR_FLOATARRAY, 8
 - DEVVAR_LONG64ARRAY, 8
 - DEVVAR_LONGARRAY, 8
 - DEVVAR_LONGSTRINGARRAY, 8
 - DEVVAR_SHORTARRAY, 8
 - DEVVAR_STRINGARRAY, 8
 - DEVVAR_ULONG64ARRAY, 8
 - DEVVAR_ULONGARRAY, 8
 - DEVVAR_USHORTARRAY, 8
 - DISABLE, 9
 - DispLevel, 10
 - ERR, 10
 - ErrSeverity, 10
 - EXPERT, 10
 - EXTRACT, 9
 - FAULT, 9
 - IMAGE, 10
 - INIT, 9
 - INSERT, 9
 - MOVING, 9
 - OFF, 9
 - ON, 9
 - OPEN, 9
 - OPERATOR, 10
 - PANIC, 10
 - READ, 9
 - READ_WITH_WRITE, 9
 - READ_WRITE, 9
 - RUNNING, 9
 - SCALAR, 10
 - SPECTRUM, 10
 - STANDBY, 9
 - TangoDataType, 8
 - TangoDevState, 8
 - UNKNOWN, 9
 - WARN, 10
 - WRITE, 9
- ERR
 - Enum, 10
- Error

- tango_free_ErrorStack, 30
- tango_print_ErrorStack, 30
- Error Handling Related Functions, 30
- ErrorStack, 48
- ErrSeverity
 - Enum, 10
- EXPERT
 - Enum, 10
- EXTRACT
 - Enum, 9
- FAULT
 - Enum, 9
- IMAGE
 - Enum, 10
- INIT
 - Enum, 9
- INSERT
 - Enum, 9
- MOVING
 - Enum, 9
- OFF
 - Enum, 9
- ON
 - Enum, 9
- OPEN
 - Enum, 9
- OPERATOR
 - Enum, 10
- PANIC
 - Enum, 10
- Property
 - tango_create_database_proxy, 32
 - tango_delete_database_proxy, 32
 - tango_delete_device_property, 35
 - tango_delete_property, 34
 - tango_free_DbData, 36
 - tango_free_DbDatum, 36
 - tango_get_device_exported, 32
 - tango_get_device_exported_for_class, 33
 - tango_get_device_property, 35
 - tango_get_object_list, 33
 - tango_get_object_property_list, 33
 - tango_get_property, 34
 - tango_put_device_property, 35
 - tango_put_property, 34
- Proxy
 - tango_create_device_proxy, 18
 - tango_delete_device_proxy, 19
 - tango_get_source, 20
 - tango_get_timeout_millis, 19
 - tango_is_locked, 21
 - tango_is_locked_by_me, 21
 - tango_lock, 20
 - tango_locking_status, 21
 - tango_set_source, 19
 - tango_set_timeout_millis, 19
 - tango_unlock, 20
- READ
 - Enum, 9
- READ_WITH_WRITE
 - Enum, 9
- READ_WRITE
 - Enum, 9
- RUNNING
 - Enum, 9
- SCALAR
 - Enum, 10
- SPECTRUM
 - Enum, 10
- STANDBY
 - Enum, 9
- Tango Attribute Related Functions, 26
- Tango C Binding Data Structures, 15
- Tango C Binding Enumerations, 7
- Tango Command Related Functions, 23
- Tango Data Type Related Definitions, 11
- Tango Property and Database Related Functions, 31
- Tango Proxy Related Functions, 18
- tango_attribute_list_query
 - Attribute, 27
- tango_command_inout
 - Command, 24
- tango_command_list_query
 - Command, 23
- tango_command_query
 - Command, 23
- tango_create_database_proxy
 - Property, 32
- tango_create_device_proxy
 - Proxy, 18
- tango_delete_database_proxy
 - Property, 32
- tango_delete_device_property
 - Property, 35
- tango_delete_device_proxy
 - Proxy, 19
- tango_delete_property
 - Property, 34
- tango_free_AttributeData

- Attribute, 29
- tango_free_AttributeDataList
 - Attribute, 29
- tango_free_AttributeInfoList
 - Attribute, 29
- tango_free_CommandData
 - Command, 24
- tango_free_CommandInfo
 - Command, 24
- tango_free_CommandInfoList
 - Command, 24
- tango_free_DbData
 - Property, 36
- tango_free_DbDatum
 - Property, 36
- tango_free_ErrorStack
 - Error, 30
- tango_free_VarStringArray
 - Attribute, 29
- tango_get_attribute_config
 - Attribute, 27
- tango_get_attribute_list
 - Attribute, 27
- tango_get_device_exported
 - Property, 32
- tango_get_device_exported_for_class
 - Property, 33
- tango_get_device_property
 - Property, 35
- tango_get_object_list
 - Property, 33
- tango_get_object_property_list
 - Property, 33
- tango_get_property
 - Property, 34
- tango_get_source
 - Proxy, 20
- tango_get_timeout_millis
 - Proxy, 19
- tango_is_locked
 - Proxy, 21
- tango_is_locked_by_me
 - Proxy, 21
- tango_lock
 - Proxy, 20
- tango_locking_status
 - Proxy, 21
- tango_print_ErrorStack
 - Error, 30
- tango_put_device_property
 - Property, 35
- tango_put_property
 - Property, 34
- tango_read_attribute

- Attribute, 27
- tango_read_attributes
 - Attribute, 28
- tango_set_source
 - Proxy, 19
- tango_set_timeout_millis
 - Proxy, 19
- tango_unlock
 - Proxy, 20
- tango_write_attribute
 - Attribute, 28
- tango_write_attributes
 - Attribute, 28
- TangoAttributeData, 49
- TangoCommandData, 50
- TangoDataType
 - Enum, 8
- TangoDevEncoded, 51
- TangoDevState
 - Enum, 8
- TangoPropertyData, 52

UNKNOWN

- Enum, 9

- VarBoolArray, 53
- VarCharArray, 54
- VarDoubleArray, 55
- VarDoubleStringArray, 56
- VarEncodedArray, 57
- VarFloatArray, 58
- VarLong64Array, 59
- VarLongArray, 60
- VarLongStringArray, 61
- VarShortArray, 62
- VarStateArray, 63
- VarStringArray, 64
- VarULong64Array, 65
- VarULongArray, 66
- VarUShortArray, 67

WARN

- Enum, 10

WRITE

- Enum, 9