TANGO-DLL

Documentation

In der Murch 15
35579 Wetzlar
Germany
Tel.: +49/6441/9116-0
www.marzhauser.com

Inhaltsverzeichnis

1. Introduction

The TANGO-DLL (programming interface for TANGO controllers) is designed to help software developers writing applications for 2/4-phase stepper motors fast and effectively without the need of hardware-oriented programming. The TANGO-DLL supports all commands of the TANGO controller.

1.1. Functional Range

- Windows DLL 32-bit and 64-bit
- Supports TANGO stepper motor controllers
- Control via RS232, or Virtual COM Port (USB, PCI and PCI-E)
- Supports most controller commands directly
- Up to 4 axes per TANGO
- Up to 8 TANGO controllers

1.2. System Requirements

The Tango-DLL can be used on all Windows PCs from Windows XP to Windows 10.
It requires the *Microsoft Visual C++ 2010 Redistributable Package,* which often is already installed on Windows PCs. If not, it can be downloaded from the www.microsoft.com website.

1.3. Supported Development Environments

The Tango-DLL is available as 32 Bit and 64 Bit version.
It has been tested on operating systems Windows XP, Windows 7, Windows 8 and Windows 10 using following development tools:

- Microsoft Visual Studio 2010 languages Visual Basic, C# and C++
- National Instruments LabVIEW
- Embarcadero Delphi 2007 and Delphi XE
- Java

- Compatibility is assumed for all other programming environments which are able to use DLL.

(DLL = Dynamic Link Library, generally means a dynamic library. In programming, a software library is a collection of program functions for tasks belonging together. Other than programs, libraries are not independently operating units, but auxiliary modules, which are made available to programs.)

2. DLL-Interface

Main part of the Tango DLL is the data file Tango_DLL.dll. Use this file for developing own programs to configure the TANGO, send commands, retrieve the status of inputs and outputs, etc.

2.1. General Information

All functions are declared with a 32-bit integer return value. A return value of 0 (zero) indicates the error free execution of the function. In case of errors (e.g. a timeout), the corresponding error code (see **Error Codes**) is returned.

The examples provided in this documentation exclusively use „LSX_" commands in which the first value stands for the TANGO ID (LSID). This ID is used to address a several controllers simultaneously. As the "LSX_" commands currently only support one controller, we recommend using the "LS_" commands. With this, the first value of the Tango-ID is not needed in function calls, neither is a CreateLSID required.

Example

„LS_"-Command:
pTango->MoveAbs(50.0, 50.0, 50.0, 10.0, TRUE);

„LSX_"-Command:
pTango->MoveAbs(**1**, 50.0, 50.0, 50.0, 10.0, TRUE);
// the first value is the LSID, which is not needed with „LS_" commands

With functions such as LSX_MoveAbs, values of 4 axes have to be passed to the function.
If the controller has only 1-3 axes, values of the not available axes are ignored; they can be set to 0.

2.2. Integration in Visual C++

An enclosure of Tango_DLL.dll has been created for Visual C++. The class CTango loads the DLL and all pointers on function calls dynamically. There is no „LS_" or „LSX_" prefix in the function names of the Tango object.
(Example pTango->Calibrate() instead of LS_Calibrate).
Only one instance should be created of the class CTango, as with Tango-DLL, momentarily, it is not possible to operate several controllers at the same time.
The required files for your C/C++ Application Tango.h and Tango.cpp can be found on the CD in the directory Software\API\Examples\Visual_C\SourceCode.

Required files:

- Tango_DLL.dll,
- Tango.h and
- Tango.cpp

Visual C++ example for controlling a Tango:

```
...
pTango = new CTango();
...

pTango->ConnectSimple(1, "COM3", 57600, TRUE);
pTango->MoveAbs(30, 50, 70, 0, TRUE);
pTango->Disconnect();
delete pTango;
```

2.3. Integration in Visual Basic

In order to use the functions of Tango-DLL, the file Tango.vb must be added to the project.
The file Tango.vb can be found on the CD in directory Software\API\Examples\Visual_Basic\SourceCode.

Required files: Tango_DLL.dll and Tango.vb

Visual Basic example for controlling a Tango:

```
Dim return value As Integer
Dim return value2 As Integer
Dim return value3 As Integer

...
Return value = LS_ConnectSimple(1, „COM3“, 57600, 1)
Return value2 = LS_MoveAbs(30, 50, 70, 0, 1)
Return value3 = LS_Disconnect
```

2.4. Integration in LabVIEW

This DLL import description can be used with every LabVIEW Version, which supports DLL import functionality.

In order to use the functions of TANGO-DLL with LabVIEW, the TANGO-DLL has to be imported to LabVIEW. Therefore follow the steps listed below:

1) Start LabVIEW
2) In LabVIEW window: Tools → Import → DLL select the first radio button and press next.

3) In the 2 corresponding fields select files "TANGO_DLL.dll" and "TANGOLSX_API.h" from CD directory / Software / API&DLL / LabVIEW.

4) “Including Paths” in the next window need not to be configured.
5) In the next window the included functions of the TANGO_DLL.dll are listed and selectable. It is recommended to select all functions. You may notice, that only half of the functions included in TANGO_DLL.dll are found in the TANGOLSX_API.h which is correct, because all functions exist in “LS_*function*” and in “LSX_*function*” notation.
6) The TANGOLSX_API.h defines just the “LSX” functions, which should be preferred to use anyway.

7) After selecting the path and name for the project library the error handling mode should at least contain a simple error handling or even an error handling with return function of TANGO_DLL.dll included.
8) The configuration of the VIs should not be changed and the import process can start.

LabVIEW starting example for controlling a TANGO:

This example creates a TANGO-ID number to select the TANGO, which is addressed for the command. A connection to the TANGO is established with virtual COM-Port 1 and Baud-Rate 57600. The actual position of all axes is read out and the TANGO is disconnected. Last step is to free the created TANGO-ID number.

Remark:
“Get” functions defined in TANGO_DLL.dll often have pointer as parameters. These pointer are displayed as inputs and outputs in LabVIEW VIs because LabVIEW is not able to detect whether this pointer is needed as input or output.

In all such "Get" VIs just connect only required output parameters. It is useless to connect input parameters because they will be ignored anyway and won’t have any effect.

Program Example:

Required LabVIEW-Version: LabVIEW 2011 and newer

An example program of controlling a TANGO via LabVIEW can be found on CD in directory Software/API&DLL/LabVIEW/TANGO_Example_LV2011. This example is implemented in LV2011 and is not compatible with elder versions. It gives an overview of how the TANGO_DLL.dll can be used with LabVIEW and how the TANGO can be controlled with a LabVIEW environment.

This example VI looks for a TANGO (connected with the PC and switched to power on) in Device Manager and writes the corresponding COM-Port in VISA-Ressourcename as a pre-selection. The default baud-rate is 57600. After selecting the correct COM-Port the user is able to connect to TANGO.
The program gives you an overview over the actual position of all active axes, the values for analog outputs and if a limit switch is active or not (limit switches can only be active, as long as no calibration and range measure drive has been performed).

Functions included in TANGO example VI:

- Calibrate (looks for the backward limit switches)
- Range Measure (looks for the forward limit switches)
- Center Drive (Drives all axes with a limit switch into its middle position → range measure is required as precondition)
- Manual Control (Move a single axis with configured step width)
- Move Absolute (Moves all active axes to an absolute position entered in destination)
- Change value of analog output 1 & 2
- Directly send commands like “?pos” or “?version” (Please be careful, here you have full access to all parameters of the controller)
- Movement demos like “Sequence” or “Meander”
- Set the actual position of all axes to zero
- Check and change “velocity” and “acceleration” of every axis
- Display the range values for limit switches (calibration and range measure is required before)

3. General Information of DLL Usage

The following flow chart shows how to establish and end Tango communication and is valid for all different physical layer like RS232, USB, PCI and PCI-E. All Tango application programs, independent of chosen and involved programming language, should follow this guideline.
DLL functions are listed and described in detail in next chapters.

3.1. Initialization of Controller

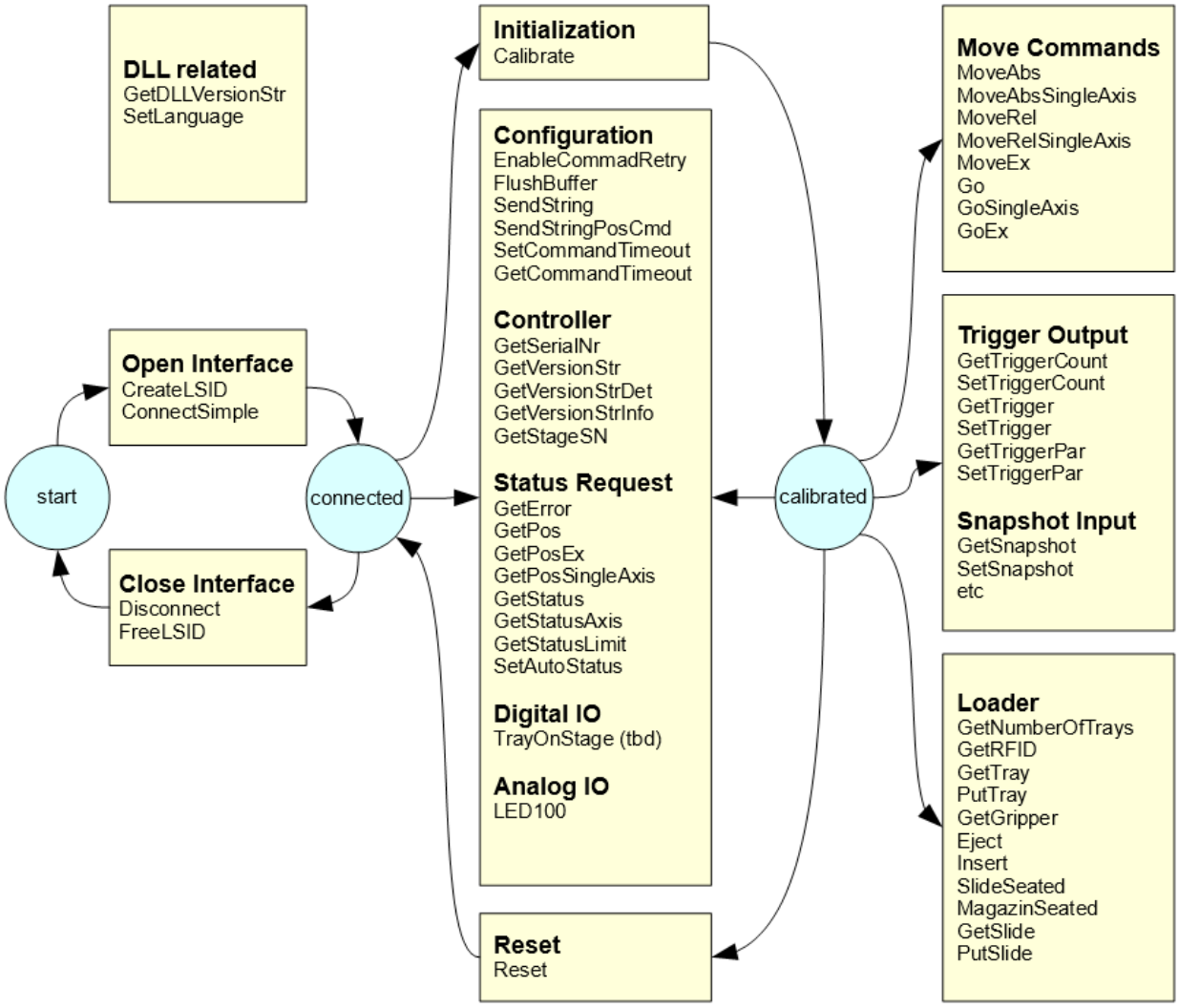
Most Märzhäuser stages are ETS coded. The Tango uses the available ETS data for stage initialization. Several parameters then are correctly predefined and write protected.
Note: Any mechanics may be damaged if wrong parameters are used. Pleas be careful to use correct stage data to prevent any damage. Follow below flow chart to transmit individual settings.

3.2. Own Program Section

In the own program section, the user can program desired functionality of the controller. This includes movements, if desired depending on status of digital I/Os as well as setting trigger signals depending on the position, etc.

3.3. API State Diagram

The API state shows which DLL functions usually require an initialisation as precondition. This means the axes must be moved at least to a reference point. Usually a limit switch is taken as reference.



4. Functions

4.1. Quick Reference

DLL Configuration / Interface:

Controller Information:

Status Requests:

Controller Settings:

Move Commands and Position Management:

Joystick and Handwheel:

Control Console with Trackball and Joyspeed Keys (Customized Application):

Limit Switches (Hardware and Software):

Digital and Analog Inputs and Outputs:

Encoder Settings:

Closed Loop Settings:

Trigger Output:

Snapshot-Input:

SlideExpress Interface:

TrayExpress Interface:

4.2. DLL Configuration / Interface

4.2.1 CreateLSID	
Description	This must always be the first command before establish a new connection. This commands the DLL to generate a unique ID to be used to establish a connection. Send this ID as 1st parameter in all subsequent commands to address one single Tango out of multiple connected Tangos. DLL provides up to 8 ID's , e.g. is able to connect up to 8 Tango controller simultaneously.
C++	int LSX_CreateLSID(int *plLSID);
Parameters	***LSID***: Contains a new Tango ID-Number after calling CreateLSID, which must be used for all subsequent commands belonging to this device.
Example	int Tango1, Tango2; pTango->CreateLSID(&Tango1); *// create ID for first Tango* pTango->CreateLSID(&Tango2); *// create ID for second Tango*

4.2.2 ConnectSimple	
Description	Connect to Tango. Hint: Use parameter ID given from command CreateLSID(). Without connection setup, connection is not possible.
C++	int LSX_ConnectSimple(int lLSID, int lAnInterfaceType, char *pcAComName, int lABaudRate, BOOL bAShowProt);
Parameters	***AnInterfaceType***: Interface type = 1 (always 1 for RS232, PCI and USB) Interface type = -1 (connects the DLL to the first USB or PCI TANGO found on the computer, without specifying a COM port) ***AComName***: Name of COM-Interface, e.g. ´´COM2´´ ***ABaudRate***: e.g. 57600 Baud (only important for RS232) ***AShowProt***: Determines, if interface protocol shall be shown
Example	pTango->ConnectSimple(1, 1, ´´COM2´´, 57600, TRUE); pTango->ConnectSimple(1, -1, NULL, 57600, TRUE); // Autoconnect with the first found USB or PCI TANGO in the system

4.2.3 Disconnect	
Description	Disconnect from Tango. After calling this function, commands can no longer be sent to the Tango Controller. This function should be called just before closing the program.
C++	int LSX_Disconnect(int lLSID);
Parameters	-
Example	pTango->Disconnect(1);

4.2.4 EnableCommandRetry	
Description	This function enables/disables repeated sending of commands in case of errors (Default enabled).
C++	int LSX_EnableCommandRetry (int lLSID, BOOL bAValue);
Parameters	***AValue***: TRUE → in case of errors Tango DLL repeats sending certain command (especially in case of WaitForAxisStop) FALSE → disable repeated sending
Example	pTango->EnableCommandRetry(1, FALSE);

4.2.5 FlushBuffer	
Description	Clear communication input buffer. Can be used in error situations to remove no longer needed feedback messages from the input buffer.
C++	int LSX_FlushBuffer (int lLSID, int lAValue);
Parameters	***AValue***: not used momentarily, can be set = 0
Example	pTango->FlushBuffer(1, 0);

4.2.6 FreeLSID	
Description	Sets a created Tango ID-Number free again. This is used as an additional parameter in Tango-DLL commands to select the Tango to which command is aimed at from a range of connected Tangos. FreeLSID should not be called before Disconnect.
C++	int LSX_FreeLSID(int lLSID);
Parameters	***LSID***: The given Tango ID-Number, which is to be set free. Do not try to use the ID after FreeLSID has been executed.
Example	int Tango1; pTango->CreateLSID(&Tango1); pTango->ConnectSimple(Tango1, …); pTango->Disconnect(Tango1); pTango-> FreeLSID(Tango1);

4.2.7 SendString

Description	Sends an ASCII string to the Tango.
C++	int LSX_SendString (int lLSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);
Parameters	***Str*** → Zero-terminated string, which is to be sent to controller. String must end with a carriage return (\r). ***Ret*** → Buffer, containing return message from Tango, in case ReadLine = TRUE or also ZERO (NULL), in case ReadLine = FALSE; ***MaxLen*** → Max. amount of characters allowed to be copied into buffer ***ReadLine*** → TRUE = read return message from Tango FALSE = don't wait for return message ***TimeOut*** → Max. waiting period for return message [ms]
Example	pTango->SendString(1, ´´?version\r´´, pcVer, 256, TRUE, 1000); *// Read version number, 1 Second Timeout* pTango->SendString(1, ´´!baud 115200\r´´, NULL, 0, FALSE, 0); *// set max. baud rate for RS232*

4.2.8 SendStringPosCmd

Description	Send move command to Tango as a string and wait for return message.
C++	int LSX_SendStringPosCmd (int lLSID, char *pcStr, char *pcRet, int lMaxLen, BOOL bReadLine, int lTimeOut);
Parameters	***Str*** → Zero-terminated ASCII string, which is to be sent to the controller ***Ret*** → Buffer, containing return message from Tango, in case ReadLine = TRUE Or also ZERO (NULL), in case ReadLine = FALSE; ***MaxLen*** → Max. amount of characters allowed copied into buffer ***ReadLine*** → TRUE = read return message from Tango FALSE = don't wait for return message ***TimeOut*** → Max. waiting period for return message [ms]
Example	pTango->SendStringPosCmd(1, ´´!moa 1 2\r´´, pcRet , 256, TRUE, 10000);

4.2.9 SetAbortFlag	
Description	Set flag so that communication with Tango is cut off. A function, which, when calling LSX_SetAbortFlag is still waiting for return message from controller (e.g. drive commands), then returns with an error message. The use of this function especially makes sense for programs with message processing routines or with multiple threads, in case, for example, a drive movement shall be stopped quickly.
C++	int LSX_SetAbortFlag (int lLSID);
Parameters	-
Example	pTango->SetAbortFlag(1); pTango->StopAxes(1); *// closes communication with Tango and sends stop command for all axes*

4.2.10 SetShowProt	
Description	Switches the interface protocol window on / off.
C++	int LSX_SetShowProt (int lLSID, BOOL bShowProt);
Parameters	***ShowProt***: TRUE = show Interface Protocol window FALSE = hide Interface Protocol window
Example	pTango->SetShowProt(1, TRUE); *// Show interface protocol for Tango1, in case not already visible*

4.2.11 GetCommandTimeout	
Description	read current DLL timeout for read, move and calibration
C++	int LSX_GetCommandTimeout (int lLSID, int *toRead, int *toMove, int *toCal);
Parameters	***toRead***: DLL standard timeout to get answer from controller (default 1000 ms) ***toMove***: DLL timeout for axes moves in [ms] ***toCal***: DLL timout for calibration in [ms]
Example	pTango->GetCommandTimeout(1, &tR, &tM, &tC);

4.2.12 SetCommandTimeout	
Description	set DLL timeout for read, move and calibration
C++	int LSX_SetCommandTimeout (int lLSID, int toRead, int toMove, int toCal);
Parameters	***toRead***: do not modify DLL standard timeout default 1000 ms ***toMove***: timeout for move in [ms] (consider speed and acceleration) ***toCal***: timout for calibration in [ms] (consider axes length , speed and acceleration)
Example	pTango->SetCommandTimeout(1, tR, tM, tC);

4.2.13 GetDLLVersionString	
Description	get DLL version string
C++	int LSX_GetDLLVesionString (int lLSID, char *pcVers, int lMaxLen);
Parameters	***pcVers*** → Buffer, containing return message from DLL ***lMaxLen*** → Max. amount of characters allowed copied into buffer
Example	pTango->GetDLLVesionString (lLSID, pcVers, lMaxLen);

4.2.14 LoadConfig	
Description	Load configuration data from certain file
C++	int LSX_LoadConfig (int lLSID, char *pcFileName);
Parameters	***pcFileName*** → file name to be used to read data from. Data must be simple ASCII text only.
Example	pTango-> LoadConfig (lLSID, pcFileName);

4.2.15 Connect	
Description	Connect using previously loaded configuration data
C++	int LSX_Connect (int lLSID);
Parameters	
Example	pTango-> LoadConfig (lLSID);

4.2.16 SaveConfig	
Description	Save configuration data to certain file
C++	int LSX_SaveConfig (int lLSID, char *pcFileName);
Parameters	***pcFileName*** → file name to be used to write data to. Data is simple ASCII text only.
Example	pTango-> SaveConfig (lLSID, pcFileName);

4.2.17 SetLanguage	
Description	Set language of protocol window
C++	int LSX_SaveConfig (int lLSID, char *pcPLN);
Parameters	***pcPLN*** → if string contains “germ” or “deut” language is switched to german if string contains “fren” or “fran” language is switched to french all other strings switch to english
Example	pTango-> SaveConfig (lLSID, pcPLN);

4.3. Controller Information

4.3.1 GetSerialNr	
Description	Reads out the Tango serial number.
C++	int LSX_GetSerialNr (int lLSID, char *pcSerialNr, int lMaxLen);
Parameters	***SerialNr***: Pointer to a buffer, in which the serial number will be returned ***MaxLen***: Max. amount of digits allowed to be copied into buffer Example value 090103001 = 09 = YY, 01 = WW, 03 = 3Axes max., 001 Index
Example	pTango->GetSerialNr(1, pcSerialNr, 256);

4.3.2 GetVersionStr	
Description	Returns current firmware version number (?ver).
C++	int LSX_GetVersionStr (int lLSID, char *pcVers, int lMaxLen);
Parameters	***Vers***: Pointer to a character buffer, in which the version number will be returned ***MaxLen***: Max. amount of characters allowed to be copied into buffer
Example	pTango->GetVersionStr(1, pcVers, 64); *// retrieve version number*

4.3.3 GetVersionStrDet	
Description	Retrieves detailed configuration of Tango (?det) as ASCII digits.
C++	int LSX_GetVersionStrDet (int lLSID, char *pcVersDet, int lMaxLen);
Parameters	***VersDet:*** Pointer to a buffer, in which the string will be returned ***MaxLen:*** Max. amount of characters allowed to be copied into buffer
Example	pTango->GetVersionStrDet(1, pcVersDet, 16); *// retrieve detailed configuration*

4.3.4 GetVersionStrInfo	
Description	Provides optional internal information on the controller version (?iver).
C++	int LSX_GetVersionStrInfo (int lLSID, char *pcVersInfo, int lMaxLen);
Parameters	***VersInfo:*** Pointer to a buffer ***MaxLen:*** Max. amount of characters to be copied into buffer
Example	pTango->GetVersionStrInfo(1, pcVersInfo, 16);

4.3.5 GetStageSN	
Description	Provides optional internal information on the stage serial number (?stagesn).
C++	int LSX_GetStageSN (int lLSID, char *pcSN, int lMaxLen);
Parameters	***pcSN:*** Pointer to a buffer ***MaxLen:*** Max. amount of characters to be copied into buffer
Example	pTango->GetVersionStrInfo(1, pcSN, 16);

4.4. Status Requests

4.4.1 GetError	
Description	Provides current error number.
C++	int LSX_GetError (int lLSID, int *plErrorCode);
Parameters	***ErrorCode***: Error number
Example	pTango->GetError(1, &ErrorCode);

4.4.2 GetPos	
Description	Retrieves current position of all axes.
C++	int LSX_GetPos (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Positions
Example	pTango->GetPos(1, &X, &Y, &Z, &A);

4.4.3 GetPosEx	
Description	Retrieves encoder or motor positions of all axes. If any axis is not available, 0.0 is returned as a value.
C++	int LSX_GetPosEx (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA, BOOL bEncoder);
Parameters	***X, Y, Z, A***: Position parameter ***Encoder*** = TRUE → Provide encoder parameters if encoder connected = FALSE → Provide motor position values
Example	pTango->GetPosEx(1, &X, &Y, &Z, &A, TRUE);

4.4.4 GetPosSingleAxis	
Description	Retrieves current position of a single axis. If axis is not available, 0.0 is returned as a value.
C++	int LSX_GetPosSingleAxis (int lLSID, int lAxis, double *pdPos);
Parameters	***Axis***: Axis of which the position parameters shall be retrieved from, X, Y, Z and A, numbered from 1 to 4 ***Pos***: Positions
Example	pTango->GetPosSingleAxis(1, 2, &Pos); *// retrieves position of Y-Axis*

4.4.5 GetStatus	
Description	Provides current status of the controller.
C++	int LSX_GetStatus (int lLSID, char *pcStat, int lMaxLen);
Parameters	***Stat***: Pointer to a buffer, in which the status string will be returned ***MaxLen***: Max. amount of characters allowed to be copied into buffer
Example	pTango->GetStatus(1, &Stat, 16);

4.4.6 GetStatusAxis	
Description	Provides current status of the axes.
C++	int LSX_GetStatusAxis (int lLSID, char *pcStatusAxisStr, int lMaxLen);
Parameters	***StatusAxisStr***: Pointer to a buffer in which status string will be returned ***MaxLen***: Max. amount of characters allowed to be copied into buffer e.g.: @ M -- J -- C -- S -- A -- D -- U T @ = Axis stands still M = Axis is in motion = Axis is not enabled J = Joystick switched on C = Axis is in closed loop A = Return message after calibration (cal) E = Error when calibrating (limit switch not cleared correctly) D = Return message after measuring stage travel range (rm) U = Setup mode T = Timeout
Example	pTango->GetStatusAxis(1, &StatusAxisStr, 16);

4.4.7 GetStatusLimit	
Description	Provides current status of software limits of each axis.
C++	int LSX_GetStatusLimit (int lLSID, char *pcLimit, int lMaxLen);
Parameters	***Limit***: Pointer to a buffer, in which the status of the axes will be returned e.g.: AA A DD LL L L A = Axis has been calibrated D = Stage travel range has been measured (rm) L = Software limit has been set = Software limit remains unchanged ***MaxLen***: Max. amount of characters allowed to be copied into the buffer
Example	pTango->GetStatusLimit(1, &Limit, 32);

4.4.8 SetAutoStatus	
Description	Switches Auto-Status on/off. Please note: As a rule, AutoStatus mode should not be changed as Tango DLL sets correct mode for travel commands etc., changing Autostatus manually to a value of 0, 2 or 3 could cause errors.
C++	int LSX_SetAutoStatus (int lLSID, int lValue);
Parameters	***Value***: AutoStatus mode: 0 → Controller sends no status 1 → Controller automatically sends ´´Position reached´´ messages 2 → Controller automatically sends ´´Position reached´´ and status messages 3 → There is only one carriage return sent for ´´Position reached´´
Example	pTango->SetAutoStatus(1, 1);

4.5. Settings

4.5.1 GetAccel	
Description	Retrieves acceleration.
C++	int LSX_GetAccelFunc (double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Acceleration values [m/s²]
Example	pTango->GetAccel(1, &X, &Y, &Z, &A);

4.5.2 SetAccel	
Description	Set acceleration.
C++	int LSX_SetAccel (int lLSID, double dX, double dY, double dZ, double dA);

Parameters	**X, Y, Z, A**: 0.01 - 20.00 [m/s²]
Example	pTango->SetAccel(1, 1.0, 1.5, 0, 0);

4.5.3 GetActiveAxes

Description	Provides the axis enable states.
C++	int LSX_GetActiveAxes (int lLSID, int *plFlags);
Parameters	***Flags***: 32-Bit Integer. After calling this function the axis bitmask is returned in Bits 0-4 Bit 0 = 1 → X-Axis cleared Bit 2 = 0 → Z-Axis not cleared
Example	pTango->GetActiveAxes(1, &Flags);

4.5.4 GetAccelFunc

Description	Retrieves acceleration function.
C++	int LSX_GetAccelFunc (int lLSID, int *lX, int *lY, int *lZ, int *lR);
Parameters	***lX, lY, lZ, lR***: Acceleration function 0 indicate trapezoidal 1 indicate sinusoidal
Example	pTango->GetAccel(1, &lX, &lY, &lZ, &lR);

4.5.5 SetAccelFunc

Description	Sets acceleration function (0 for trapezoidal, 1 for sinusoidal).
C++	int LSX_SetAccelFunc (int lLSID, int lX, int lY, int lZ, int lR);
Parameters	***lX, lY, lZ, lR***: Acceleration function 0 indicate trapezoidal 1 indicate sinusoidal
Example	pTango->SetAccel(1, lX, lY, lZ, lR);

4.5.6 SetActiveAxes

Description	Enable or disable axes.
C++	int LSX_SetActiveAxes (int lLSID, int lFlags);
Parameters	***Flags***: Bit mask, bits 0 to 4 represent axes X to A Bit 0 = 1 → X-Axis disabled Bit 2 = 0 → Z-Axis enabled
Example	pTango->SetActiveAxes(1, 3); *// X- and Y-Axis cleared (Bits 0 and 1 set),* *Z-Axis not cleared (Bit 2 = 0)*

4.5.7 GetAxisDirection

Description	Retrieves axis directions.
C++	int LSX_GetAxisDirection (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameters	***XD, YD, ZD, AD***: 4 32-Bit Integers 0 → normal rotating direction 1 → reversed rotating direction
Example	pTango->GetAxisDirection(1, &XD, &YD,&ZD,&AD);

4.5.8 SetAxisDirection

Description	Set axis directions.
C++	int LSX_SetAxisDirection (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameters	***XD, YD, ZD, AD***: 4 32-Bit Integers 0 → normal motor turning direction 1 → reverse reversed motor turning direction
Example	pTango->SetAxisDirection(1, 1, 0, 0, 0); *// reverse direction of X-Axis*

4.5.9 GetCalibBackSpeed

Description	Retrieves revolving speed at which axes are driven from limit switches when calibrating. Speed is equivalent to issued value * 0.01 rev/sec.
C++	int LSX_GetCalibBackSpeed (int lLSID, int *plSpeed);
Parameters	***Speed***: Speed value in 1/100 revolutions/second
Example	pTango->GetCalibBackSpeed(1, &lSpeed);

4.5.10	SetCalibBackSpeed
Description	Sets revolving speed at which axes are driven from limit switches when calibrating. Speed is equivalent to issued value * 0.01 rev/sec
C++	int LSX_SetCalibBackSpeed (int lLSID, int lSpeed);
Parameters	***Speed***: Speed value in 1/100 revolutions/second (within parameters of 1 to 100)
Example	pTango->SetCalibBackSpeed(1, 10); *// when calibrating, limit switches are left at 0.1 rev/sec*

4.5.11	GetCalibOffset
Description	Retrieves zero position offset of axes.
C++	int LSX_GetCalibOffset (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA)
Parameters	***X, Y, Z, A***: zero position offset from cal switch, depending on dimensions
Example	pTango->GetCalibOffset(1, &X, &Y, &Z, &A);

4.5.12	SetCalibOffset
Description	Sets zero position offset of axes. The axis zero position is moved from the hardware cal limit switch by this amount.
C++	int LSX_SetCalibOffset (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: typically 0-5 [mm]
Example	pTango->SetCalibOffset(1, 1, 1, 1, 1); *// when calibrating, axes X, Y, Z and A are each moved for 1mm (at dimension 2 2 2 2)* *from zero limit switch towards stage center and then zero position is set (software limit)*

4.5.13	GetCalibrateDir
Description	Retrieves calibrating direction.
C++	int LSX_GetCalibrateDir (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameters	***XD, YD, ZD, AD***: 32-Bit Integer 0 → normal calibration direction 1 → reversed calibration direction
Example	pTango->GetCalibrateDir(1, &XD, &YD,&ZD,&AD);

4.5.14 SetCalibrateDir	
Description	Set calibrating direction.
C++	int LSX_SetCalibrateDir (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameters	***XD, YD, ZD, AD***: 32-Bit Integer 0 → normal calibration direction 1 → reverse calibration direction
Example	pTango->(1, 1, 1, 0, 0);

4.5.15 GetCurrentDelay	
Description	Provides time delay for motor current reduction.
C++	int LSX_GetCurrentDelay (int lLSID, int *plX, int *plY, int *plZ, int *plA);
Parameters	***X, Y, Z, A***: Time delay [ms]
Example	pTango->GetCurrentDelay(1, &X, &Y,&Z,&A);

4.5.16 SetCurrentDelay	
Description	Sets the time delay, after which the motor current is reduced.
C++	int LSX_SetCurrentDelay (int lLSID, int lX, int lY, int lZ, int lA);
Parameters	***X, Y, Z, A***: 010000 [ms] (A delay of 0 disables the current reduction)
Example	pTango->SetCurrentDelay(1, 100, 300, 1000, 0);

4.5.17 GetDimensions	
Description	Provides the applied measuring units of axes
C++	int LSX_GetDimensions (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameters	***XD, YD, ZD, AD***: Dimension units 0 → Microsteps 1 → µm 2 → mm (Pre-set) 3 → Degree 4 → Revolutions 5 → cm 6 → m 7 → Inch 8 → mil (1/1000 Inch)
Example	pTango->GetDimensions(1, &XD, &YD,&ZD,&AD);

4.5.18 SetDimensions	
Description	Set measuring units of axes.
C++	int LSX_SetDimensions (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameters	***XD, YD, ZD, AD***: Dimension units 0 → Microsteps 1 → µm 2 → mm (Pre-set) 3 → Degree 4 → Revolutions 5 → cm 6 → m 7 → Inch 8 → mil (1/1000 Inch)
Example	pTango->SetDimensions(1, 3, 2, 2, 1); *// X-Axis in degree, Y- and Z-Axis in mm and A-Axis in µm*

4.5.19 GetGear	
Description	Retrieves gear ratio.
C++	int LSX_GetGear (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Gear ratio values
Example	pTango->GetGear(1, &X, &Y, &Z, &A);

4.5.20 SetGear	
Description	Set gear ratio.
C++	int LSX_SetGear (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A:*** 0.01 - 1000
Example	pTango->SetGear(1, 4.0, 2.0, 1.0, 1.0); *// programs gear ratios ¼ for Z, ½ for Y and 1/1 for Z and A*

4.5.21 GetMotorCurrent	
Description	Retrieves electrical motor current.
C++	int LSX_GetMotorCurrent (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Electrical motor currents in [A]
Example	pTango->GetMotorCurrent(1, &X, &Y, &Z, &A);

4.5.22 SetMotorCurrent	
Description	Set electrical current of motor.
C++	int LSX_SetMotorCurrent (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Motor current X, Y, Z and A-Axis in [A]
Example	pTango->SetMotorCurrent(1, 1.0, 1.0, 0.8, 0.8); *// motor current X- and Y-Axis 1 Ampere; Z- and A-Axis 0.8 Ampere*

4.5.23 GetMotorSteps	
Description	Retrieves number of motor steps.
C++	int LSX_GetMotorSteps (int lLSID, int *lX, int *lY, int *lZ, int *lA);
Parameters	***X, Y, Z, A***: Number of motor steps
Example	pTango->GetMotorSteps(1, &X, &Y, &Z, &A);

4.5.24 SetMotorSteps	
Description	Set number of motor steps. (default 200 for 1,8° stepper motors)
C++	int LSX_SetMotorSteps (int lLSID, int lX, int lY, int lZ, int lA);
Parameters	***X, Y, Z, A***: Motor steps X, Y, Z and A-Axis
Example	pTango->SetMotorCurrent(1, 200, 200, 200, 20); *// set X, Y, Z to default and A axis to 20 for special motor*

4.5.25 GetPitch	
Description	Provides spindle pitch.
C++	int LSX_GetPitch (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Spindle pitch [mm]
Example	pTango->GetPitch(1, &X, &Y, &Z, &A);

4.5.26 SetPitch	
Description	Set spindle pitch.
C++	int LSX_SetPitch (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: 0.001 - 68 [mm]
Example	pTango->SetPitch(1, 4, 4, 4, 4); *// Set spindle pitch of all axes to 4mm*

4.5.27 GetPowerAmplifier	
Description	Provides, whether amplifiers are switched on or off.
C++	int LSX_GetPowerAmplifier (int lLSID, BOOL *pbAmplifier);
Parameters	***Amplifier***: TRUE → Amplifiers are switched on FALSE → Amplifiers are switched off
Example	pTango->GetPowerAmplifier(1, &Amplifier);

4.5.28 SetPowerAmplifier	
Description	Switch amplifier on / off.
C++	int LSX_SetPowerAmplifier (int lLSID, BOOL bAmplifier);
Parameters	***Amplifier***: TRUE → Switch amplifiers on FALSE → Switch amplifiers off
Example	pTango->SetPowerAmplifier(1, TRUE); *// switches amplifiers on*

4.5.29 GetReduction	
Description	Retrieves motor current reduction factor.
C++	int LSX_GetReduction (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA)
Parameters	***X, Y, Z, A***: Electrical motor current reduction (Within parameters from 0 to 1)
Example	pTango->GetReduction(1, &X, &Y, &Z, &A);

4.5.30 SetReduction	
Description	Set reduction factor of motor current.
C++	int LSX_SetReduction (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A:*** 0 - 1.0
Example	pTango->SetReduction(1, 0.1, 0.7, 0.5, 0.5); *// standby current X-Axis = 0.1*rated current, Y-Axis = 0.7*rated current,* *Z- and A-Axis = 0,5*rated current*

4.5.31 GetRMOffset	
Description	Retrieves axis position offsets to RM limit switch.
C++	int LSX_GetRMOffset (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Limit switch position offset, depending on measuring unit (dimension).
Example	pTango->GetRMOffset(1, &X, &Y, &Z, &A);

4.5.32 SetRMOffset	
Description	Sets RM position offset of axes. The axis stops this amount before the hardware RM endswitch.
C++	int LSX_SetRMOffset (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: typically 0-5 [mm]
Example	pTango->SetRMOffset(1, 1, 1, 1, 1); *// limit positions of axes are each moved for 1mm (at dimension 2 2 2 2) towards stage center*

4.5.33 GetSpeedPoti	
Description	Shows, whether the speed potentiometer functionality is switched on or off.
C++	int LSX_GetSpeedPoti (int lLSID, BOOL *pbSpePoti);
Parameter:	The SpePoti flag shows, whether potentiometer is switched on or off
Example	pTango->(1, &flag);

4.5.34 SetSpeedPoti	
Description	Switches Speed Potentiometer functionality on or off.
C++	int LSX_SetSpeedPoti (int lLSID, BOOL bSpeedPoti);
Parameters	***SpeedPoti*** = FALSE → pre-set speed (vel) is used as movement speed = TRUE → pre-set speed (vel) can be reduced depending on the speed-potentiometer deflection
Example	pTango->SetSpeedPoti(1, TRUE); *// potentiometer is switched on*

4.5.35 GetStopAccel	
Description	Provides deceleration for error conditions.
C++	int LSX_GetStopAccel (int lLSID, double *pdXD, double *pdYD, double *pdZD, double *pdAD);
Parameters	***XD, YD, ZD, AD***: Deceleration values [m/s²]
Example	pTango->GetStopAccel(1, &XD, &YD, &ZD, &AD);

4.5.36 SetStopAccel	
Description	Deceleration value used when moving into a limit switch or causing a stop condition. If the axis acceleration (set with LSX_SetAccel) is higher, then this higher value will be used.
C++	int LSX_SetStopAccel (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Brake acceleration, within parameters 0.01 to 20 [m/s²]
Example	pTango->SetStopAccel(1, 1.5, 1.5, 1.5, 1.5);

4.5.37 GetStopPolarity	
Description	Retrieves active polarity of the stop input signal.
C++	int LSX_GetStopPolarity (int lLSID, BOOL *pbHighActiv);
Parameters	***HighActiv***: TRUE → stop input is high active FALSE → stop input is low active
Example	pTango->GetStopPolarity(1, &HighActiv);

4.5.38 SetStopPolarity	
Description	Set polarity for active stop input signal. As the stop input has a pull up resistor to 5V, ensure that switches contact to ground. A normally open contact will require a low active setting while a normally closed contact requires the high active setting.
C++	int LSX_SetStopPolarity (int lLSID, BOOL bHighActiv);
Parameters	***HighActiv***: TRUE→ stop input high active FALSE → stop input low active
Example	pTango->SetStopPolarity(1, FALSE); *// stop input is low active (e.g. normally open switch to ground)*

4.5.39 GetVel	
Description	Retrieves velocity of all axes.
C++	int LSX_GetVel (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***pdX, pdY, pdZ, pdA***: Velocity values [r/sec]
Example	pTango->GetVel(1, &X, &Y, &Z, &A);

4.5.40 SetVel	
Description	Set velocity of all axes.
C++	int LSX_SetVel (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: >0 – max. speed [r/sec]
Example	pTango->SetVel(1, 20.0, 15.0, 0.5, 10);

4.5.41 GetVelFac	
Description	Retrieves velocity reduction factor of all axes.
C++	int LSX_GetVelFac (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Velocity factor
Example	pTango->GetVelFac(1, &X, &Y, &Z, &A);

4.5.42 SetVelFac	
Description	Set velocity reduction factor.
C++	int LSX_SetVelFac (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Velocity reduction factor, within parameters 0.01 -- 1.00
Example	pTango->SetVelFac(1, 1, 1, 0.1, 0.1); *// reduces velocity of Z and A axes to 1/10 of nominal velocity*

4.5.43 LStepSave	
Description	Save current configuration in Tango (EEPROM).
C++	int LSX_LStepSave (int lLSID);
Parameters	-
Example	pTango->LStepSave(1);

4.5.44 SetAccelSingleAxis	
Description	Set acceleration of a single axis.
C++	int LSX_SetAccelSingleAxis (int lLSID, int lAxis, double dAccel);
Parameters	***Axis***: X, Y, Z, A numbered from 1 to 4 ***Accel***: Acceleration 0.01 - 20.00 [m/s²]
Example	pTango->SetAccelSingleAxis(1, 3, 1,0); *// sets acceleration of Z-Axis to 1.0 m/s²*

4.5.45 SetVelSingleAxis	
Description	Set velocity of a single axis.
C++	int LSX_SetVelSingleAxis (int lLSID, int lAxis, double dVel);
Parameters	***Axis***: X, Y, Z, A numbered from 1 to 4 ***Vel***: >0 – max. speed [r/sec]
Example	pTango->SetVelSingleAxis(1, 2, 10.0); *// sets speed of Y-Axis to 10 r/sec*

4.5.46 SoftwareReset	
Description	Software is reset to starting condition (reboot).
C++	int LSX_SoftwareReset (int lLSID);
Parameters	-
Example	pTango->SoftwareReset(1);

4.5.47	IsVel
Description	Read the actual velocities at which the axes are currently travelling. Unlike '**?vel**' or '**?speed**' this instruction returns the currently travelled (true) speed of the axes, even when controlled by a HDI device.
C++	int LSX_IsVel (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***pdX, pdY, pd Z, pdA***: actual axes velocities in [mm/s]
Example	pTango->IsVel(1, &vx, &vy, &vz, &va);

4.5.48	IsVelSingleAxis
Description	Read the actual velocity at which an axis is currently travelling. Unlike '**?vel**' or '**?speed**' this instruction returns the currently travelled (true) speed of the axes, even when controlled by a HDI device.
C++	int LSX_IsVelSingleAxis (int lLSID, int lAxis, double *pdVel);
Parameters	***lLaxis:*** *X, Y, Z, A numbered from 1 to 4* ***pdVel:*** actual axis velocity in [mm/s]
Example	pTango->IsVel(1, 2, &vel); //returns actual velocity of y axis

4.6. Move Commands and Positioning Management

4.6.1	Calibrate
Description	All enabled axes will be calibrated. Axes are driven towards smaller position values until reaching the cal limit switch and then driven with reduced speed in opposite direction until limit switch is no longer active. If a position offset is configured, the axis continues traveling for that distance. Then the zero point is set.
C++	int LSX_Calibrate (int lLSID);
Parameters	-
Example	pTango->Calibrate(1);

4.6.2	CalibrateEx
Description	Calibrates single axes. Only calibrates axes with corresponding Bit set in transferred Integer value.
C++	int LSX_CalibrateEx (int lLSID, int lFlags);
Parameters	***Flags***: Bit mask Bit 0=X, Bit 1=Y, Bit 2=Z, Bit 3=A If Bit 2 = 1 → calibrate Z-Axis

	If Bit 2 = 0 → do not calibrate Z-Axis
Example	pTango->CalibrateEx(1, 6); *// only calibrate Y- and Z-Axis (Bit 1 and 2 set)*

4.6.3 ClearPos

Description	Sets current position and internal position counter to 0. This function is needed for endless axes, as controller can only process ±1,000 motor revolutions within its parameters. This instruction will be ignored for axes with encoders.
C++	int LSX_ClearPos (int lLSID, int lFlags);
Parameters	***Flags***: Bit mask Bit 0=X, Bit 1=Y, Bit 2=Z, Bit 3=A Bit 0 = 1 → position of X-Axis is set to zero. Bit 1 = 0 → function is not executed fofor Y-Axis.

4.6.4 GetDelay	
Description	Retrieves time delay (wait time) until a commanded move is executed.
C++	int LSX_GetDelay (int lLSID, int *plDelay);
Parameters	***Delay***: Delay [ms]
Example	pTango->GetDelay(1, &Delay);

4.6.5 SetDelay	
Description	Sets the time for which move commands are delayed. Before each positioning the controller waits for this period of time delay.
C++	int LSX_SetDelay (int lLSID, int lDelay);
Parameters	***Delay***: 0 - 10000 [ms]
Example	pTango->SetDelay(1, 1000); *// 1 Second delay until a move command is executed*

4.6.6 GetDistance	
Description	Retrieve distance values last used for LSX_MoveRelShort.
C++	int LSX_GetDistance (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Current distances of all axes, depending on corresponding measuring unit.
Example	pTango->GetDistance(1, &X, &Y, &Z, &A);

4.6.7 SetDistance	
Description	Set distance. Sets distance parameters for command LSX_MoveRelShort. This enables very fast equal distance relative positioning without the need of communication overhead.
C++	int LSX_SetDistance (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Min-/max- travel range, values depend on measuring unit.
Example	pTango->SetDistance(1, 1, 2, 0, 0); *// sets distances for axes X to 1mm and Y to 2mm (if dimension=2), Z and A are not moved when calling function LSX_MoveRelShort*

4.6.8 MoveAbs	
Description	All axes are moved absolute positions. Axes X, Y, Z and A are positioned at transferred position values.
C++	int LSX_MoveAbs (int lLSID, double dX, double dY, double dZ, double dA, BOOL bWait);
Parameters	***X, Y, Z, A***: ± Travel range, command depends on measuring unit ***Wait***: Determines, whether function shall return after reaching position (= TRUE) or directly after sending the command (= FALSE)
Example	pTango->MoveAbs(1, 10.0, 10.0, -10.0, 10.0, TRUE);

4.6.9 MoveAbsSingleAxis	
Description	Positions a single axis at the transferred position.
C++	int LSX_MoveAbsSingleAxis (int lLSID, int lAxis, double dValue, BOOL bWait);
Parameters	***Axis***: X, Y, Z and A, numbered from 1 to 4 ***Value***: Position, command depends on measuring unit (dimension)
Example	pTango->MoveAbsSingleAxis(1, 2, 10.0); *// position Y-Axis absolutely at 10mm (dimension=2)*

4.6.10 MoveEx

Description	Extended move command. Function LSX_MoveEx can execute relative and absolute travel commands, synchronously as well as asynchronously. The number of axes, which are to be moved, can be determined by using AxisCount parameter. For example this function can be used to move X and Y.
C++	int LSX_MoveEx (int lLSID, double dX, double dY, double dZ, double dA, BOOL bRelative, BOOL bWait, int lAxisCount);
Parameters	***X, Y, Z, A***: Position vectors ***Relative***: When Relative = FALSE, values of X, Y, Z and A are interpreted as absolute coordinates when Relative = TRUE, they are interpreted as relative coordinates to current position ***Wait***: If Wait = TRUE is set, function doesn't return before reaching the target position, otherwise it returns immediately after sending the command to the Tango. ***AxisCount***: Number of axes, which are to be moved e.g. if AxisCount = 1, only X is moved e.g. if AxisCount = 2, X and Y are moved ...
Example	pTango->MoveEx(1, 2.0, 3.0, 0, 0, TRUE, TRUE, 2); *// X and Y are moved relatively by 2 or 3, function call returns when positions are reached*

4.6.11 MoveRel

Description	Move relative position. Axes X, Y, Z and A are moved by the transmitted distances. All axes reach their destinations simultaneously.
C++	int LSX_MoveRel (int lLSID, double dX, double dY, double dZ, double dA, BOOL bWait);
Parameters	***X, Y, Z, A***: +/- Travel range, command depends on measuring unit (dimension) ***Wait***: TRUE = function waits until position is reached FALSE = function does not wait
Example	pTango->MoveRel(1, 10.0, 10.0, -10.0, 10.0, TRUE);

4.6.12 MoveRelShort	
Description	Relative positioning (short command). This command may be used to execute several fast equal distance relative moves. Distances have to be pre-set once with LSX_SetDistance.
C++	int LSX_MoveRelShort (int lLSID);
Parameters	-
Example	pTango->SetDistance(1, 1.0, 1.0, 0, 0); for (i = 0; i < 10; i++) pTango->MoveRelShort(1); *// position X- and Y-Axis 10 times relatively by 1mm*

4.6.13 MoveRelSingleAxis	
Description	Move single axis relative.
C++	int LSX_MoveRelSingleAxis (int lLSID, int lAxis, double dValue, BOOL bWait);
Parameters	***Axis***: X, Y, Z and A numbered from 1 to 4 ***Value***: Distance, command depends on set measuring unit
Example	pTango->MoveRelSingleAxis(1, 3, 5,0); *// Z-Axis is moved by 5mm in positive direction*

4.6.14 RMeasure	
Description	Travels to maximum position of all enabled axes. Axes are driven towards larger position values until reaching rm limit switch and then driven with reduced speed in opposite direction until limit switch is no longer active. If a rm position offset is configured, the axis continues traveling for that distance. Then the max. possible travel range is set. Only to be executed when the stage features limit switches on either end. After this command the controller remembers the switch position and disables a possible security speed limitation.
C++	int LSX_RMeasure (int lLSID);
Parameters	-
Example	pTango->RMeasure(1);

4.6.15 RMeasureEx	
Description	Measure maximum position of axes (max. travel range). Moves the stage towards the RM limit switch only for the axes whose corresponding axis bit mask is set.
C++	int LSX_RMeasureEx (int lLSID, int lFlags);
Parameters	***Flags***: Bit mask Bit 2 = 1 → calibrate Z-Axis Bit 2 = 0 → Do not calibrate Z-Axis ...
Example	pTango->RMeasureEx(1, 2); *// only measure maximum position of Y-Axis*

4.6.16 SetPos	
Description	Set position.
C++	int LSX_SetPos (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Min- / max. range of travel, command depends on dimension
Example	pTango->SetPos(1, 10, 10, 0, 0); // Set current position to this values

4.6.17 StopAxes	
Description	Abort. Stops all moving axes.
C++	int LSX_StopAxes (int lLSID);
Parameters	-
Example	pTango->StopAxes(1);

4.6.18 WaitForAxisStop	
Description	Function returns as soon as the axes selected by the bit mask “lAFlags” have reached their target positions or the timeout is exceeded. LSX_WaitForAxisStop uses '?statusaxis', to poll axis status.
C++	int LSX_WaitForAxisStop (int lLSID, int lAFlags, int lATimeoutValue, BOOL *pbATimeout);
Parameters	***AFlags***: Bit mask Bit 0: X-Axis Bit 1: Y-Axis Bit 2: Z-Axis Bit 3: A-Axis ***AtimeoutValue***: Timeout in milliseconds WaitForAxisStop returns latest after this period of time pbATimeout is set to “TRUE”, if axes are still in motion. Setting lATimeoutValue = 0 disables the Timeout (wait infinite) ***pbATimeout*** Flag: Shows whether a Timeout has occurred
Example	pTango->WaitForAxisStop(1, 3, 0, flag); *// wait until X- and Y-Axes have stopped, no Timeout* pTango->WaitForAxisStop(1, 7, 10000, flag); *// wait until X-, Y- and Z-Axis has stopped, 10 sec. Timeout*

4.6.19 Go	
Description	All axes are moved to given absolute positions. You may send Go while preceding Go is in progress. This command is designed to be called directly from mouse events to move axes. Axes X, Y, Z and A are positioned at transferred position values.
C++	int LSX_Go (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: ± Travel range, command depends on measuring unit
Example	pTango->Go(1, 10.0, 10.0, -10.0, 10.0);

4.6.20 GoSingleAxis	
Description	One axes is moved to given absolute position. You may send GoSingleAxis while preceding GoSingleAxis is in progress. This command is designed to be called directly from mouse events to move axes. Addressed Axis X, Y, Z or A is positioned to transferred position.
C++	int LSX_GoSingleAxis (int lLSID, int lAxis, double dYValue);
Parameters	***X, Y, Z, A***: ± Travel range, command depends on measuring unit
Example	pTango->Go(1, 2, 12.34); //move Y to target position 12.34

4.6.21 GoEx	
Description	Similar like Go() command with additional parameter. The number of axes, which are to be moved, can be determined by using AxisCount parameter. For example this function can be used to move X and Y.
C++	int LSX_GoEx (int lLSID, double dX, double dY, double dZ, double dA, int lAxisCount);
Parameters	***X, Y, Z, A***: Position vectors ***Relative***: When Relative = FALSE, values of X, Y, Z and A are interpreted as absolute coordinates when Relative = TRUE, they are interpreted as relative coordinates to current position ***Wait***: If Wait = TRUE is set, function doesn't return before reaching the target position, otherwise it returns immediately after sending the command to the Tango. ***AxisCount***: Number of axes, which are to be moved e.g. if AxisCount = 1, only X is moved e.g. if AxisCount = 2, X and Y are moved ...
Example	pTango->GoEx(1, 2.0, 3.0, 56.78, 67.89, 2); *// X and Y are moved relatively by 2 or 3 while Z and A will not move*

4.7. Joystick and Handwheel

4.7.1 GetDigJoySpeed	
Description	Retrieves current travel speed (initiated by SetDigJoySpeed digital Joystick command).

C++	int LSX_GetDigJoySpeed (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Speed values [r/sec]
Example	pTango->GetDigJoySpeed(1, &X, &Y, &Z, &A);

4.7.2 SetDigJoySpeed

Description	This command moves axes at a constant speed. To stop the axes, a speed of 0 has to be set. Else the constant velocity is maintained until approaching a limit switch.
C++	int LSX_SetDigJoySpeed (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Speed [r/sec], within parameter range: + max. speed
Example	pTango->SetDigJoySpeed(1, 0, 10.0, 25.0, 0); *// Axes X and A - speed 0 and Joystick operation ´´OFF´´,* *Axis Y - speed 10.0 r/sec and Joystick operation ´´ON´´,* *Axis Z -speed 25.0 r/sec and Joystick operation ´´ON´´*

4.7.3 GetHandWheel

Description	Retrieves hand wheel status.
C++	int LSX_GetHandWheel (int lLSID, BOOL *pbHandWheelOn, BOOL *pbPositionCount, BOOL *pbEncoder);
Parameters	***HandWheelOn***: TRUE = hand wheel switched on FALSE = hand wheel switched off ***PositionCount***: TRUE = position count switched on FALSE = position count switched off ***Encoder***: TRUE = encoder values, if available
Example	pTango->GetHandWheel(1, &HandWheelOn, &PositionCount, &Encoder);

4.7.4 GetJoystick	
Description	Retrieves analogue Joystick status.
C++	int LSX_GetJoystick (int lLSID, BOOL *pbJoystickOn, BOOL *pbManual, BOOL *pbPositionCount, BOOL *pbEncoder);
Parameters	***JoystickOn***: TRUE = Joystick switched on ***Manual***: FALSE = Joystick switch set on automatic TRUE = Joystick is switched on manually via switch ***PositionCount***: TRUE = position count switched on ***Encoder***: TRUE = encoder values, if available
Example	pTango->GetJoystick(1, &JoystickOn, &Manual, &PositionCount, &Encoder);

4.7.5 GetJoystickDir	
Description	Retrieves axis direction for the analog Joystick and other HDI input devices.
C++	int LSX_GetJoystickDir (int lLSID, int *plXD, int *plYD, int *plZD, int *plAD);
Parameters	***XD, YD, ZD, AD:*** 0 → Axis disabled for Joystick (deflection ignored) 1 → positive axis direction, current reduction disabled -1 → negative axis direction, current reduction disabled 2 → positive axis direction with current reduction (default) -2 → negative axis direction with current reduction
Example	pTango->GetJoystickDir(1, &XD, &YD, &ZD, &AD);

4.7.6 SetJoystickDir	
Description	Sets axis direction for Joystick and other HDI input devices.
C++	int LSX_SetJoystickDir (int lLSID, int lXD, int lYD, int lZD, int lAD);
Parameters	***XD, YD, ZD, AD:*** 0 → Axis disabled for Joystick (deflection ignored) 1 → positive axis direction, current reduction disabled -1 → negative axis direction, current reduction disabled 2 → positive axis direction with current reduction (default) -2 → negative axis direction with current reduction
Example	pTango->SetJoystickDir(1, 1, 1, -1, 0); *// X- and Y-Axis positive direction, Z-Axis negative direction, A-Axis blocked*

4.7.7 GetJoystickWindow	
Description	Retrieves Joystick idle window.
C++	int LSX_GetJoystickWindow (int lLSID, int *plAValue);
Parameters	***AValue***: Analogue signal range (as digits) in which axes do not move.
Example	pTango->GetJoystickWindow(1, &AValue);

4.7.8 SetJoystickWindow	
Description	Set Joystick idle window. A value in digits which configures an angle where a analogue Joystick deflection has no effect. Used to compensate for mechanical and signal noise effects which else would cause a minor motion of the axes.
C++	int LSX_SetJoystickWindow (int lLSID, int lAValue);
Parameters	***AValue***: Analogue signal range (as digits) in which axes do not move. 0 ... 100
Example	pTango->SetJoystickWindow(1, 30);

4.7.9 SetHandWheelOff	
Description	Switch hand wheel off.
C++	int LSX_SetHandWheelOff (int lLSID);
Parameters	-
Example	pTango->SetHandWheelOff(1);

4.7.10 SetHandWheelOn	
Description	Switch hand wheel on.
C++	int LSX_SetHandWheelOn (int lLSID, BOOL bPositionCount, BOOL bEncoder);
Parameters	***PositionCount*** = TRUE → position counter on = FALSE → position counter off ***Encoder*** = TRUE → encoder values, if encoders available
Example	pTango->SetHandWheelOn(1, TRUE, TRUE); *// switch on hand wheel with position count (encoder values)*

4.7.11 SetJoystickOff	
Description	Switch analogue Joystick off.
C++	int LSX_SetJoystickOff (int lLSID);
Parameters	-
Example	pTango->SetJoystickOff(1);

4.7.12 SetJoystickOn	
Description	Switch analogue Joystick on.
C++	int LSX_SetJoystickOn (int lLSID, BOOL bPositionCount, BOOL bEncoder);
Parameters	***PositionCount*** = TRUE → position count on = FALSE → position count off ***Encoder*** = TRUE → encoder values, if encoders available
Example	pTango->SetJoystickOn(1, TRUE, TRUE); *// switch on joystick with position count (encoder values)*

4.7.13 GetHwFactor	
Description	Read hand wheel factor of all axes, in [mm per knob rotation]
C++	int LSX_GetHwFactor (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	Pointer to double
Example	pTango->GetHwFactor(1, &dX, &dY, &dZ, &dA);

4.7.14 SetHwFactor	
Description	Set hand wheel factor for all axes, in [mm per knob rotation]
C++	int LSX_SetHwFactor (int lLSID, double dX, double dY, double dZ, double dA)
Parameters	Double values
Example	pTango->SetHwFactor(1, dX, dY, dZ, dA);

4.7.15 GetHwFactorB	
Description	Read second hand wheel factor of all axes, in [mm per knob rotation]
C++	int LSX_GetHwFactorB (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	Pointer to double
Example	pTango->GetHwFactorB(1, &dX, &dY, &dZ, &dA);

4.7.16 SetHwFactorB	
Description	Set second hand wheel factor for all axes, in [mm per knob rotation]
C++	int LSX_SetHwFactorB (int lLSID, double dX, double dY, double dZ, double dA)
Parameters	Double values
Example	pTango->SetHwFactorB(1, dX, dY, dZ, dA);

4.7.17 GetZwTravel	
Description	Read z-wheel travel distances, in [mm per knob rotation]
C++	int LSX_GetZwTravel (int lLSID, int lIndex, double *pdDistance);
Parameters	lIndex: 1: Get setting for standard distance 2: Get setting for slow distance 3: Get setting for fast distance dDistance: Pointer to double
Example	pTango-> GetZwTravel (1, lIndex, &dDistance);

4.7.18 SetZwTravel	
Description	Set z-wheel travel distances, in [mm per knob rotation]
C++	int LSX_SetZwTravel (int lLSID, int lIndex, double dDistance);
Parameters	lIndex: 1: Set standard distance 2: Set slow distance 3: Set fast distance dDistance: Double value
Example	pTango-> SetZwTravel (1, lIndex, dDistance);

4.7.19 GetKey	
Description	Get HDI device key states
C++	int LSX_GetKey (int lLSID, BOOL *pbKey1, BOOL *pbKey2, BOOL *pbKey3, BOOL *pbKey4);
Parameters	Pointers to BOOL, TRUE=Key pressed
Example	pTango-> GetKey(1, &bKey[0], &bKey[1], &bKey[2], &bKey[3]);

4.7.20 GetKeyLatch	
Description	Get and clear HDI device key states
C++	int LSX_GetKeyLatch (int lLSID, BOOL *pbKey1, BOOL *pbKey2, BOOL *pbKey3, BOOL *pbKey4);
Parameters	Pointers to BOOL, TRUE=Key was or is pressed
Example	pTango-> GetKeyLatch(1, &bKey[0], &bKey[1], &bKey[2], &bKey[3]);

4.7.21 ClearKeyLatch	
Description	Clear latched key state(s)
C++	int LSX_ClearKeyLatch (int lLSID, int lKey);
Parameters	lKey: 0 = clear latched keystate of all 4 keys 1 = clear latched keystate of key 1 only 2 = clear latched keystate of key 2 only 3 = clear latched keystate of key 3 only 4 = clear latched keystate of key 4 only
Example	pTango-> ClearKeyLatch(1, 0); // Clear all

4.8. Control Console with Trackball and Joyspeed Keys

4.8.1 GetBPZ	
Description	Retrieves status of a custom-built control console with trackball.
C++	int LSX_GetBPZ (int lLSID, int *plAValue);
Parameters	***AValue***: 0 → control console is ´´OFF´´ 1 → control console active, trackball operated at 0,1µm step resolution. 2 → control console active, trackball operated with trackball factor.
Example	pTango->GetBPZ(1, &AValue);

4.8.2 SetBPZ

Description	Switches custom-built control console on / off.
C++	int LSX_SetBPZ (int lLSID, int lAValue);
Parameters	***AValue***: 0...2 0 → control console is ´´OFF´´ 1 → activate control console and operate trackball at 0,1µm step resolution. 2 → activate control console and operate trackball with trackball factor.
Example	pTango->SetBPZ(1, 1);

4.8.3 GetBPZJoyspeed

Description	Retrieves custom-built control console Joystick speed.
C++	int LSX_GetBPZJoyspeed (int lLSID, int lAPar, double *pdAValue);
Parameters	***APar***: 1, 2 or 3 (console keys for speed selection: slow, medium, fast) ***AValue***: max. speed [r/sec]
Example	pTango->GetBPZJoyspeed(1, &AValue); *// retrieve set speed of key 1 (slow)*

4.8.4 SetBPZJoyspeed	
Description	Set custom-built control console joystick speed.
C++	int LSX_SetBPZJoyspeed (int lLSID, int lAPar, double dAValue);
Parameters	***APar***: 1, 2 or 3 (console keys for speed selection: slow, medium, fast) ***AValue***: ±max. speed [r/sec]
Example	pTango->SetBPZJoyspeed(1, 1, 25); *// Set key 1 parameter (slow) to speed 25*

4.8.5 GetBPZTrackballBackLash	
Description	Retrieves custom-built control console trackball backlash.
C++	int LSX_GetBPZTrackballBackLash (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z A***: backlash [mm]
Example	pTango->GetBPZTrackballBackLash(1, &X, &Y, &Z, &A);

4.8.6 SetBPZTrackballBackLash	
Description	Set custom-built control console trackball backlash.
C++	int LSX_SetBPZTrackballBackLash (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: 0.001 to 0.15 mm
Example	pTango->SetBPZTrackballBackLash(1, 0.01, 0.01, 0.01, 0.01); // Set backlash for all axes to 10µm

4.8.7 GetBPZTrackballFactor	
Description	Retrieves control console trackball factor.
C++	int LSX_GetBPZTrackballFactor (int lLSID, double *pdAValue);
Parameters	***AValue***: Trackball factor e.g. AValue of 3 means that one trackball pulse results in 3 motor increments.
Example	pTango->GetBPZTrackballFactor(1, &AValue);

4.8.8 SetBPZTrackballFactor	
Description	Set custom-built control console trackball factor.
C++	int LSX_SetBPZTrackballFactor (int lLSID, double dAValue);
Parameters	***AValue***: 0.01 ... 100 AValue = 1 → Trackball factor = 1, i.e. one trackball impulse results in one motor increment
Example	pTango->SetBPZTrackballFactor(1, 1,0);

4.9. Limit Switches (Hardware and Software)

4.9.1 GetAutoLimitAfterCalibRM	
Description	Provides, whether internal software limits are set when calibrating (cal) or measuring stage travel range (rm).
C++	int LSX_GetAutoLimitAfterCalibRM (int lLSID, int *plFlags);
Parameters	***Flags***: Bit mask: Bit0=X, Bit1=Y, Bit2=Z, Bit3=A Bit 0 = 1 → no travel range limits are set from X-Axis calibration or range measure Bit 1 = 0 → software limits are set for Y-Axis (cal/rm)
Example	pTango->GetAutoLimitAfterCalibRM(1, &Flags);

4.9.2 SetAutoLimitAfterCalibRM	
Description	Prevents setting of internal software limits when calibrating or measuring travel range.
C++	int LSX_SetAutoLimitAfterCalibRM (int lLSID, int lFlags);
Parameters	***Flags***: Bit mask: Bit0=X, Bit1=Y, Bit2=Z, Bit3=A Bit 0 = 1 → no travel range limits are set from X-Axis calibration or range measure Bit 1 = 0 → software limits are set for Y-Axis (cal/rm)
Example	pTango->SetAutoLimitAfterCalibRM(1, Flags);

4.9.3 GetLimit	
Description	Provides soft travel range limits.
C++	int LSX_GetLimit (int lLSID, int lAxis, double *pdMinRange, double *pdMaxRange);
Parameters	***Axis***: Axis from which travel range limits are to be retrieved (X, Y, Z, A numbered from 1=X to 4=A) ***MinRange***: lower travel range limit, unit depends on dimension

	MaxRange: upper travel range limit, unit depends on dimension
Example	pTango->GetLimit(1, &MinRange, &MaxRange);

4.9.4 SetLimit	
Description	Set soft travel range limits.
C++	int LSX_SetLimit (int lLSID, int lAxis, double dMinRange, double dMaxRange);
Parameters	***Axis***: Axis from which travel range limits are to be retrieved (X, Y, Z, A numbered from 1=X to 4=A) ***MinRange***: lower travel range limit, unit depends on dimension ***MaxRange***: upper travel range limit, unit depends on dimension
Example	pTango->SetLimit(1, 1, -10.0, 20.0); *// assign X-Axis –10 as lower and 20 as upper travel range limits*

4.9.5 GetLimitControl	
Description	Retrieves, whether area control (limits) is switched on or off.
C++	int LSX_GetLimitControl (int lLSID, int lAxis, BOOL *pbActive);
Parameters	***Axis***: X, Y, Z and A, numbered from 1=X to 4=A ***Active***: TRUE = area control of corresponding axis is active FALSE = area control of corresponding axis is deactivated
Example	pTango->GetLimitControl(1, 2, &Active);

4.9.6 SetLimitControl	
Description	Switches area control on / off.
C++	int LSX_SetLimitControl (int lLSID, int lAxis, BOOL bActive);
Parameters	***Axis***: X, Y, Z and A, numbered from 1=X to 4=A ***Active***: TRUE = activate area control of corresponding axis FALSE = disable area control of corresponding axis
Example	pTango->SetLimitControl(1, 2, TRUE); *// Area control of Y-Axis is active*

4.9.7 GetSwitchActive	
Description	Provides, whether hardware limit switches are enabled.
C++	int LSX_GetSwitchActive (int lLSID, int *plXA, int *plYA, int *plZA, int *plAA);
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) The limit switch is enabled if the corresponding bit is set.
Example	pTango->GetSwitchActive(1, &XA, &YA, &ZA, &AA);

4.9.8 SetSwitchActive	
Description	Switches limit switches on / off.
C++	int LSX_SetSwitchActive (int lLSID, int lXA, int lYA, int lZA, int lAA);
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) The limit switch is enabled if the corresponding bit is set.
Example	pTango->SetSwitchActive(1, 7, 1, 5, 0); *// X-Axis: All limit switches enabled, Y-Axis: Only Zero limit switch enabled,* *// Z-Axis: E0 and EE switches enabled (default,) A-Axis: All limit switches ignored*

4.9.9 GetSwitches	
Description	Retrieves actuation status of all limit switches.
C++	int LSX_GetSwitches (int lLSID, int *plFlags);
Parameters	***Flags***: Pointer on Integer Value, which includes status of all limit switches as bit mask In bit mask, status of limit switches is encoded as follows: Limit switch EE (rm)Ref. E0 (cal) Axis AZYX AZYX AZYX Bit 0000 0000 0000 E.g.: Flags = 0x003 → E0 of X- and Y-Axis are actuated Flags = 0x200 → EE of Y-Axis is actuated
Example	pTango->GetSwitches(1, &Flags);

4.9.10 GetSwitchPolarity	
Description	Retrieves polarity of limit switches.
C++	int LSX_GetSwitchPolarity (int lLSID, int *plXP, int *plYP, int *plZP, int *plAP);
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), the corresponding switch is interpreted active when high. If bit is reset (0), the corresponding switch is active low.
Example	pTango->GetSwitchPolarity(1, &XP, &YP, &ZP, &AP);

4.9.11 SetSwitchPolarity	
Description	Sets polarity of limit switches.
C++	int LSX_SetSwitchPolarity (int lLSID, int lXP, int lYP, int lZP, int lAP);
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), the corresponding switch is interpreted active when high. If bit is reset (0), the corresponding switch is active low.
Example	pTango->SetSwitchPolarity(1, 7, 0, 0, 0); *// all limit switches of X-Axis are high active,* *all limit switches of Y-, Z- and A-Axis are low active*

4.9.12 GetSwitchType	
Description	Retrieves type of limit switches.
C++	int LSX_GetSwitchType (int lLSID, int *plXP, int *plYP, int *plZP, int *plAP);
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), input is for NPN type limit switch. If bit is reset (0), input is for for PNP type limit switch (default).
Example	pTango->GetSwitchType(1, &XP, &YP, &ZP, &RP);

4.9.13 SetSwitchType	
Description	Sets type of limit switches.
C++	int LSX_SetSwitchType (int lLSID, int lXP, int lYP, int lZP, int lAP);
Parameters	A bit mask is supplied for each axis: Bit 0 → zero limit switch (cal, “E0”) Bit 1 → reference limit switch (unused) Bit 2 → end limit switch (rm, “EE”) If bit is set (1), input is configured for NPN type limit switch using pull-up resistor. If bit is reset (0), input is configured for for PNP type limit switch with pull down resistor (default).
Example	pTango->SetSwitchType(1, XP, YP, ZP, AP);

4.10. Digital and Analog Inputs and Outputs

4.10.1 GetAnalogInput	
Description	Retrieves current A/D conversion result of an analogue channel.
C++	int LSX_GetAnalogInput (int lLSID, int lIndex, int *plValue);
Parameters	***Index***: 0...15 (analog channel), 0...9 = HDI connector, pins 1...10 10 = ANAIN0 of AUX-IO connector ***Value***: Pointer to Integer value, to which the channel’s A/D conversion result is written. 0...5V analog = 0...1023
Example	pTango->GetAnalogInput(1, 0, &Input); // Read chnannel 0

4.10.2 GetDigitalInputs	
Description	Retrieve signal level of all 16 digital input pins (I/O extension).
C++	int LSX_GetDigitalInputs (int lLSID, int *plValue);
Parameters	***Value***: Pointer to Integer value, to which the status of all inputs is written (as bit mask). LSB = Digital input 0
Example	int inputs; pTango->GetDigitalInputs(1, &inputs);

	if (Inputs & 16) ... *// if input 4 is set ...*

4.10.3 GetDigitalInputsE

Description	Retrieve signal level of additional digital inputs (16...31).
C++	int LSX_GetDigitalInputsE (int lLSID, int *plValue);
Parameters	***Value***: Pointer on a 32-Bit Integer, which returns the inputs 16...31 in the bits 0...15
Example	int ext_inputs; pTango->GetDigitalInputsE(1, &ext_inputs);

4.10.4 SetAnalogOutput

Description	Set analogue output signals.
C++	int LSX_SetAnalogOutput (int lLSID, int lIndex, int lValue);
Parameters	***Index***: 0,1 (analogue circuits) ***Value***: 0...100 [%]
Example	pTango->SetAnalogOutput(1, 0, 100); *// set analogue output 0 to max. voltage (10V)*

4.10.5 SetDigIO_Distance	
Description	**NOT SUPPORTED BY TANGO** Function of digital inputs / outputs. Activate an output depending on preset distance before or after reaching designated position.
C++	int LSX_SetDigIO_Distance (int lLSID, int lIndex, BOOL bFkt, double dDist, int lAxis);
Parameters	***Index***: 0 to 15 (output pin) ***Fkt*** = FALSE → activation of an output depending on set distance before reaching determined position ***Fkt*** = TRUE → activation of an output depending on set distance after start position ***Dist***: Distance, depends on selected dimension (unit) ***Axis***: X, Y, Z and A, numbered from 1 to 4
Example	pTango->SetDigIO_Distance(1, 7, FALSE, 78.9, 3); *// output 7 is activated 78.9mm before reaching final position (Z-Axis)*

4.10.6 SetDigIO_EmergencyStop	
Description	**NOT SUPPORTED BY TANGO** Function of digital inputs / outputs. Assignment of Emergency-Stop pin functionality.
C++	int LSX_SetDigIO_EmergencyStop (int lLSID, int lIndex);
Parameters	***Index***: 0 to 15 (input/output)
Example	pTango->SetDigIO_EmergencyStop(1, 15); *// Pin 15 is used for Emergency-Stop*

4.10.7 SetDigIO_Off	
Description	**NOT SUPPORTED BY TANGO** Switch off digital inputs / outputs function. (Does not affect inputs / outputs states).
C++	int LSX_SetDigIO_Off (int lLSID, int lIndex);
Parameters	Index: 0 to 15 (individual Input/Output pins), 16 (all 16 port pins)
Example	pTango->SetDigIO_Off(1, 0); *// Function of I/O pin 0 is switched ´Off´´*

4.10.8 SetDigIO_Polarity	
Description	Set polarity of digital inputs / outputs.
C++	int LSX_SetDigIO_Polarity (int lLSID, int lIndex, BOOL bHigh);
Parameters	***Index***: 0 to 15 (individual I/O pin), 16 (all 16 port pins) ***High*** = TRUE → high active ***High*** = FALSE → low active
Example	pTango->SetDigIO_Polarity(1, 3, TRUE); *// input pin / output pin 3 high active*

4.10.9 SetDigitalOutput	
Description	Set individual digital output pin.
C++	int LSX_SetDigitalOutput (int lLSID, int lIndex, BOOL bValue);
Parameters	***Index***: 0 to 15 ***Value***: Set pin level to FALSE = low TRUE = high
Example	pTango->SetDigitalOutput(1, 0, TRUE); *// set output pin 0 to´1´*

4.10.10 SetDigitalOutputs	
Description	Set all digital output pins (0-7) of the TANGO PCI-E or DT-E I/O1 port.
C++	int LSX_SetDigitalOutputs (int lLSID, int lValue);
Parameters	***Value***: Bit mask, bits 0-7 determine value that is set for outputs 0-7
Example	pTango->SetDigitalOutputs(1, 3); *// 3 = set outputs 0 and 1 to 1, remaining pins to 0*

4.10.11 SetDigitalOutputsE	
Description	Set digital outputs of the TANGO PCI-E or DT-E Multi I/O port.
C++	int LSX_SetDigitalOutputsE (int lLSID, int lValue);
Parameters	***Value***: Bit mask, bits 0-7 determine value that is set for outputs 0-7
Example	pTango->SetDigitalOutputsE(1, 5); *// 5 = set outputs 0 and 2 to 1, remaining pins to 0*

4.10.12 SetAuxDigitalOutput

Description	Set digital outputs of the AUX-I/O port. TANGO 3 mini: 0 = Bit 0: AUX mini Pin 6 (TAKT_OUT, default LED100 on/off pin) 1 = Bit 1: AUX mini Pin 7 (VR_OUT) 2 = Bit 2: AUX mini Pin 8 (SHUTTER_OUT) 3 = Bit 3: AUX mini Pin 9 (TRIGGER_OUT) Other TANGO controllers: 0 = Bit 0: AUX I/O Pin 5 (TAKT_OUT, default LED100 on/off pin) 1 = Bit 1: AUX I/O Pin 6 (VR_OUT) 2 = Bit 2: AUX I/O Pin 7 (SHUTTER_OUT) 3 = Bit 3: AUX I/O Pin 8 (TRIGGER_OUT)
C++	int LSX_SetAuxDigitalOutput (int lLSID, int lIndex, BOOL bValue);
Parameters	***Index***: 0 to 3 ***Value***: Set pin level to FALSE = low TRUE = high
Example	pTango->SetAuxDigitalOutput(1, 0, TRUE); *// set output 0 to high*

4.10.13 SetLedBright	
Description	Set the brightness of the LED100 illumination, when connected in the default configuration (ANOUT0 and TAKT_OUT) to the AUX I/O or AUX mini port. The SetLedBright function also controls the TAKT_OUT digital pin in order to entirely switch of LED100 with the LED-DR1 driver.
C++	int LSX_SetLedBright (int lLSID, double dBright);
Parameters	***dBright***: Brightness of the LED100 -1 = OFF A negative value <0 switches the LED entirely off (digital pin) 0 … 100 Brightness in %, up to 3 fractional digits supported
Example	pTango->SetLedBright(1, -1); *// set led off* pTango->SetLedBright(1, 0); *// set led to lowest possible brightness* pTango->SetLedBright(1, 12.345); *// set led to 12.345% brightness* pTango->SetLedBright(1, 100); *// set led to max. brightness*

4.11. Encoder Settings

4.11.1 ClearEncoder	
Description	Reset encoder positions to zero.
C++	int LSX_ClearEncoder (int lLSID, int lAxis);
Parameters	***Axis***: X, Y, Z and A, numbered from 1 to 4
Example	pTango->ClearEncoder(1, 2); *// reset encoder counter of Y-Axis to zero*

4.11.2 GetEncoder	
Description	Retrieves all encoder positions.
C++	int LSX_GetEncoder (int lLSID, double *pdXP, double *pdYP, double *pdZP, double *pdAP);
Parameters	***XP, YP, ZP, AP***: Counter values, 4x interpolated
Example	pTango->GetEncoder(1, &XP, &YP, &ZP, &AP);

4.11.3 GetEncoderActive	
Description	Retrieves which encoder will be activated after calibration. Please note: This function is corresponding to the „?encmask“ command!
C++	int LSX_GetEncoderActive (int lLSID, int *plFlags);
Parameters	***Flags***: Encoder mask (flags)

	Bit 0 = X encoder will be activated Bit 1 = Y encoder will be activated Bit 2 = Z encoder will be activated
Example	pTango->GetEncoderActive(1, &Flags);

4.11.4 SetEncoderActive

Description	Retrieves which encoder is activated after calibration Please note: This function is corresponding to „!encmask“ command.
C++	int LSX_SetEncoderActive (int lLSID, int lFlags);
Parameters	***Value***: Encoder mask (flags) Bit 0 = X encoder will be activated Bit 1 = Y encoder will be activated Bit 2 = Z encoder will be activated
Example	pTango->SetEncoderActive(1, 0); *// No encoder will be used* pTango->SetEncoderActive(1, 2); *// encoder of Y-Axis will be activated after calibration*

4.11.5 GetEncoderMask

Description	Retrieve status of encoders. Please note: This function is corresponding to „?enc“ command.
C++	LSX_GetEncoderMask (int lLSID, int *plFlags);
Parameters	***Flags***: Active encoder mask (flags) Bit 0 = X encoder is active / inactive Bit 1 = Y encoder is active / inactive Bit 2 = Z encoder is active / inactive
Example	int EncMask; pTango->GetEncoderMask(1, &EncMask); if (EncMask & 2) ... *// if encoder of Y-Axis connected + active ...*

4.11.6 SetEncoderMask

Description	Activates / deactivates encoders manually. Please note: This function is corresponding to „!enc“ command. Do not use in closed loop. Encoders should always be activated with Calibrate command.
C++	int LSX_SetEncoderMask (int lLSID, int lValue);
Parameters	***Value***: Active encoder mask (flags) Bit 0 = (activate)/deactivate X encoder Bit 1 = (activate)/deactivate Y encoder Bit 2 = (activate)/deactivate Z encoder
Example	pTango->SetEncoderMask(1, 0); *// deactivate all encoders* pTango->SetEncoderMask (1, 2); *// deactivate X and Z encoders, activate Y-Axis encoder*

4.11.7 GetEncoderPeriod	
Description	Retrieves encoder signal period length.
C++	int LSX_GetEncoderPeriod (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Period length [mm]
Example	pTango->GetEncoderPeriod(1, &X, &Y, &Z, &A);

4.11.8 SetEncoderPeriod	
Description	Set encoder signal period length.
C++	int LSX_SetEncoderPeriod (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: 0.0001 - 4 mm
Example	pTango->SetEncoderPeriod(1, 0.5, 0.5, 0.5, 0.5); *// encoder signal period of all axes is set to 0.5mm*

4.11.9 GetEncoderPosition	
Description	Retrieves position response type.
C++	int LSX_GetEncoderPosition (int lLSID, BOOL *pbValue);
Parameters	***Value***: TRUE → axis position values will be read from the encoder, if activated. Else the position will be taken from the motor position. FALSE → Position will be taken from the motor position.
Example	pTango->GetEncoderPosition(1, &Value);

4.11.10 SetEncoderPosition	
Description	Switches encoder value display on / off.
C++	int LSX_SetEncoderPosition (int lLSID, BOOL bValue);
Parameters	***Value***: TRUE → axis position values will be read from the encoder, if activated. Else the position will be taken from the motor position. FALSE → Position will be taken from the motor position.
Example	pTango->SetEncoderPosition(1, TRUE);

4.11.11 GetEncoderRefSignal	
Description	Retrieves whether the encoder reference signal is evaluated when calibrating.
C++	int LSX_GetEncoderRefSignal (int lLSID, int *plXR, int *plYR, int *plZR, int *plAR);
Parameters	1 → encoder reference signal is evaluated while calibrating 0 → reference signal is not evaluated, zero position is set at the CAL end switch
Example	pTango->GetEncoderRefSignal(1, &X, &Y, &Z, &A);

4.11.12 SetEncoderRefSignal	
Description	Evaluate reference signal from encoder when calibrating.
C++	int LSX_SetEncoderRefSignal (int lLSID, int lXR, int lYR, int lZR, int lAR);
Parameters	***XR, YR, ZR, AR***: 0 (encoder reference signal is evaluated while calibrating) or 1 (reference signal is not evaluated, zero position is set at the CAL end switch)
Example	pTango->SetEncoderRefSignal(1, 1, 1, 0, 0); *// when calibrating, reference signals of encoders X and Y are evaluated*

4.12. Closed Loop Settings

4.12.1 ClearCtrFastMoveCounter	
Description	If position difference is larger than lock-in range, a new vector will be started and corresponding counter will be increased by one.
C++	int LSX_ClearCtrFastMoveCounter (int lLSID);
Parameters	-
Example	pTango->ClearCtrFastMoveCounter(1);

4.12.2 GetController	
Description	Retrieve Closed Loop mode.
C++	int LSX_GetController (int lLSID, int *plXC, int *plYC, int *plZC, int *plRC);
Parameters	***Controller mode XC, YC, ZC, AC***: 0 → controller ´´OFF´´ 1 → controller ´´OFF after reaching target position´´ 2 → controller ´´Always ON´´ 3 → controller ´´OFF after reaching designated end position´´ with current reduction 4 → controller ´´Always ON´´ with current reduction
Example	pTango->GetController(1, &X, &Y, &Z, &A);

4.12.3 SetController	
Description	Set Closed Loop mode.
C++	int LSX_SetController (int lLSID, int lXC, int lYC, int lZC, int lAC);
Parameters	Controller mode ***XC, YC, ZC, AC***: 0 → controller ´´OFF´´ 1 → controller ´´OFF after reaching target position´´ 2 → controller ´´Always ON´´

	3 → controller ´´OFF after reaching designated end position´´ with current reduction 4 → controller ´´Always ON´´ with current reduction
Example	pTango->SetController(1, 2, 2, 0, 0); *// Enable permanent closed loop for X and Y axes*

4.12.4 GetControllerCall	
Description	Provides Closed Loop interval time.
C++	int LSX_GetControllerCall (int lLSID, int *plCtrCall);
Parameter:	***CtrCall***: Controller call time [ms]
Example	pTango->GetControllerCall(1, &CtrCall);

4.12.5 SetControllerCall	
Description	Set Closed Loop interval time.
C++	int LSX_SetControllerCall (int lLSID, int lCtrCall);
Parameters	***CtrCall***: Controller call time [ms]
Example	pTango->SetControllerCall(1, 5); *// CtrCall = 5 means: Closed Loop controller is called every 5 milliseconds*

4.12.6 GetControllerFactor	
Description	Retrieve Closed Loop controller factors.
C++	int LSX_GetControllerFactor (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Closed Loop factors
Example	pTango->GetControllerFactor(1, &X, &Y, &Z, &A);

4.12.7 SetControllerFactor	
Description	Set Closed Loop controller factor.
C++	int LSX_SetControllerFactor (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: Position difference amplification factor 1 - 64
Example	pTango->SetControllerFactor(1, 2, 2, 2, 0); *//Closed Loop amplification is set to 2 for X, Y and Z axes*

4.12.8 GetControllerSteps	
Description	Retrieves length of controller steps.
C++	int LSX_GetControllerSteps (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Length of controller steps [mm]
Example	pTango->GetControllerSteps(1, &X, &Y, &Z, &A);

4.12.9 SetControllerSteps	
Description	Set controller steps.
C++	int LSX_SetControllerSteps (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: 1 - spindle pitch (values depend on dimension)
Example	pTango->SetControllerSteps(1, 4, 5, 7, 9);

4.12.10 GetControllerTimeout	
Description	Retrieves controller timeout.
C++	Int LSX_GetControllerTimeout (int lLSID, int *plACtrTimeout);
Parameters	***ACtrTimeout***: Timeout [ms], If the Closed Loop controller is unable to settle in the target window for this time, the move is aborted (move function calls return with error code 4013).
Example	pTango->GetControllerTimeout(1, &ACtrTimeout);

4.12.11 SetControllerTimeout	
Description	Set controller timeout.
C++	int LSX_SetControllerTimeout (int lLSID, int lACtrTimeout);
Parameters	***ACtrTimeout***: Timeout 0 – 10000 ms, If the Closed Loop controller is unable to settle in the target window for this time, the move is aborted (move function calls return with error code 4013). This time should be set longer than the target window delay (TWDelay).
Example	pTango->SetControllerTimeout(1, 500); *// Abort after trying to settle in the target window for 500ms*

4.12.12 GetControllerTWDelay	
Description	Retrieve controller delay.
C++	int LSX_GetControllerTWDelay (int lLSID, int *plCtrTWDelay);
Parameters	***CtrTWDelay***: Controller delay [ms]
Example	pTango->GetControllerTWDelay(1, &CtrTWDelay);

4.12.13 SetControllerTWDelay	
Description	Set controller delay.
C++	int LSX_SetControllerTWDelay (int lLSID, int lCtrTWDelay);
Parameters	***CtrTWDelay***: Controller delay 0 - 250 ms Time for which the axis has to remain in the target window. Moves are delayed by at least this time.
Example	pTango->SetControllerTWDelay(1, 0); *// controller delay switched off, closed loop end position will be inaccurate*

4.12.14 GetCtrFastMove	
Description	Retrieves setting of FastMove function.
C++	int LSX_GetCtrFastMove (int lLSID, BOOL *pbActive);
Parameters	***Active***: TRUE → FastMove function active
Example	pTango->GetCtrFastMove(1, &Active);

4.12.15 GetCtrFastMoveCounter	
Description	If position difference is larger than lock-in range, a new vector will be started and corresponding counter will be increased by one. Function provides Fast Move counts.
C++	int LSX_GetCtrFastMoveCounter (int lLSID, int *plXC, int *plYC, int *plZC, int *plAC);
Parameters	***XC, YC, ZC, AC***: Number of carried out Fast Move functions
Example	pTango->GetCtrFastMoveCounter(1, &XC, &YC,&ZC,&AC);

4.12.16 GetTargetWindow	
Description	Retrieves closed loop target windows of all axes.
C++	int LSX_GetTargetWindow (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Target window, depends on selected dimension
Example	pTango->GetTargetWindow(1, &X, &Y, &Z, &A);

4.12.17 SetTargetWindow	
Description	Set closed loop controller target windows. The closed loop controller has to settle within ± this window size for the specified delay time.
C++	int LSX_SetTargetWindow (int lLSID, double dX, double dY, double dZ, double dA);
Parameters	***X, Y, Z, A***: 1 - 25000 (motor increments) 0.1 - 1000 (µm) 0.0001 - 1 (mm) (values depend on dimension)
Example	pTango->SetTargetWindow(1, 1.0, 0.001, 0.001, 0.0005);

4.12.18 SetCtrFastMoveOff	
Description	FastMove function deactivated.
C++	int LSX_SetCtrFastMoveOff (int lLSID);
Parameters	-
Example	pTango->SetCtrFastMoveOff(1);

4.12.19 SetCtrFastMoveOn	
Description	Activate FastMove function , meaning a new vector is started if controller position difference is larger than the lock-in range.
C++	int LSX_SetCtrFastMoveOn (int lLSID);
Parameters	-
Example	pTango->SetCtrFastMoveOn(1);

4.13. Trigger Output

4.13.1 GetTrigCount	
Description	Retrieve trigger counter value.
C++	int LSX_GetTrigCount (int lLSID, int *plValue);
Parameters	***Value***: Number of executed triggers
Example	pTango->GetTrigCount(1, &Value);

4.13.2 SetTrigCount	
Description	Set trigger counter value.
C++	int LSX_SetTrigCount (int lLSID, int lValue);
Parameters	***Value***: 0 to 2147483647
Example	pTango->SetTrigCount(1, 0);

4.13.3 GetTrigger	
Description	Retrieve trigger setting.
C++	int LSX_GetTrigger (int lLSID, BOOL *pbATrigger);
Parameters	***ATrigger:*** TRUE → trigger is ´´On´´ FALSE → trigger is ´´Off´´
Example	pTango->GetTrigger(1, &ATrigger);

4.13.4 SetTrigger	
Description	Switch trigger on / off.
C++	int LSX_SetTrigger (int lLSID, BOOL bATrigger);
Parameters	***ATrigger*** = TRUE → switch trigger on = FALSE → switch trigger off
Example	pTango->SetTrigger(1, TRUE);

4.13.5 GetTriggerPar	
Description	Retrieves trigger parameters.
C++	int LSX_GetTriggerPar (int lLSID, int *plAxis, int *plMode, int *plSignal, double *pdDistance);
Parameters	***Axis***: Axis 1...4 ***Mode***: Trigger mode (see command !trigm) ***Signal***: Trigger signal (see command !trigs) ***Distance***: Trigger distance (see command !trigd)
Example	pTango->GetTriggerPar(1, &Axis, &Mode, & Signal, &Distance);

4.13.6 SetTriggerPar	
Description	Set trigger parameters.
C++	int LSX_SetTriggerPar (int lLSID, int lAxis, int lMode, int lSignal, double dDistance);
Parameters	***Axis***: Axis 1...4 ***Mode***: Trigger mode (see command !trigm) ***Signal***: Trigger signal (see command !trigs) ***Distance***: Trigger distance (see command !trigd)
Example	pTango->SetTriggerPar(1, 1, 3, 2, 5.0);

4.14. Snapshot Input

4.14.1 GetSnapshot	
Description	Provides current Snapshot state, if it is ON (enabled) or OFF (disabled).
C++	int LSX_GetSnapshot (int lLSID, BOOL *pbASnapshot);
Parameters	***ASnapshot:*** TRUE → Snapshot is "On" (enabled) FALSE → Snapshot is "Off" (disabled)
Example	pTango->GetSnapshot(1, &ASnapshot);

4.14.2 SetSnapshot	
Description	Switch Snapshot functionality ON or OFF.
C++	int LSX_SetSnapshot (int lLSID, BOOL bASnapshot);

Parameters	***ASnapshot***: TRUE → switch Snapshot "On" (enable) FALSE → switch Snapshot "Off" (disable)
Example	pTango->SetSnapshot(1, TRUE); // Globally enable the snapshot functionality

4.14.3 GetSnapshotMode

Description	Provides the current Snapshot mode.
C++	int LSX_GetSnapshotMode (int lLSID, int*plMode);
Parameters	***Mode:*** 0-11 (refer to snsm documentation in TANGO Instruction Set)
Example	pTango->GetSnapshotMode(1, &Mode);

4.14.4 SetSnapshotMode

Description	Sets the Snapshot mode (functionality).
C++	int LSX_SetSnapshotMode (int lLSID, int lMode);
Parameters	***Mode***: 0-11 (refer to snsm documentation in TANGO Instruction Set)
Example	pTango->SetSnapshotMode(1, 0); // Set mode to 0 = capture positions @ HDI F2 key

4.14.5 GetSnapshotCount

Description	Snapshot counter. It counts the snapshot events = number of captured positions / entries in the position array (see SnapshotPosArray).
C++	int LSX_GetSnapshotCount (int lLSID, int *plSnsCount);
Parameters	***SnsCount***: Amount of captured Snapshots (= available position array entries)
Example	pTango->GetSnapshotCount(1, &SnsCount);

4.14.6 SetSnapshotCount

Description	Manipulate Snapshot counter (captured positions), truncate position array entries.
C++	int LSX_SetSnapshotCount (int lLSID, int lSnsCount);
Parameters	***SnsCount***: Amount of available position array entries
Example	pTango->SetSnapshotCount(1, 5); // Truncate position array to 5 entries.

4.14.7 GetSnapshotFilter

Description	Retrieve input filter times for signal chatter.
C++	int LSX_GetSnapshotFilter (int lLSID, int *plTime);
Parameters	***Time***: Filter time [ms]
Example	pTango->GetSnapshotFilter(1, &Time);

4.14.8 SetSnapshotFilter	
Description	Set input filter when switches chatter.
C++	int LSX_SetSnapshotFilter (int lLSID, int lTime);
Parameters	***Time***: Filter time, within 0-100 ms
Example	pTango->SetSnapshotFilter(1, 0); *// no filter, fast response (e.g. for TTL signals)*

4.14.9 GetSnapshotPar	
Description	Retrieve Snapshot parameters.
C++	int LSX_GetSnapshotPar (int lLSID, BOOL *pbHigh, BOOL *pbAutoMode);
Parameters	***High***: TRUE → snapshot is high active FALSE → snapshot is low active ***AutoMode***: TRUE → snapshot ´´Automatic´´: Position is automatically moved to after first snapshot pulse (corresponds to SnapshotMode 1) FALSE → snapshot capture mode (corresponds to SnapshotMode 0)
Example	pTango->GetSnapshotPar(1, &High, &AutoMode);

4.14.10 SetSnapshotPar

Description	Set Snapshot parameters 8polarity and mode 0 or 1). The AutoMode might interfere with a previously set SnapshotMode, if that was set to a mode higher than 1).
C++	int LSX_SetSnapshotPar (int lLSID, BOOL bHigh, BOOL bAutoMode);
Parameters	***High***: TRUE → snapshot is high active FALSE → snapshot is low active ***AutoMode***: TRUE → snapshot ´´Automatic´´: Position is automatically moved to after first snapshot pulse (corresponds to SnapshotMode 1) FALSE → snapshot capture mode (corresponds to SnapshotMode 0)
Example	pTango->SetSnapshotPar(1, TRUE, FALSE);

4.14.11 GetSnapshotPos

Description	Retrieve position that was captured on the Snapshot event.
C++	int LSX_GetSnapshotPos (int lLSID, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***X, Y, Z, A***: Position values
Example	pTango->GetSnapshotPos(1, &X, &Y, &Z, &A);

4.14.12 GetSnapshotPosArray

Description	Retrieve Snapshot position from Array.
C++	int LSX_GetSnapshotPosArray (int lLSID, int lIndex, double *pdX, double *pdY, double *pdZ, double *pdA);
Parameters	***Index***: Index of snapshot positions (from =1 to SnapshotCount, max. entries is 1024) ***X, Y, Z, A***: Position values
Example	pTango->GetSnapshotPosArray(1, 2, &X, &Y, &Z, &A); *// 2 = Read positions captured on the second snapshot event (second array entry)*

4.14.13 SetSnapshotPosArray	
Description	Set, append or change entries of the position array.
C++	int LSX_SetSnapshotPosArray (int lLSID, int lIndex, double dX, double dY, double dZ, double dA);
Parameters	***Index***: Index of snapshot positions (1-1024) Index must be within the number of existing entries (or one above to append) appending is also possible by using Index = -1, which is easier to handle ***X, Y, Z, A***: Position values
Example	pTango->SetSnapshotPosArray(1, -1, 0.55, 2.4, 0.0, 0.0); *// Append a position array entry by software*

4.14.14 ClearSnapshotPosArray	
Description	Deletes the entire position array (clear all entries).
C++	int LSX_ClearSnapshotPosArray (int lLSID,);
Parameters	-
Example	pTango->ClearSnapshotPosArray(1); *// Delete the entire PosArray*

4.14.15 GetSnapshotIndex	
Description	Retrieve the current Snapshot index, e.g. to identify where it is in "Automatic" mode. Remarks: The index goes from 0 to SnapshotCount-1, so index "0" is PosArray(1).
C++	int LSX_GetSnapshotIndex (int lLSID, int *plSnsIndex);
Parameters	***SnsIndex***: Current position of the index pointer within the position array
Example	pTango->GetSnapshotIndex(1, &SnsIndex);

4.14.16 SetSnapshotIndex	
Description	Manipulate Snapshot index (set index to a different position array entry) Remarks: The index goes from 0 to SnapshotCount-1, so index "0" is PosArray(1).
C++	int LSX_SetSnapshotIndex (int lLSID, int lSnsIndex);
Parameters	***SnsIndex***: Required position of the index pointer within the PosArray, e.g. for SnapshotMode “Automatic”
Example	pTango->SetSnapshotIndex(1, 5); // Set pointer to Index 5

5. SlideExpress Interface

This chapter describes additional DLL functions usable with SlideExpress. From application point of view there are only few differences between previous top loader and new front loader system.

Constant Name	Meaning	Top Loader	Front Loader
MAXMAGA	number of magazines	4	3
MAXROW	number of rows	50	30
MAXCOL	Number of columns	4	4

5.1. Eject

Description	Move magazine(s) and allow user access
C++	int LSX_Eject (int lLSID, int maga, int keep);
Parameters	maga → magazine number [1..MAXMAGA] keep → 0 to empty gripper before eject magazine(s) or 1 to keep slide(s) in gripper
Example	pTango->Eject(1, 1, 0);

5.2. Insert

Description	Magazine(s) are inserted and tested if seated and which slides are present. This function is precondition to use SlideSeated() and MagazinSeated()
C++	int LSX_Insert (int lLSID);
Parameters	-
Example	pTango->Insert(1);

5.3. SlideSeated

Description	Query if slide is present (seated) or not or unknown.
C++	int LSX_SlideSeated (int lLSID, int col, int row, int *status);
Parameters	col → col number [1..MAXCOL] row→ row number [1..MAXROW] status → returns slide status (-1 = unknown, 0 = empty, 1 = seated)
Example	pTango->SlideSeated (1, 4, 30, &status);

5.4. MagazinSeated

Description	Query if magazin is present (seated) or not or unknown.
C++	int LSX_MagazinSeated (int lLSID, int maga, int *status);
Parameters	maga → magazine number [1..MAXMAGA] status → returns magazine status (-1 = unknown, 0 = empty, 1 = seated)
Example	pTango->MagazinSeated (1, 1, &status); //check if magazine 1 is seated

5.5. GetGripper

Description	Query gripper status information. Returns status of gripper 1 and 2.
C++	int LSX_GetGripper (int lLSID, int *c1, int *r1, int *c2, int *r2);
Parameters	c1 → column number [-1, 0, 1..MAXCOL] of slide 1 in gripper r1 → row number [-1, 0, 1..MAXROW] of slide 1 in gripper c2 → column number [-1, 0, 1..MAXCOL] of slide 2 in gripper r2 → row number [-1, 0, 1..MAXROW] of slide 2 in gripper
Example	pTango-> GetGripper (1, &c1, &r1, &c2, &r2); //check status of gripper 1 and 2 c1, c2 → -1 = unknown, 0 = empty or 1 to 4 for magazine number r1, r2 → -1 = unknown, 0 = empty or 1 to 50 for slot number c1=1,r1=0 indicates priority slide 1 in gripper (obsolete for front loader) c2=1,r2=0 indicates priority slide 2 in gripper (obsolete for front loader)

5.6. SetGripper

Description	Set gripper status information. (possibly useful for slide sorting tasks)
C++	int LSX_SetGripper (int lLSID, int c1, int r1, int c2, int r2);
Parameters	c1 → column number [-1, 0, 1..MAXCOL] of slide 1 in gripper r1 → row number [-1, 0, 1..MAXROW] of slide 1 in gripper c2 → column number [-1, 0, 1..MAXCOL] of slide 2 in gripper r2 → row number [-1, 0, 1..MAXROW] of slide 2 in gripper
Example	pTango->SetGripper (1, 0, 0, 0, 0); //set gripper to "empty"

5.7. GetSlide

Description	Get slide(s) from addressed position in magazine or priority handler.
C++	int LSX_GetSlide (int lLSID, int col, int row, int mode);
Parameters	col → column number [1..MAXCOL] row → row number [1..MAXROW] (obsolete: ~~or [0] for priority handler)~~ mode → (0 = inspection, 1 = oiler, 2 = bar code reader)
Example	pTango-> GetSlide (1, 1, 1, 0);

5.8. PutSlide

Description	Put slide(s) back to addressed position in magazine or priority handler.
C++	int LSX_PutSlide (int lLSID, int col, int row);
Parameters	col → column [1..MAXCOL] row → slot number [1..MAXROW] (obsolete: ~~or [0] for priority handler)~~ If both parameters are 0 the DLL transmits !putslide without arguments. In this case Tango uses known gripper information to put slides back (if any).
Example	pTango->PutSlide (1, 4, 50); //put slide to magazine 4 slot 50.

Obsolete:

5.9. GetPrioHandlerPos

Description	Query actual priority handler position.
C++	int LSX_GetPrioHandlerPos (int lLSID, int *php);
Parameters	php → return value of actual priority handler position (55 = unknown, 0 = middle, -1 = shift in, 1 = pulled out)
Example	pTango-> GetPrioHandlerPos (1, &php);

Obsolete:

5.10. SetPrioHandlerPos

Description	Enables user to shift priority handler to required position. Handler is locked at destination or after 30s timeout
C++	int LSX_SetPrioHandlerPos (int lLSID, int php);
Parameters	php → specify destination 0 = middle, -1 = shift in, 1 = pulled out
Example	pTango-> SetPrioHandlerPos (1, 1); //enable user to pull out priority handler

6. TrayExpress Interface

This chapter describes optional DLL functions to be used in conjunction for TrayExpress.

6.1. Eject

Description	Eject magazine The TrayExpress moves magazine downwards and opens front cover to allow user operations like removing trays or loading trays.
C++	int LSX_Eject (int lLSID, int maga, int keep);
Parameters	maga → magazine number [1] (currently only 1 allowed) keep → 0 to empty gripper before eject magazine or 1 to keep tray in gripper
Example	pTango->Eject(1, 1, 0);

6.2. Insert

Description	From Cover is closed and magazine is inserted and tested if seated and which trays are present. This function is precondition to use SlideSeated() and MagazinSeated()
C++	int LSX_Insert (int lLSID);
Parameters	-
Example	pTango->Insert(1);

6.3. SlideSeated

Description	Query if tray is present (seated) or not or unknown.
C++	int LSX_SlideSeated (int lLSID, int maga, int slot, int *status);
Parameters	maga → magazine number [1] slot → slot number [1..50] status → returns slide status (-1 = unknown, 0 = empty, 1 = seated)
Example	pTango->SlideSeated (1, 1, 1, &status);

6.4. MagazinSeated

Description	Query if magazine is present (seated) or not or unknown.
C++	int LSX_MagazinSeated (int lLSID, int maga, int *status);
Parameters	maga → magazine number [1] status → returns magazine status (-1 = unknown, 0 = empty, 1 = seated)
Example	pTango->MagazinSeated (1, 1, &status); //check if magazine 1 is seated

6.5. GetGripper

Description	Query gripper status information. Returns status of gripper.
C++	int LSX_GetGripper (int lLSID, int *c1, int *s1, int *c2, int *s2);
Parameters	c1 → magazine number [-1, 0, 1..4] of slide in gripper s1 → slot number [-1, 0, 1..24] of slide in gripper c2 → dummy for compatibility with slide express s2 → dummy for compatibility with slide express
Example	pTango-> GetGripper (1, &c1, &s1, &c2, &s2); //check status of gripper 1 and 2 c1 → -1 = unknown, 0 = empty or 1 (magazine number) s1 → -1 = unknown, 0 = empty or 1 to 24 for slot number

6.6. SetGripper

Description	Set gripper status information. (possibly useful for tray sorting tasks)
C++	int LSX_SetGripper (int lLSID, int c1, int s1, int c2, int s2);
Parameters	c1 → magazine number [-1, 0, 1..4] of slide in gripper s1 → slot number [-1, 0, 1..50] of slide in gripper c2 → dummy for compatibility with slide express s2 → dummy for compatibility with slide express
Example	pTango->SetGripper (1, 0, 0, 0, 0); //set gripper to "empty"

6.7. GetTray

Description	Get tray from addressed position in magazine
C++	int LSX_GetTray (int lLSID, int slot, int mode);
Parameters	slot → slot number [1..24] mode → (0 = inspection, 1 = oiler, 2 = bar code reader)
Example	pTango-> GetTray (1, 1, 0);

6.8. PutTray

Description	Put tray back to addressed position in magazine
C++	int LSX_PutTray (int lLSID, int slot);
Parameters	slot → slot number [1..24]
Example	pTango->PutSlide (1, 10); //put tray to magazine slot 10.

6.9. GetRFID

Description	Get RFID of addressed tray (if properly seated in magazine)
C++	int LSX_GetRFID (int lLSID, int slot, int bank, int *plRFID);
Parameters	slot → slot number [1..MAXSLOT] bank → bank number [0 to 64] plRFID → pointer to int returns data stored in RFID transponder device
Example	pTango-> GetTray (1, 1, 0);

6.10. SetRFID

Description	Set RFID stores data into addressed magazine slot if tray is properly seated
C++	int LSX_SetRFID (int lLSID, int slot, int bank, int rfdata);
Parameters	slot → slot number [1..MAXSLOT] bank → bank number [2 to 64] (bank 0 and 1 are not writeable) rfdata → int contains customer data to be coded into RFID transponder device
Example	pTango-> SetTray (1, 1, 0);

6.11. GetNumberOfSlots

Description	Get number of available slots per magazine
C++	int LSX_GetNumberOfSlots (int lLSID, int *plSlots);
Parameters	plSlots → returns number of slots per magazine
Example	pTango-> GetNumberOfSlots (1, plSlots);

6.12. GetNumberOfMagazines

Description	Get number of available magazines Returns always 1 and is available for compatibility to SlideExpress only
C++	int LSX_GetNumberOfMagazines (int lLSID, int *plMagazines);
Parameters	plMagazines → pointer to int returns number [1]
Example	pTango-> GetNumberOfMagazines (1, plMagazines);

7. Express Interface Extensions

Following commands are superset of SlideExpress and TrayExpress commands and expand commands of previous 2 chapters.

7.1. GetLoaderType

Description	Get loader type Response depends on system configuration.
C++	int LSX_GetLoaderType (int lLSID, int *plLoaderType);
Parameters	plLoaderType → pointer to int returns loader type 0 => Tango / no Express Type 1 => SlideExpress 2 => Manual System (OEM special) 3 => Loader System (OEM special loader master device 1st Tango) 4 => Loader System (OEM special loader slave device 2nd Tango) Reserved for future expansion
Example	pTango->GetLoaderType (1, plLoaderType);

7.2. GetNumberOfRows

Description	Get number of magazine rows, e.g. max. number of slots to insert trays Response is number of magazine rows.
C++	int LSX_GetNumberOfRows (int lLSID, int *plRows);
Parameters	plRows → pointer to int returns number of magazine rows (1 for manual, 35 for loader system)
Example	pTango->GetNumberOfRows (1, plRows);

7.3. GetNumberOfColumns

Description	Get number of magazine columns, e.g. max number of slide sensors per slot/tray. Response is number of magazine columns.
C++	int LSX_GetNumberOfColumns (int lLSID, int *plCols);
Parameters	plCols → pointer to int returns number of magazine column (6 manual, 6 for loader system)
Example	pTango->GetNumberOfColumns (1, plCols);

7.4. GetTraySN

Description	Get tray SN returns unique tray RFID serial number of addressed slot / tray.
C++	int LSX_GetTraySN (int lLSID, int slot, int *plTraySN);
Parameters	plTraySN → pointer to int returns unique tray RFID serial number
Example	pTango->GetTraySN (1, 1, plTraySN);

7.5. GetTrayType

Description	GetTrayType returns tray type of addressed tray. (Data is read from RFID transponder.)
C++	int LSX_GetTrayType (int lLSID, int slot, int *plTrayType);
Parameters	plTrayType → pointer to int returns tray type (user coded data)
Example	pTango->GetTrayType (1, 1, plTrayType);

7.6. SetTrayType

Description	SetTrayType stores tray type data into RFID transponder of addressed slot / tray.
C++	int LSX_SetTrayType (int lLSID, int slot, int aTrayType);
Parameters	aTrayType → int data contains information of required tray type
Example	int aTrayType = 0x0100010a; //see customer specification requirements for explanation pTango->SetTrayType (1, 1, aTrayType);

7.7. SetCabinLED

Description	SetCabinLED on or off.
C++	int LSX_SetCabinLED (int lOn);
Parameters	lOn → 0 to switch OFF or 1 to switch ON the loader illumination
Example	pTango->SetCabinLED (1, 1); //switch ON illumination pTango->SetCabinLED (1, 0); //switch OFF

7.8. GetCabinLED

Description	GetCabinLED returns actual state of loader illumination
C++	int LSX_GetCabinLED (int lLSID, int *plState);
Parameters	plState → pointer to int returns illumination state
Example	pTango->GetCabinLED (1,plState);

7.9. SetLabelLED

Description	SetLabelLED on or off.
C++	int LSX_SetLabelLED (int lOn);
Parameters	lOn → 0 to switch OFF or 1 to switch ON the label illumination
Example	pTango->SetLabelLED (1, 1); //switch ON illumination pTango->SetLabelLED (1, 0); //switch OFF

7.10. GetLabelLED

Description	GetLabelLED returns actual state of label illumination
C++	int LSX_GetLabelLED (int lLSID, int *plState);
Parameters	plState → pointer to int returns illumination state
Example	pTango->GetLabelLED (1,plState);

8. Error Codes

8.1. Tango Error Messages

1	no valid axis name
2	no executable instruction
3	command line too long
4	unknown keyword
5	number outside range
6	wrong number of arguments
7	missing '!' or '?'
8	TVR not possible while axis is moving
9	switching axis on/off not possible while TVR is active
10	function not configured
11	automatic move not possible while jogging manual
12	limit switch activated
13	function not executable because encoder detected
14	error during calibration (limit switch not released)
27	STOP input active
29	amplifier off
50	one argument only expected
51	argument is not a number
52	keyword BEGIN or EOF missing
53	unexpected geo type
58	unexpected sequence
59	alpha and beta must not be equal
70	wrong CPLD data
71	ETS error
72	parameter is write protected
73	internal error
74	closed loop switched off due to parameter change
75	axis correction disabled or not enabled
76	io extension error
77	internal bus communication error
78	HDI input device error
79	xPos module error
80	internal error: HDI ISR not running
81	internal error: Encoder ISR not running
82	overload on motor connector +5V
83	overload on AUX I/O +5V supply
84	overload on encoder +5V supply
85	overload on AUX I/O +24V supply
86	low brake output voltage

Error messages from any Tango Express configuration or similar configurations

100	hardware missing (IO1)	
101	magazine not correct seated	
102	magazine slot is empty	
103	magazine slot is occopied	
104	sensor reports get failure	(during pull from magazine)
105	sensor reports put failure	(during insert in magazine)
106	sensor overmodulation	
107	magazine unknown	
108	ejector timeout	(magnet or other part)
109	priority handler is rear	
110	priority handler is in front	

111 priority handler is not locked
112 priority handler position not clear
113 priority handler timeout (front)
114 priority handler timeout (middle)
115 priority handler timeout (rear)
116 timeout open door
117 timeout close door
118 no priority handler available
119 gripper is not empty
120 gripper contains unknown clip or slide(s)
121 system not yet initialised
122 clip not correct seated in gripper
123 clamp not open
124 tray on stage
125 no tray in gripper
126 step mode finished
127 POS3 stop input
128 no tray on stage
129 crash detection

RFID error messages

These error messages may be generated in case an RFID device is connected to the Tango controller.

130 RF connect
131 RF timeout
132 RF address
133 RF NAK
134 RF sync
135 RF cancel
136 RF not OK
137 RF length
138 RF chksum

Piezo Z stage error messages

These error messages may be generated in case a Piezo Z stage device is connected to the Tango controller.

140 Piezo connect
141 Piezo timeout
142 Piezo address

Loader subsystem error messages

These error messages may be generated from Tango communication to any loader subsystem.

150 HL connect
151 HL timeout
152 HL cal X
153 HL cal Y
154 HL cal Z
155 HL insert
156 HL eject
157 HL get tray
158 HL put tray
159 HL protocol
160 HL hall tray detection
161 clamp electronic
162 no tray on stage but RFID read
163 escape position Z
164 HL tray alignment
165 tray on stage but no RFID

8.2. DLL Error Messages

0	no error
4001	internal error
4002	internal error
4003	function call with wrong LSID value or maximum of 8 open connections reached
4004	Unknown interface type (may appear with Connect...)
4005	Error while initializing interface
4006	No connection with controller (e.g. if SetPitch is called before Connect)
4007	Timeout while reading from interface
4008	Error during command transmission to Tango controller
4009	Command aborted (with SetAbortFlag)
4010	Command is not supported by Tango controller
4011	Manual Joystick mode switched on (may appear with SetJoystickOn/Off)
4012	No move command possible, because manual joystick enabled
4013	Closed Loop Controller Timeout (could not settle within target window)
4015	Limit switch activated in travel direction
4016	Repeated vector start!! (Closed Loop controller)
4017	Error while calibrating (Limit switch not correctly released)
4101	No valid axis name
4102	No executable instruction
4103	Too many characters in command line
4104	Invalid instruction
4105	Number is not inside allowed range
4106	Wrong number of parameters
4107	Either ! or ? is missing
4108	No TVR possible, while axis active
4109	-
4110	Function not configured
4111	-
4112	Limit switch active
4113	Function not executable, because encoder detected

9. Document Revision History

No.	Revision	Date	Changes	Remarks
01	A	26. Feb. 2009	Initial version	
02	B	27. Oct. 2011	New MW logo and appearance, Added new Error Codes, Added HwFactor, HwFactorB, ZwFactor, GetKey, GetKeyLatch, ClearKeyLatch	
03	C	22. Mar. 2013	Added: GetAccelFunc, SetAccelFunc GetSwitchType, SetSwitchType GetMotorSteps, SetMotorSteps Chapter 5: SlideExpress Interface	
04	D	08. Nov. 2013	Added: Chapter 2.4 LabVIEW Support	
05	E	24. Mar. 2014	Chapter 2.4 reformatted to Arial text	
06	F	18. Sep. 2014	Added: GetCommandTimeout SetCommandTimeout	
07	G	11. Jul. 2016	general review Chapter 6: TrayExpress interface	
08	H	04. Jul 2017	Added: GetSnapshotMode SetSnapshotMode SetSnapshotCount SetSnapshotPosArray ClearSnapshotPosArray GetSnapshotIndex SetSnapshotIndex Updated Error Codes Added ConnectSimple Interface Type -1	Based on Tango_DLL 1.384 (ML)
09	I	16. Aug. 2017	Added: SetAuxDigitalOutput Corrected IO descriptions	Based on Tango_DLL 1.385 (ML)
10	J	19. Oct. 2017	Added: SetLedBright	Based on Tango_DLL 1.387 (ML)
11	K	01. Nov. 2017	Added: Chapter 3.3 API State Diagram	
12	L	22.Jan. 2018	new: Chapter 7 Express IFC Extensions	Implemented since version 1.388 (FD)
13	M	28.Aug. 2018	Update of Chapter 8	
14	N	18.Dec. 2018	new: SetCabinLED / GetCabinLED SetLabelLED / GetLabelLED	Implemented since version 1.397 (FD)
15	O	19.Feb. 2019	Calibrate returns more specific error code GetLoaderType return value expanded prevent endless loop at removed USB	Implemented since version 1.398 (FD)
16	P	8. Mar. 2019	Update list of Tango error messages	