XL3 Control API Manual

1 February, 2023

NTi Audio, Im alten Riet 102, 9494 Schaan, Liechtenstein

Introduction

Using the Control API, the XL3 can be controlled and queried remotely, with a command set, from external client software. The command set allows you to set up the device, Start/Stop measurements, and retrieve measurement results.

> This document refers to XL3 firmware version 1.11 or higher

The XL3 provides three independent remote API interfaces

Control API	for controlling and querying the device
Data Streaming API	for retrieving measurement data
Audio Streaming API	for retrieving audio data

This document details the Control API. The Streaming APIs are further described in a separate XL3-Remote-Streaming-Manual.

> XL3 supports sFTP for file access.

Communication with the APIs is possible through a

- TCP Socket or
- WebSocket

> API (Programming Interface) Option required
> To include any of the three remote API interfaces in your application, you need the API (Programming Interface) Option installed on your XL3.

Quick Start

1. Connect the XL3 with the supplied USB-C cable to a USB port of your PC
2. Install MobaXterm free Home Edition PC software from https://mobaxterm.mobatek.net/download.html
3. When the software is running click **Start local terminal**
4. In the MobaXterm terminal window, type

```
nc usb.local 50300
```

5. The response should be a message that identifies the XL3, for example:

```
NTi Audio XL3 Control API, A3A-00100-D0, 1.11
```

6. Now you can use the Commands (see page 15) listed later in this document to get or set the values in your XL3. For example, get the resolution of the spectrum with

```
MEAS:SLM:SPEC:RES?
```

   the response could be

```
1/1
```

How to connect the XL3 physically

The XL3 is equipped with two USB ports (C and A) and Wi-Fi. Use one of the following possibilities to connect to the device:

Locally (USB-C to PC)

Use the USB-C port to connect a USB cable to your PC.

> ⓘ Only one XL3 is supported.

> ⓘ Besides using the IP Address, the XL3 can be reached using “usb.local”

Over Network

- **Wi-Fi**
- **Ethernet**
  XL3 supports USB (A or C) to Ethernet adapters.

> ⓘ On networks, the XL3 is identified by
> the Serial Number (e.g. “xl3-00100.local”) or
> the IP Address. (e.g. 192.168.201.123)

Worldwide Access - NTi Connect

Worldwide access is provided by the NTi Connect Service https://connect.nti-audio.com[1]

- Easy and secure access to webpage and data files
- Remote API from all around the globe
- No static/public IP / VPN or port forwarding required

This NTi Connect service is free up to a monthly data volume of 2 GB. Data download speed for access above 2 GB is slowed down. The “NTi Connect Open Data” subscription retains the communication at full speed, e.g. for downloading longer audio recordings.

1 http://connect.nti-audio.com

Communication Protocols

The APIs are available using the following communication protocols:

TCP

The APIs are available in a LAN over the following ports:

- Control API: 50300
- Data Streaming API: 50310
- Audio Streaming API: 50311

Example

```
nc XL3-00228.local 50300
Password:
1234
NTi Audio XL3 Control API, A3A-00100-D0, 1.11
```

WebSockets

WebSockets are using port 80/443 and are designed for communicating in WANs but also are available in LANs. NTi Connect offers internet access to the API WebSockets.

	LAN	**NTi Connect**
Control API	ws://xl3_addess/control/	wss://connect.nti-audio.com/api/ConnectKey/control/
Data Streaming	ws://xl3_addess/data/	wss://connect.nti-audio.com/api/ConnectKey/data/
Audio Streaming	ws://xl3_addess/audio/	wss://connect.nti-audio.com/api/ConnectKey/audio/

Password Message

After a connection is established, a password prompt is sent. Enter the Web Server Password from the XL3 in the *System Settings/Connections* screen. (If you don't remember the password enter a new one on the XL3)

> For a direct USB connection, any password will be accepted.

After entering the correct password, the XL3 confirms with the interface identification message

```
nc XL3-00228.local 50300
Password:
1234
NTi Audio XL3 Control API, A3A-00100-D0, 1.11
```

In case an incorrect password is supplied, the device sends an incorrect password message

```
Incorrect password
```

and the connection is closed.

If you try to open the same connection a second time you get the response message

```
Already in use
```

Code examples (Python 3.8)

TCP

```
import socket

MyXL3 = "xl3-00100.local"
# MyXL3 = "192.168.201.100"
# MyXL3 = "usb.local"

XL3Password = b"1234"

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((MyXL3, 50300))

# use makefile to provide a readline to the socket
sockFile = sock.makefile(mode='r')

passwordPrompt = sockFile.readline()
sock.send(XL3Password+b'\n')

identificationMessage = sockFile.readline()

sock.send(b"*idn?\n")
idn = sockFile.readline()

sockFile.close()
sock.close()
```

WebSocket

```
from websocket import create_connection
# websocket --> install websocket-client
# https://github.com/websocket-client/websocket-client

MyXL3 = "xl3-00100.local"
# MyXL3 = "192.168.201.100"
# MyXL3 = "usb.local"

XL3Password = b"1234"

url = "ws://" + MyXL3 + "/control/"
ws = create_connection(url)

passwordPrompt = ws.recv()
ws.send(XL3Password+b'\n')

identificationMessage = ws.recv()

ws.send(b"*idn?\n")
idn = ws.recv()

ws.close()
```

NTi Connect

```
from websocket import create_connection
# websocket --> install websocket-client
# https://github.com/websocket-client/websocket-client

ConnectKey = "ABCDE-FGHIJ"

XL3Password = b"1234"

url = "wss://connect.nti-audio.com//api//" + ConnectKey + "/control/"
ws = create_connection(url)

passwordPrompt = ws.recv()
ws.send((XL3Password+b'\n'))

identificationMessage = ws.recv()

ws.send(b"*idn?\n")
idn = ws.recv()

ws.close()
```

Commands

Command Structure

Remote commands are sent in ASCII format to the XL3. The line feed character (LF, 0x0A) is the message terminator for XL3 commands. So every command transmission from your PC to your XL3 or vice versa has to be terminated with a line feed LF. The measurement commands are divided into subsystems (i.e. logical groups).

Subsystem	Function
*	Device status common commands
INITiate	Status control for a measurement
MEASurement	Measurement result query commands
INPut	Sensor settings command
CALIbrate	Sensor calibration commands
ASD	Microphone Automatic Sensor Detection commands
SYSTem	System status commands

Command Format

The XL3 accepts either full keywords (long form) of commands or their abbreviations (short form). In the command list, the CAPITAL letters indicate the abbreviation. However, the XL3 accepts both lowercase and UPPERCASE letters as input, i.e. commands and parameters are not case-sensitive.

The command description contains special symbols (see page 43) and parameter types (see page 43).

> **Unquoted strings** in commands may not contain blanks or any other characters which may be interpreted as control characters as separators etc. Strings embedded in quotation marks may contain all characters except the message termination character (LF).

Command Responses

Commands that query the XL3 for values have a “?“ at the end of the command. Query commands return a response (Answer). Commands that set values in the XL3, have no “?“. Set commands do not return an Answer.

XL3 responds to each command with LF. The XL3 sends the response only after the command is executed. In other words, until the LF is returned, no other command can be processed by the XL3.

When there is no response, something went wrong. Please check the error queue (see page 14). This is especially helpful to synchronize to commands that need a little time. e.g. for the `INIT:START` command, the answer LF is sent when the measurement starts (after the settling time reaches 0). This makes it easy to synchronize further actions.

Error Queue

Errors are stored in the error queue and can be queried with the `SYSTem:ERRor?` command. If an error occurred during the execution of a command or a query, a corresponding error is pushed into the error queue. If the command is a query, a semicolon is returned instead of the expected answer.

Multiple Commands

Multiple commands separated by semi-colons ";" are supported

e.g. `meas:init;:syst:err?`

Notice that a command tree reset character (":" as shown in the example above) is required for a subsequent command if the last node is not re-used. The commands are processed one after the other. The next command is only executed when the last one is completed. The response of multiple commands is separated by ";" and one LF at the line end. If there is a semi-colon without a preceding value it is an error. Check the error queue.

Timeouts

When using the API, the following timeouts should be used (or longer ones):

For what?	Example	Timeout (minimum)
General	`MEAS:INIT`	3 sec
Starting a measurement	`INIT START`	13 sec
Switching measurement functions	`MEAS:FUNC SLM`	5.5 sec

The Command List

Device Status

`*IDN?`

Shortcut Identification: Reads the unique identification of the XL3.

Availability

Answer <Manufacturer>,<Unit>,<Serial Number>,<FW Version>

STR, STR, STR, STR

Common command according to IEEE488.2-1992 10.14.3

Example

```
*IDN?
NTi Audio XL3 Control API, A3A-00129-B1, 0.90.4760
```

`*CLS`

Shortcut Executes a status reset: Clears the error queue and the output buffer.

Availability always

Example

```
*CLS
LF
```

INITiate Subsystem

`INITiate`

Shortcut Starts or stops a measurement

Availability Sound Level Meter

Parameter [START|STOP]

CHARDAT

Example

```
INIT START
```

Details Time-dependent parameters like LAeq, LAFmax, etc. are undefined until START has been initiated. The start procedure may take a few seconds. If required, query INIT:STATE? to see, whether the start procedure has completed. When a measurement is stopped with STOP, the calculation of time-dependent parameters is stopped and the result stays constant.

> For the START command, the answer LF is sent at the moment the measurement really starts (after the settling time reaches 0). This makes it easy to synchronize other actions.

> For the STOP command, the answer LF is sent after the save dialog is closed. (In automatic save mode after 7 seconds). When logging is on and the save dialog configuration is set to “manually” or “prompt” the system will hang. Workaround: As long as the xl3 is logging the device always shows a save dialog after stop. To avoid this save dialog, switch off logging (data and audio) and in */System Settings/General* set the Save Configuration to “manually”. If logging is needed, set the Save Configuration to “automatic” and the save dialog will close after 7 seconds.
> The save dialog will be suppressed for remote in a future version.

`INITiate:STATe?`

Shortcut Queries the run status of a measurement

Availability always

Answer [STOPPED|FROZEN|SETTLING|RUNNING|PAUSED|UNDEFINED]

CHARDAT

Example

```
INIT:STATE?
RUNNING
```

MEASure Subsystem

`MEASure:INITiate`

Shortcut Triggers a measurement

Availability always

Example

```
MEAS:INIT
```

Details All measurements results of the MEASure subsystem are stored synchronously by this command.

Before the first MEAS:INIT has been sent, all measurement values are undefined.

A typical workflow is

```
INIT START
MEAS:INIT
MEAS:SLM:123? <para1>, <para2>
MEAS:INIT
MEAS:SLM:123? <para1>, <para2>
...
```

`MEASure:TIMer?`

Shortcut Queries the actual measurement timer value.

Availability Sound Level Meter

Answer <timer> sec

float UNIT

0.1 seconds resolution (1 decimal)

Example

```
MEAS:INIT
MEAS:TIMER?
3765.0 sec
```

Details This represents the time since initiating START

MEASure:FUNCtion

Shortcut	Defines the active measurement function
Availability	Run state = STOPPED
Parameter	[SLM|RT|SI]
	CHARDAT
Example	`MEAS:FUNC SLM`
Details	Switching between measurement functions may take 1-2 seconds.
Restrictions	The SI parameter is only available if the Sound Insulation Option is installed

MEASure:FUNCtion?

Shortcut	Queries the active measurement function
Availability	always
Answer	[SLM|RT|SI]
	CHARDAT
Example	`MEAS:FUNC?` `SLM`

MEASure: SLM Subsystem

`MEASure:SLM:123?`

Shortcut	Queries a broadband measurement result of the Sound Level Meter.
Availability	Sound Level Meter
Parameter	[LxS|LxSMAX|LxSMIN|LxF|LxFMAX|LxFMIN|LxEQ|Prev_LxEQ|LxPK|LxPKMAX|LyEQ_gt|LyEQ_gtMAX|LAFT3|LAFT3EQ|LAFT5|LAFT5EQ|LAFT5EQ-LAEQ|LCEQ-LAEQ|k1|k2] CHARDAT Replace lowercase letters as follows. x = [A|C|Z] y = [A|C] t = [5sec|10min|15min|60min] One of the four settings specified on the */Sound Level Meter/Gliding Leq Levels* page of the XL3 e.g. LAEQ_g5sec for LAEQ_g5“ or LCEQ_g15minMAX or LCEQ_g15’max
Parameter (Additional with installed Extended Acoustic Pack)	[LxI|LxIEQ|LxIMAX|LxIMIN|Ln%|LAIEQ-LAEQ] CHARDAT x = [A|C|Z] n = [1|5|10|50|90|95|99] One of the seven statistic values specified on the */Sound Level Meter/Level Statistics* page of the XL3, e.g. L90.0% (if the decimal place is zero you can also use L90%)
Answer	<Level> dB, [OK|UNDEF|LOW|OVLD] float UNIT CHARDAT

Example	`INIT START`
	`MEAS:INIT`
	`MEAS:SLM:123? LASMAX`
	`53.8 dB, OK`

Details

Returns a broadband result parameter that has been stored by the last MEAS:INIT command. If an error occurs, e.g. the parameter is unknown, a ";" is returned.

Statistic values:

- For custom settings use the custom values to read
  e.g. MEAS:SLM:123? L33.3%
- Remotely changing/reading the settings is not implemented.
- Be aware of the decimal separator. Use the Decimal Separator Configuration from the */System Settings/General* page.

> All parameter keywords are listed in their full keyword notation even if there are some lowercase letters used to describe special cases. No parameter keywords have abbreviations.

Call with multiple parameters

This command accepts up to 10 parameters, each has to be separated by a comma.

Example

```
INIT START
MEAS:INIT
MEAS:SLM:123? LASMAX, LAFMAX, LZSMAX, LZFMAX
52.1 dB, OK;54.8 dB, OK;6dB, OK;65.3 dB, OK
```

Details

Because the line feed character (LF) is reserved as the message termination character, the responses for each parameter are separated by semi-colons.

If the query for a specific parameter generates an error an empty field is returned and the corresponding errors are pushed into the error queue. E.g. if L55% is not available:

```
MEAS:SLM:123? LASMAX, L55%, LAFMAX, L5%
52.1 dB, OK;;54.8 dB, OK;
```

`MEASure:SLM:123:DT?`

Shortcut	Queries a broadband dt measurement result of the Sound Level Meter.
Availability	Sound Level Meter
Parameter	[LxSMAX|LxSMIN|LxFMAX|LxFMIN|LxEQ|LxPKMAX] CHARDAT x = [A|C|Z]
Parameter (Additional with installed Extended Acoustic Pack)	[LxIMAX|LxIMIN|LxE] CHARDAT x = [A|C|Z]
Answer	<Level> dB, [OK|UNDEF|LOW|OVLD] float UNIT CHARDAT

Example	`INIT START`
	`MEAS:INIT`
	`MEAS:SLM:123:dt? LASMAX`
	`53.8 dB, OK`

Details	Queries a broadband result parameter of the Sound Level Meter that has been stored with the last MEAS:INIT command. dt measurements are cleared after each MEAS:INIT, so this function returns the e.g. LEQ between two MEAS:INIT commands. The values have the same meaning as the dt values found in XL3 log files. If the parameter is unknown, a ";" is returned. ⓘ All parameter keywords are listed in their full keyword notation even if there are some lowercase letters used to describe special cases. Parameter keywords don't have abbreviations.
Call with multiple parameters	This command accepts up to 10 parameters, each has to be separated by a comma.

Example

```
INIT START
MEAS:INIT
MEAS:SLM:123:dt? LASMAX, LAFMAX, LZSMAX, LZFMAX
52.1 dB, OK;54.8 dB, OK;6dB, OK;65.3 dB, OK
```

Details

Because the line feed character (LF) is reserved as the message termination character, the responses for each parameter are separated by semi-colons.

If the query for a specific parameter generates an error, an empty field is returned e.g. for the command

MEAS:SLM:123:dt? LASMAX, LAIMAX, LAFMAX, LCIMAX

if the Extended Acoustic Pack Option is not installed, the following string is returned:

52.1 dB, OK;;63.7 dB, OK;

and the corresponding errors are pushed into the error queue.

`MEASure:SLM:SPECtrum?`

Alternative command keywords for XL2 compatibility: MEASure:SLM:RTA?

Shortcut	Queries the spectral results of the Sound Level Meter.
Availability	Sound Level Meter
Parameter	[LIVE|MAX|MIN|EQ|CAPT|HOLD3] CHARDAT
Parameter (Additional with installed Extended Acoustic Pack)	[E|n%] CHARDAT n = [1|5|10|50|90|95|99] One of the seven statistic values specified on the */Sound Level Meter/Level Statistics* page of the XL3, e.g. 10.0% (if the decimal place is zero you can also use 10%)
Answer	{$Level_n$,} dB|dBu|dBV|V, [OK|UNDEF|OVLD] floats UNIT CHARDAT 1/1 Oct: n = 12, f_{start} = 8 Hz 1/3 Oct: n = 36, f_{start} = 6.3 Hz Levels sorted from lowest to highest frequency

Example	`INIT START`
	`MEAS:INIT`
	`MEAS:SLM:SPEC? EQ`
	`46.3,50.7,34.5,45.4,42.2,37.2,39.0,39.8,32.1,28.5,29.8,31.0 dB, LOW`

Details	Queries the spectral results of the Sound Level Meter that have been stored by the last MEAS:INIT command. If the parameter is unknown, a ";" is returned. The unit (dB, dBu, dBV, V) is adopted by the setting of the user interface.

`MEASure:SLM:SPECtrum:DT?`

Alternative command keywords for XL2 compatibility: MEASure:SLM:RTA:DT?

Shortcut	Queries the dt spectral results of the Sound Level Meter.
Availability	Sound Level Meter
Parameter	[EQ] CHARDAT
Parameter (Additional with installed Extended Acoustic Pack)	[E] CHARDAT
Answer	{Level$_n$,} dB|dBu|dBV|V, [OK|UNDEF|OVLD] floats UNIT CHARDAT 1/1 Oct: n = 12, f_{start} = 8 Hz 1/3 Oct: n = 36, f_{start} = 6.3 Hz Levels sorted from lowest to highest frequency

Example	`INIT START`
	`MEAS:INIT`
	`MEAS:SLM:SPEC:DT? EQ`
	`46.3,50.7,34.5,45.4,42.2,37.2,39.0,39.8,32.1,28.5,29.8,31.0 dB, LOW`

Details	Queries the spectral results parameter of the Sound Level Meter that has been stored by the last MEAS:INIT command. dt measurements are cleared after each MEAS:INIT, so this function returns the LEQ of LE between two MEAS:INIT commands. The values have the same meaning as the dt values found in XL2 log files. If the parameter is unknown, a ";" is returned. The unit (dB, dBu, dBV, V) is adopted by the setting of the user interface.

`MEASure:SLM:SPECtrum:RESolution`

Alternative command keywords for XL2 compatibility: MEASure:SLM:RTA:RESOlution

Shortcut	Defines the resolution, in which the RTA results are acquired.
Availability	Sound Level Meter with run state = STOPPED
Parameter	[1/1|1/3|OCT|TERZ]
	CHARDAT
Example	`MEAS:SLM:SPEC:RES 1/3`

`MEASure:SLM:SPECtrum:RESolution?`

Alternative command keywords for XL2 compatibility: MEASure:SLM:RTA:RESOlution?

Shortcut	Queries the resolution, in which the RTA results are acquired.
Availability	Sound Level Meter
Answer	[1/1|1/3]
	CHARDAT
Example	`MEAS:SLM:SPEC:RES?` `1/3`
Details	The response of the alternative query returns the standard parameter keywords.

`MEASure:SLM:SPECtrum:WEIGhting`

Alternative command keywords for XL2 compatibility: MEASure:SLM:RTA:WEIGhting

Shortcut	Defines the frequency and time weighting, in which the RTA results are acquired.
Availability	Sound Level Meter with run state = STOPPED
Parameter	[AF|AS|CF|CS|ZF|ZS]
	CHARDAT
Example	`MEAS:SLM:SPEC:WEIG CF`

`MEASure:SLM:SPECtrum:WEIGhting?`

Alternative command keywords for XL2 compatibility: MEASure:SLM:RTA:WEIGhting?

Shortcut	Queries the frequency and time weighting, in which the RTA results are acquired.
Availability	Sound Level Meter
Answer	[AF|AS|CF|CS|ZF|ZS]
	CHARDAT
Example	`MEAS:SLM:SPEC:WEIG?` `ZF`
Details	The response of the alternative query returns the standard parameter keywords.

`MEASure:SLM:GLEQtime#`

Shortcut Sets one of the 4 gliding LEQ times

Availability Sound Level Meter with run state = STOPPED

Suffix # is a value from 1 to 4:

Parameter [1SEC|5SEC|10SEC|15SEC|30SEC|1MIN|5MIN|10MIN|15MIN|30MIN|60MIN]

CHARDAT

Example

```
MEAS:SLM:GLEQ2 10MIN
```

`MEASure:SLM:GLEQtime#?`

Shortcut Queries one of the gliding LEQ time settings.

Availability Sound Level Meter

Suffix # is a value from 1 to 4

Answer [1SEC|5SEC|10SEC|15SEC|30SEC|1MIN|
5MIN|10MIN|15MIN|30MIN|60MIN]

CHARDAT

Example

```
MEAS:SLM:GLEQ2?
10MIN
```

`MEASure:SLM:GLEQtime:RESet`

Shortcut Resets all 4 gliding LEQ times to their default setting

Availability Sound Level Meter with run state = STOPPED

Example

```
MEAS:SLM:GLEQ:RES
```

`MEASure:SLM:SPLit:ENABle`

Shortcut Enables the daily splitting of the logging/recording.

Availability Sound Level Meter with run state = STOPPED

Parameter [OFF|ON]

CHARDAT

Example

```
MEAS:SLM:SPLIT:ENAB ON
```

`MEASure:SLM:SPLit:ENABle?`

Shortcut Queries the setting of the daily splitting.

Availability Sound Level Meter

Answer [OFF|ON]

CHARDAT

Example

```
MEAS:SLM:SPLIT:ENAB?
ON
```

`MEASure:SLM:SPLit:OFFSet`

Shortcut Defines the time offset since midnight in seconds for the daily splitting of the logging/recording.

Availability Sound Level Meter with run state = STOPPED

Parameter <offset>

NUM_L

Unsigned int [0…86399]

Example

```
MEAS:SLM:SPL:OFFS 10800
```

`MEASure:SLM:SPLit:OFFSet?`

Shortcut	Queries the time offset since midnight for the daily splitting.
Availability	Sound Level Meter
Answer	<offset> NUM_L Unsigned int
Example	`MEAS:SLM:SPL:OFFS?` `10800`

MEASure:RT60 Subsystem

`MEASure:RT60?`

Shortcut	Queries the average results of the RT60 analyzer
Availability	Reverberation time measurement function
Parameter	[EDT|T15|T20|T30] CHARDAT
Answer	<results> array of float and array of status characters 1/1 Oct: n = 8 1/3 Oct: n = 32
Example	`MEAS:RT60? T30` `(0.18,0.16,0.31,0.18,0.39,0.41,0.36,0.33),(<,<,-,-,-,-,-,-),OK`
Details	Each value in the values array has its own status character in the second status array, the characters have the following meaning: – ok, undefined < decay too short > decay too long N low SNR D insufficient SNR ~ decay not linear E error This command returns the average results. If no measurement has been started undefined value is returned (see second example), an average result is available when at least a cycle was measured, after each cycle the intermediate average result is returned and when the measurement is terminated the final average is returned. The number of values in the array depends on the selected octave resolution.

MEASure:RT60:RESolution

Shortcut Defines the resolution, in which the RT60 results are acquired.

Availability always

Parameter [1/1|1/3]

CHARDAT

Example

```
MEAS:RT60:RES 1/1
```

Restrictions The parameter 1/3 is only available if the Extended Room Acoustics Option is installed.

MEASure:RT60:RESolution?

Shortcut Queries the resolution, in which the RT60 results are acquired.

Availability always

Answer [1/1|1/3]

CHARDAT

Example

```
MEAS:RT60:RES?
1/1
```

`MEASure:RT60:TRIGger:LEVel:MINimum`

Shortcut Defines the minimum trigger level.

Availability always

Parameter <level>

NUM_L

Float value in dBSPL

Example

```
MEAS:RT60:TRIG:LEV:MIN 80.0
```

Details The value must be in the range [-100 … +200]

`MEASure:RT60:TRIGger:LEVel:MINimum?`

Shortcut Queries the minimum trigger level.

Availability always

Answer <level>

NUM_L

Float value in dBSPL

Example

```
MEAS:RT60:TRIG:LEV:MIN?
80.0
```

INPut Subsystem

`INPut:PHANtom`

Shortcut	Configures the phantom power setting.
Availability	When an ASD sensor is not connected.
Parameter	[OFF|ON]
	CHARDAT
Example	`INP:PHAN OFF`

`INPut:PHANtom?`

Shortcut	Queries the phantom power setting.
Availability	always
Answer	[off|on]
	CHARDAT
Example	`INP:PHAN?` `off`

CALIbrate Subsystem

The abbreviation CALI instead of CAL was only defined for compatibility with XL2 commands.

`CALIbrate:MICrophone:TYPE?`

Shortcut Queries the sensor type

Availability always

Answer STR

Example

```
CALI:MIC:TYPE?
M4260
```

Details If no ASD microphone is currently connected, the command returns *noASD*

`CALIbrate:MICrophone:SERIal?`

Shortcut Queries the sensor serial number

Availability always

Answer STR

Example

```
CALI:MIC:SERI?
1234
```

Details If no ASD microphone is currently connected, the command returns *0*

`CALIbrate:MICrophone:SENSitivity:VALUe?`

Shortcut Queries the microphone sensitivity in V/Pa

Availability always

Answer <sens> V/Pa, OK

floats UNIT CHARDAT

Example

```
CALI:MIC:SENS:VALU?
20.0e-3 V/Pa, OK
```

CALIbrate:MICrophone:CIC

Shortcut Switches M2340 microphones (and MA230 mic preamps) into self-test mode. This means, the microphone generates a reference tone (square wave with 31.25 Hz or 1 kHz) which can be measured by the XL3.

Availability always

Parameter [OFF|F1|F2]

CHARDAT

Example

```
CALI:MIC:CIC OFF
```

Details To avoid CIC staying on (for example, if remote connection breaks during CIC measurement), CIC turns off automatically after 60 seconds. If you need the CIC signal for a longer time, retrigger the signal by sending the F1 or F2 command again within 60 seconds.

This command will fail and return with an error code if no sensor is connected or if the microphone doesn't support this operation.

CALIbrate:MICrophone:TEMPerature?

Shortcut Queries the value of the integrated temperature sensor of the M2340 microphone.

Availability always

Answer <temperature> DEG_C, OK

floats UNIT CHARDAT

Example

```
CALI:MIC:TEMP?
23.7 DEG_C, OK
```

Details This command will fail and return with an error code if no sensor is connected or if the microphone doesn't support temperature readout.

SYSTem Subsystem

These commands are subject to change

SYSTem:CONFiguration:Stream

Shortcut Sends a configuration that is then written to the XL3

Availability Run state = STOPPED

Parameter #dnnn d specifies the number of digits of nnn, nnn represents the number of bytes (characters) of the following json object,
a json object

BIN (with JSON payload)

Example

```
SYST:CONF:S #500037{"SLM":{"spectrum":{"octres":"1/3"}}}
```

Details The transferred configuration may be the whole or any partial configuration out of the complete configuration as described in Available JSON Parameters (see page 45).

If only a partial configuration is sent, only the stated configuration values are changed. All other existing configuration values are retained by the XL3.

If the JSON format is invalid the command is rejected.

The parameter may contain all values (characters) except the message termination character (LF, 0x0A).

SYSTem:CONFiguration:Stream?

Shortcut Reads a complete system configuration and returns it as json object

Availability always

Suffix # is specifies the page, valid from 1 to 16, no suffix refers to page 1

Answer #dnnn d specifies the number of digits of nnn, nnn represents the number of bytes (characters) of the following json object,
a json object

BIN (with JSON payload)

Example

```
SYST:CONF:S?
#500725{"SLM":{"spectrum":{"octres":"1/6","hold……
```

Details The system configuration is described in Available JSON Parameters (see page 45).

SYSTem:CONFiguration:Encoded

Shortcut Same as SYSTem:CONFiguration:Stream with the parameter encoded in base64

Availability Run state = STOPPED

Parameter BASE64 unquoted string

STR

Example

```
SYST:CONF:E
ewogICAgIlNMTSI6IHsKICAgICAgICAic3BlY3RydW0iOiB7CiAgICAgICAgIC……
```

Details The transferred configuration may be the whole or any partial configuration out of the complete configuration as described in Available JSON Parameters (see page 45).

If only a partial configuration is sent, only the stated configuration values are changed. All other existing configuration values are retained by the XL3.

If the JSON format is invalid or the string is not a valid base64 encoded string the command is rejected.

The decoded parameter may contain all values (characters) except the message termination character (LF, 0x0A).

SYSTem:CONFiguration:Encoded?

Shortcut Same as SYSTem:CONFiguration:Stream? with the Answer encoded in base64

Availability always

Answer BASE64 unquoted string

STR

Example

```
SYST:CONF:E?
ewogICAgIlNMTSI6IHsKICAgICAgICAic3BlY3RydW0iOiB7CiAgICAgICAgIC……
```

Details The system configuration is described in Available JSON Parameters (see page 45).

SYSTem:CONFiguration:FILE

Shortcut Instructs the XL3 to load the testconfig.xl3cfg test configuration file into the XL3

Availability Run state = STOPPED

Example

```
SYST:CONF:FILE
```

Details The file /media/Configurations/testconfig.xl3cfg must exist.

The testconfig.xl3cfg must follow the Configuration File Format (see page 51)

SYSTem:CONFiguration:FILE:LOAD

Shortcut Instructs the XL3 to load a configuration into the XL3 from a specified file

Availability Run state = STOPPED

Parameter file name

QSTR

Example

```
SYST:CONF:FILE:LOAD “<filename>”
```

Details The file /media/Configurations/<filename>.xl3cfg must exist.

The <filename>.xl3cfg must follow the Configuration File Format (see page 51)

SYSTem:CONFiguration:DOCumentation

Shortcut Creates media/Configurations/documentation.txt which contains all configuration options available with the current FW

Availability always

Example

```
SYST:CONF:DOC
```

Details documentation.txt is formatted as json pretty

`MEASure:SLM:LOGging:FSTRucture`

Shortcut Sets the device to standard or noise measurement mode

Availability Run state = STOPPED

Parameter [STANdard|NOISemonitoring]

CHARDAT

Example

```
MEAS:SLM:LOG:FSTR NOIS
```

`MEASure:SLM:LOGging:FSTRucture?`

Shortcut Gets the device mode

Availability always

Example

```
MEAS:SLM:LOG:FSTR?
NOISEMONITORING
```

`MEASure:SLM:LOGging:FFORmat`

Shortcut Selects the SLM log file format

Availability Run state = STOPPED

Parameter [TEXT_TSV|BINary]

CHARDAT

Example

```
MEAS:SLM:LOG:FFOR BIN
```

`MEASure:SLM:LOGging:FFORmat?`

Shortcut Gets the device mode

Availability always

Example

```
MEAS:SLM:LOG:FFOR?
BINARY
```

`SYSTem:ERRor?`

Shortcut Queries the error queue

Availability always

Answer {errno$_n$,}

NUM_L

n 50

Example

```
SYST:ERR?
40, 70
SYST:ERR?
0
```

Details

XL3 Error codes differ from XL2 Error codes

There are different classes of errors. Some errors refer to the command syntax, others to internal states of the XL3. Refer to the Error List (see page 46)

Every error is pushed into the error queue that must be queried to get the error number.

Several error numbers from unused parser features may be returned. These numbers are not listed here.

Appendix

List of symbols used in the command description

Symbol	Description
:	Colons separate keywords of an XL3 command.
[ ]	Square brackets enclose the *list of available parameters*, out of which 1 parameter must be selected.
|	A vertical line reads as a logical „OR“, i.e. this sign separates *alternative* parameters.
< >	Triangle brackets enclose the *variable parameters* that must be set for a user-defined value.
{ }	Braces have the same meaning as triangle brackets, except that the enclosed parameters can be included *several* times.
,	Commas separate arguments in an arguments list.
?	The question mark indicates a *query*.
( )	Round brackets enclose comments.

Parameter Types

Type	Description
STR	Unquoted String
QSTR	String embedded in double quotation marks
CHARDAT	Character Data
UNIT	Unit character data
NUM_L	Numerical Value • Integer • Float • With optional unit
BIN	IEEE488 Binary Data

Type	Description
JSON	An open standard file format

Differences between XL3 and XL2

Difference	XL2	XL3
Line ending	“CR LF”	“LF”
Response to set commands (Command without “?”)	no	“LF” after command completion of set command
Multiple Commands separated by semicolon	no	yes
Device access	Only local PC via USB virtual Com port	Through a TCP socket (see page 8) or a WebSocket (see page 8) or a browser through NTi Connect Server (see page 5)
Command Keywords	Short or any variant of the full form	Short or complete long keyword is required
Channel open	No response	On channel open the XL3 sends the Channel Identification Message or a *Password:* prompt
Symbols for second and minute in parameters	“, ‘	symbols are replaced with words “sec” and “min”

Available JSON Parameters

The XL3 can be configured using JSON and the commands `SYSTem:CONFiguration:Stream`, `SYSTem:CONFiguration:Encoded`, `SYSTem:CONFiguration:FILE`, or `SYSTem:CONFiguration:FILE:LOAD`

A full list of available configuration parameters can be found in the `/media/Configurations/documentation.txt` file. This file is automatically created when the XL3 starts up.

Error List

0	no error (queue is empty)
10	No input command to parse
14	Numeric suffix is invalid value
20	Parameter of type Numeric Value overflowed its storage
30	Wrong units for parameter
40	Wrong type of parameter(s)
41	Wrong format of a type of parameter(s)
50	Wrong number of parameters
60	Unmatched quotation mark (single/double) in parameters
65	Unmatched bracket
70	Command keywords were not recognized
80	Item numeric insertion is an invalid value
82	DESIGN ERROR: Too many item numeric insertions in parameter Spec
83	No numeric insertion value found in the input item
85	Numeric insertion value expected by translator
90	No numeric suffix value but expected
91	Numeric suffix value is out of valid range
150	Invalid Date Time Format
151	Content of Date Time input is invalid
200	IMPLEMENTATION ERROR: Called translator doesn't fit to the declared parameter list in the specs
201	IMPLEMENTATION ERROR: Called translator doesn't fit to the specific parameter declaration
202	IMPLEMENTATION ERROR: Parameter index is out of range
203	DESIGN ERROR: At least the first parameter must be defined as required in the param spec

204	IMPLEMENTATION ERROR: Requested parameter is not implemented in the local query function
300	Timeout while waiting for response message from core
301	Timeout while waiting for response message from core (configuration channel)
310	Requested broadband signal is not available (gliding eq or percentile)
310	Requested wideband signal is not available (gliding eq or percentile)
311	Requested spectral signal is not available (percentile)
350	Input verification: The ASD Hexstream is not valid
351	Input verification: The JSON input stream is not valid (see json parser error for details)
352	Input verification: The BASE64 input stream is not valid
353	Input verification: The JSON input stream is empty
370	Invalid configuration file header
371	Unexpected empty configuration file header
400	IMPLEMENTATION ERROR: No command handler specified for this command
401	Access to uninitialized mirror data
402	Invalid parameter passed to command handler (e.g. unexpected value for query)
403	Failed to translate parsed command parameter from the ParamSpec list into internal enumeration type
410	DESIGN ERROR: Dynamic downcast to branch/leaf failed
411	IMPLEMENTATION ERROR: Invalid parameter in deserialization (should never occur)
450	API option required to executed this command
550	RUNTIME ERROR: File I/O error
551	RUNTIME ERROR: File buffer memory error
553	RUNTIME ERROR: File I/O: unexpected content
560	Value input verification: Invalid enum value (not in list)
561	Value input verification: Enum type is not text

562	Value input verification: Float type is not a number
563	Value input verification: Signed integer type is not a number
564	Value input verification: Unsigned integer type is not a number
565	Value input verification: Value is not a text
600	Failure while handling response message received from the core
601	Message received from the core does not contain valid RT data
750	IMPLEMENTATION ERROR: Not further specified error during execution (parsing, translation, handling)
751	IMPLEMENTATION ERROR: Exception during error translation (exception due to unknoen error, recompilation with new geberated code required)
800	Error queue overflow
1001	Value is out of range
1002	Command rejected: Measurement is running
1003	Not implemented
1004	Parameter is not available
1005	Request forbidden
1006	Internal memory error
1007	ASD device not present
1008	ASD page index out of valid range
1009	ASD operation failed
1010	License required
1011	(internal error) Attempt to load configuration parameter while not in loading state
1012	Configuration contains incomplete SLM limit setup
1013	Configuration file I/O error
1014	Memory error while processing configuration file
1015	Configuration file format is invalid

1016	Configuration file header is invalid
1017	Failed to parse configuration
1018	Internal error: Timeout while waiting for load response
1019	Memory error while loading configuration
1020	Internal error: Unknown error
1021	Internal error: Error not further specified
1022	The specified audio sample rate for recording is not allowed for compressed format
1023	Configuration load error: Specified Enum parameter not found
1024	Configuration load error: Enum type mismatch
1025	Configuration load error: Float type mismatch
1026	Configuration load error: Signed Int type mismatch
1027	Configuration load error: Unsigned Int type mismatch
1028	Configuration load error: Text type mismatch
1029	Warning: The configuration file contains unknown entries
1030	Configuration load error: Selected accessory doesn’t fit to the connected microphone
1031	Configuration load error: Selected diffuse field correction doesn’t fit to the connected microphone
1032	The connected sensor (microphone) doesn’t support the current operation
1033	There is no sensor (microphone) connected
1034	The current sensor (microphone) operation has failed
1035	Measurement Series is active
1036	Microphone is disconnected
1037	The preamplifier is not supported
1038	Unexpected change of preamplifier
1039	The preamplifier doesn't fit to the data from NTi server

1040	Updating Mic Model Number failed
1041	Updating Capsule Type failed
1042	Updating Microphone Sensitivity failed
1043	Unexpected change of the connected adapter
1044	Update to CS011 model number not allowed for this adapter
1045	This adapter model doesn’t support writing a new serial number
1046	Failed to update the serial number

Configuration File Format

A configuration file must have the extension *.xl3cfg*.

The configuration file consists of a header and a configuration data part. These two parts are divided by the separator row *##CONFIG:*

Each part is JSON formatted.

”MeasurementID”: [SLM|RT|SI] instructs the XL3 which screen to show

- SLM - Sound Level Meter
- RT - Reverberation Time
- SI - Sound Insulation

”UUID”: a Universal Unique Identifier

Example content of a <filename>.xl3cfg

```
{
"MeasurementID": "SLM",
"UUID": "00000000-0000-0000-0000-000000000000"
}
##CONFIG:
{"SLM":{"spectrum":{"octres":"1/3"}}}
```