NWP320

Network input panel - 2 x XLR + 3.5 mm jack + BT (4 CH)

Table of contents

NWP320 Commands List

ASCII Commands (NWP320 1.0.2)

This is the list ASCII Commands supported by this device. A ASCII command always follows the same structure:

```
#|Destination|Source|Type^Target^Command|Arguments|CRC|CRLF
```

This format uses 3 separator characters for different levels of separating each value in the message:

- Message separator '|':
  This separates a message into 7 blocks (if you include the start '#' and end <CRLF>)
- Block separator '^':
  This splits a block into logical elements. This is used to split the command in the message type, 'command' and 'target'
- Value separator '>':
  This splits up a logical single value in their primitives, for example: a target consists of a channel type and a channel index, split by '>'

Messages are Case sensitive, if the example shows the text in uppercase, this should always be uppercase!

Destination

The target device. This consists of 2 parts: `Device>Address`.

Device

This is the device type: **NWP320**

Address

This is the user-configurable device address, default: **1**. You can also leave this field blank; this causes all NWP320 devices that receive this command to respond.

Examples	Destination
default destination	`NWP320>1`
broadcast to all Nwp320 devices	`NWP320`

Device Matching

If the device type or address does not match, the message will be ignored. Device Address 0 is a special address and will always match (this can be seen as a broadcast)

Destination	Device address: `NWP320>2`	Remarks
`NWP320>2`	Destination Matches Device	this is an exact match
`NWP320>1`	Message ignored	the destination address does not match
`NWP320`	Destination Matches Device	the destination address will always match
`NWP320>0`	Destination Matches Device	Equivalent to `NWP320`
`CLIENT>2`	Message ignored	the device type does not match
`CLIENT`	Message ignored	the device type does not match

Source (optional)

The source address is optional when sending, but the device will always fill this field with its own address.

Examples	sent message	response message `NWP320>2`
broadcast to a Nwp320	`#|NWP320||...|<CRLF>`	`#||NWP320>2|...|<CRLF>`
send to a specific Nwp320	`#|NWP320>2||...|<CRLF>`	`#||NWP320>2|...|<CRLF>`
use a source address in the request message	`#|NWP320|CLIENT>1|...|<CRLF>`	`#|CLIENT>1|NWP320>2|...|<CRLF>`

Type

The type explains what the message wants to do. There are 3 supported message types:

Type	From	To	Explanation
`SET_REQ`	CLIENT	Nwp320	Change a setting in the Nwp320
`GET_REQ`	CLIENT	Nwp320	Request the current status of a setting in the Nwp320
`GET_RSP`	Nwp320	CLIENT	Response to either a GET_REQ or SET_REQ, if the request was valid

Command, Target, Arguments

These 3 parameters are explained together because they influence each other. The command dictates the meaning of the argument, while the target distinguishes which exact setting you want to change. the target can also influence the valid range of the argument.

Some commands (like the mixer) can have a range of arguments (for the mixer: all mixer volumes are individual arguments). In this case, the argument looks like `idx>val[^idx2>val2]`, where the part in between the brackets `[]` can appear 0 or more times.

- idx, ídx2, ...: the argument index
- val, val2, ...: the value at the specified index

VOLUME

Set a single Volume in dB

Argument (volume)
the requested Volume in dB

Target	Argument	Example (default value)
INPUT_XLR>1>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_XLR>1>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_XLR>2>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_XLR>2>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_JACK>1>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_JACK>1>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_JACK>2>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_JACK>2>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_BLUETOOTH>1>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>1>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_BLUETOOTH>2>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>2>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>1>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_DANTE>1>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>2>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_DANTE>2>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>3>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_DANTE>3>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>4>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^INPUT_DANTE>4>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>1>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>1>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>2>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>2>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>3>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>3>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>4>VOLUME>1	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>4>VOLUME>1^VOLUME|0|U|<CRLF>`

MUTE

mute an audio channel

Argument (enabled)
is the audio channel muted

Target	Argument	Example (default value)
INPUT_XLR>1>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_XLR>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_XLR>2>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_XLR>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_JACK>1>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_JACK>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_JACK>2>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_JACK>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_BLUETOOTH>1>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_BLUETOOTH>2>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>1>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_DANTE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>2>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_DANTE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>3>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_DANTE>3>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>4>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^INPUT_DANTE>4>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>1>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>2>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>3>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>3>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>4>VOLUME>1	options: TRUE,FALSE	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>4>VOLUME>1^MUTE|FALSE|U|<CRLF>`

MIXER

mixer slider for zones

Argument (volume)
mixing volume

Target	Argument index	Argument	Example (default value)
OUTPUT_DANTE>1>MIXER>1	min: 1, max: 12	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>1>MIXER>1^MIXER|1>0^2>-90^3>-90^4>-90^5>-90^6>-90^7>-9 <CRLF>`
OUTPUT_DANTE>2>MIXER>1	min: 1, max: 12	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>2>MIXER>1^MIXER|1>-90^2>0^3>-90^4>-90^5>-90^6>-90^7>-9 <CRLF>`
OUTPUT_DANTE>3>MIXER>1	min: 1, max: 12	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>3>MIXER>1^MIXER|1>-90^2>-90^3>0^4>0^5>-90^6>-90^7>-90^ <CRLF>`
OUTPUT_DANTE>4>MIXER>1	min: 1, max: 12	min: -90, max: 0	`#|NWP320>1||SET_REQ^OUTPUT_DANTE>4>MIXER>1^MIXER|1>-90^2>-90^3>-90^4>-90^5>0^6>0^7>-90^ <CRLF>`

CRC

The CRC block is calculated over the message starting from and including the first pipe "|", up to and including the last pipe **before** the CRC Block. These CRC's can ensure message integrity if desired.

CRC Type	Configuration	Format	Example	notes
None	/	U	`#|NWP320>1||SET_REQ^INPUT_XLR>1>VOLUME>1^VOLUME|0|U|<CRLF>`	'U' means unused
CRC16-ARC	• input reflected • output reflected • polynomial: 0x8005 • initial value: 0x0000 • final exor: 0x0000	XXXX	`#|ALL||SET_REQ^INPUT_LINE>1^VOLUME|0|C06C|<CRLF>`	calculator
CRC32	• input reflected • output reflected • polynomial: 0x4C11DB7 • initial value: 0xFFFFFFFF • final exor: 0xFFFFFFFF	XXXX	`#|ALL||SET_REQ^INPUT_LINE>1^VOLUME|0|D887125C|<CRLF>`	calculator

The examples in the table above are examples for calculating the CRC, they may not be a valid command for the NWP320

The CRC can ensure data integrity across unreliable data channels (RS232, RS485), but they are by no means a security measure! If someone has the knowledge and means to maliciously alter a message, correcting the CRC becomes trivial for the attacker. We support different kinds of CRC for maximum flexibility, but we recommend not using any so you do not get a false sense of security.

Stop bytes

The final 2 characters are denoted as <CRLF>, they mean "Carriage Return, Line Feed" or simply put a new line. Depending on the tool used to create the command, you can have different representations:

- CRLF
- \r\n
- 0x0D 0x0A

We support both CRLF and LF only