SONANCE

OPEN API FOR INSTALLERS

Edition 2026.24.1 - June 2026

Applies to: API 1.5 · firmware 1.8+ (incl. 2026.x)

Contents

1 Introduction

Applies to: API 1.5 · firmware 1.8+ (incl. 2026.x).

1.1 Revision History

> From edition `2026.x` onward the document uses calendar-versioned editions (derived from the release tag); earlier `R5.x` rows are retained for history.
>
> The **API Version** column is the value the firmware's `API_VERSION` register reports. It has stayed `1.5` since firmware 1.5 even though registers were later added (Zone Priority in 1.8, `TILT_SHELF` in 2026.x) — so it is not a reliable freshness signal.

Revision	Date	Changed By	API Version	Firmware
2026.24.1	09/06-2026	MAM	1.5	1.8+
5.4	08/01-2026	MAM	1.5	1.8+
5.3	19/03-2025	MAM	1.5	1.8+
5.2	21/11-2023	MAM	1.5	1.6+
5.1	24/08-2023	MAM	1.5	1.5+
5	27/06-2023	MAM	1.5	1.5+
4	16/01-2023	MAM	1.4	1.4+
3	05/09-2022	MAM	1.3	1.3+
2	09/03-2022	MAM	1.2	1.2+
1	16/12-2021	MAM	1.1	1.1+
0	11/11-2021	MAM	1.0	1.0+

1.1.1 Edition 2026.24.1: Firmware-Accurate Rebuild

- First calendar-versioned edition (supersedes the unreleased R5.x revisions).
- Corrected register definitions — value ranges, types, and enumerations — to match the firmware exactly; many had drifted (limiter thresholds, EQ and crossover filter types, gain ranges, GPIO functions, power-on mode, polarity).
- Added a full **Wire-Protocol Reference**: error-line format and error-code catalog, all command verbs, connection/session limits and keep-alive, value encoding, and subscription semantics.
- Documented route signal/clip metering (`ROUT-{RID}.DYN.*`) and clarified that `*.DYN.*` registers are subscription-only (streamed on `SUBSCRIBE DYN`).
- Corrected EQ band counts (input 5, output 10, speaker 15) and the crossover / clip-limiter value sets.

1.1.2 Revision 5.4: Documentation Update

- Updated documentation tooling and build process
- Renamed "Advanced Registers" section to "Output Speaker Processing Registers"
- Auto-generated register reference from YAML schema

1.1.3 Revision 5.3: Firmware 1.8 Changes

- Added: 4 Zone Priority Override Registers (`ZONE-{ZID}.PRIORITY-{ZP}.*`)
- Added: {ZP} Zone Priority variable definition

1.1.3.1 Registers Added (R5.3)

Register Name
ZONE-{ZID}.PRIORITY-{ZP}.SRC
ZONE-{ZID}.PRIORITY-{ZP}.AUTO
ZONE-{ZID}.PRIORITY-{ZP}.THRESHOLD
ZONE-{ZID}.PRIORITY-{ZP}.HOLD
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN_ENABLE

1.1.4 Revision 5.2: Firmware 1.6 Changes

- Updated: Input Channels updated to 8 channels. See Input Channels.

1.1.5 Revision 5.1

Clarifications and documentation updates - No API Changes

1.1.6 Revision 5: API 1.5 Changes

- Added: Dante Info Registers (SYSTEM.DANTE.*)
- Updated: Input Channels updated with Dante. See Input Channels.
- Updated: Input Sources updated with Dante. See Input Sources.
- Updated: Added Mix to Output Route Sources. See Output Route Sources.

1.1.6.1 Registers Added (R5)

Register Name
SYSTEM.DANTE.SOFTWARE_VERSION
SYSTEM.DANTE.FIRMWARE_VERSION
SYSTEM.DANTE.IP
SYSTEM.DANTE.MAC
SYSTEM.DANTE.LINK_SPEED
SYSTEM.DANTE.AES67_ENABLED
SYSTEM.DANTE.DEVICE_NAME
SYSTEM.DANTE.ENCODING
SYSTEM.DANTE.SAMPLE_RATE
SYSTEM.DANTE.CLOCK_STATE
SYSTEM.DANTE.MUTE_STATE

1.1.6.2 Registers Modified (R5)

- Added Mixes to Route Sources
- Added PowerMode AUDIO_DSP to register SETUP.POWER.POWER_ON

1.1.7 Revision 4: API 1.4 Changes

- Added: Input HPF
- Added: 5-Band Input EQ
- Added: Mixes as Zone Primary Src.
- Added: Zone Priority Src for Zone
- Added: Option to disable Mute to Zone
- Added: Zone Ducker
- Added: Option to limit zone sources (Wall Controller Specific)
- Added: Option to select output SPDIF source
- Added: Bandwidth limitation for Pink Noise Generator
- Added: Sine Generator
- Removed: Generator cannot be disabled

1.1.7.1 Registers Added (R4)

Register Name
IN.EQ.COUNT
IN-{IID}.HPF_ENABLE
IN-{IID}.EQ.BYPASS
IN-{IID}.EQ-{EID}.TYPE
IN-{IID}.EQ-{EID}.GAIN
IN-{IID}.EQ-{EID}.FREQ
IN-{IID}.EQ-{EID}.Q
IN-{IID}.EQ-{EID}.BYPASS
ZONE-{ZID}.PRIORITY_SRC
ZONE-{ZID}.MUTE_ENABLE
ZONE-{ZID}.SRC-{IID}.ENABLED
ZONE-{ZID}.DUCK.MODE
ZONE-{ZID}.DUCK.AUTO
ZONE-{ZID}.DUCK.THRESHOLD
ZONE-{ZID}.DUCK.ATTACK
ZONE-{ZID}.DUCK.RELEASE
ZONE-{ZID}.DUCK.HOLD
ZONE-{ZID}.DUCK.OVERRIDE_GAIN
ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE
GENERATOR.TYPE
GENERATOR.SINE.FREQ
GENERATOR.PINK.LPF_ENABLE
GENERATOR.PINK.LPF_FREQ
GENERATOR.PINK.HPF_ENABLE
GENERATOR.PINK.HPF_FREQ
MIX.COUNT
MIX-{MID}.NAME

Register Name
MIX-{MID}.GAIN-{IID}
ROUT-{RID}.SRC
ROUT-{RID}.SRC_CHANNEL
ROUT-{RID}.GAIN

1.1.7.2 Registers Removed (R4) GENERATOR.ENABLE |

1.1.8 Revision 3: API 1.3 Changes

- Added: Output Gain
- Added: Clip Limiter Mode
- Added: Security Registers for WebPage Security
- Added: Input Gain Min + Input Gain Max
- Added: Analog Volume Control Value register as Value
- Remove: Analog Volume Control Volume register
- Update: Input Gain - Range increase from [-10, 10] to [-15, 15] dB
- Update: Zone Gain - when using Analog Volume Control
- Update: SETUP.LAN and SETUP.WIFI registers as readonly

1.1.8.1 Registers Added (R3)

Register Name
ZONE-{ZID}.GAIN_MIN
ZONE-{ZID}.GAIN_MAX
OUT-{OID}.GAIN
OUT-{OID}.CLIP_LIMITER.MODE
VC-{VID}.VALUE
SYSTEM.SECURITY.PASSWORD_ENABLE
SYSTEM.SECURITY.PASSWORD_HASH

1.1.8.2 Registers Updated (R3)

Register Name	Change
IN-{IID}.GAIN	Limits
ZONE-{ZID}.GAIN	Limits, more
SETUP.LAN.NETWORK_MODE	Read Only
SETUP.LAN.IP	Read Only
SETUP.LAN.MASK	Read Only
SETUP.LAN.GATEWAY	Read Only
SETUP.LAN.DNS1	Read Only
SETUP.LAN.DNS2	Read Only
SETUP.WIFI.ENABLE	Read Only
SETUP.WIFI.DISABLE_LAN_CONNECTED	Read Only

Register Name	Change
SETUP.WIFI.DISABLE_AFTER	Read Only
SETUP.WIFI.MODE	Read Only
SETUP.WIFI.AP_SSID	Read Only
SETUP.WIFI.AP_PASS	Read Only
SETUP.WIFI.STA_SSID	Read Only
SETUP.WIFI.STA_PASS	Read Only

1.1.8.3 Registers Removed (R3)

Register Name
VC-{VID}.VOLUME

1.1.9 Revision 2: API 1.2 Changes

- INC Command support for Input Gain
- Added Frequency parameter for SUBSCRIBE command
- Pink NoiseGenerator

1.1.9.1 Registers Added (R2)

Register Name
SETUP.DEVICE.SERIAL
SETUP.DEVICE.FIRMWARE
SETUP.DEVICE.FIRMWARE
SETUP.DEVICE.MAC
SETUP.DEVICE.WIFI_MAC
OUT-{OID}.LIMITER.AUTO
OUT-{OID}.LIMITER.THRESHOLD
OUT-{OID}.LIMITER.ATTACK
OUT-{OID}.LIMITER.RELEASE
OUT-{OID}.LIMITER.HOLD

1.1.9.2 Revision 1: Registers Added (R1)

Register Name
ZONE-{ZID}.COMPRESSOR.HOLD
OUT-{OID}.PRESET.NAME
OUT-{OID}.PRESET.ID
OUT-{OID}.PRESET.LOCKED
OUT-{OID}.POLARITY.PROTECTED
OUT-{OID}.OUTPUT_MODE.PROTECTED

Register Name
OUT-{OID}.SPEAKER_DELAY.PROTECTED
OUT-{OID}.LIMITER.PROTECTED
OUT-{OID}.SPEAKER_EQ.PROTECTED
OUT-{OID}.XR.PROTECTED
OUT-{OID}.FIR.PROTECTED
OUT-{OID}.PEAK_LIMITER.BYPASS
OUT-{OID}.PEAK_LIMITER.KNEE
OUT-{OID}.RMS_LIMITER.BYPASS
OUT-{OID}.RMS_LIMITER.THRESHOLD
OUT-{OID}.RMS_LIMITER.ATTACK
OUT-{OID}.RMS_LIMITER.RELEASE
OUT-{OID}.RMS_LIMITER.HOLD
OUT-{OID}.RMS_LIMITER.KNEE
OUT-{OID}.CLIP_LIMITER.BYPASS
OUT-{OID}.FIR.BYPASS
OUT-{OID}.FIR.TAPS

1.2 Connecting to the Amplifier

Out of the Box the amplifier is hard-coded with the Ethernet Address 192.168.64.100. It is also possible to connect to the amplifier using Wifi. Connect to the Wifi AP (SSID) and connect using the default IP address of 192.168.4.1.

1.3 Discovery (mDNS)

If the application requires the amplifier to have a dynamic IP address, it is possible to use mDNS to locate the amplifier.

The service type is: `_pasconnect._tcp`

The following properties is defined:

- **api_version** - the api version of the device
- **device_type** - the device type. For amplifiers this will always be `PasAmpControl`
- **model** - the model name of the device
- **software_id** - software id of the amplifier (Manufacturer and Model Specific)
- **hardware_id** - hardware id of the amplifier (Model ID)

Example (Avahi for Linux):

```
$> avahi-browse -t -r _pasconnect._tcp
+ enp0s8 IPv4 PASCAL-IP1252-2122-00031.local                    _pasconnect._tcp
 ↪  local
= enp0s8 IPv4 PASCAL-IP1252-2122-00031.local                    _pasconnect._tcp
 ↪  local
   hostname = [PASCAL-IP1252-2122-00031.local.local]
```

```
address = [192.168.64.100]
port = [80]
txt = ["api_version=1.5" "device_type=PasAmpControl" "manufacturer=Pascal Audio"
 ↪  "model=IP 125.2" "software_id=2" "model_id=2"]
```

1.4 Definitions

1.4.1 {IID} - Input Channels

The following input channels is defined for the amplifier.

- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **104** - Analog Input 5 *(8 channel version only)*
- **105** - Analog Input 6 *(8 channel version only)*
- **106** - Analog Input 7 *(8 channel version only)*
- **107** - Analog Input 8 *(8 channel version only)*
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1 *(Only for Dante Enabled Amplifiers)*
- **301** - Dante 2 *(Only for Dante Enabled Amplifiers)*
- **302** - Dante 3 *(Only for Dante Enabled Amplifiers)*
- **303** - Dante 4 *(Only for Dante Enabled Amplifiers)*
- **400** - Noise Generator

1.4.2 {MID} - Mix Channels

The following mix channels is defined for the amplifier.

- **1** - Mix 1
- **2** - Mix 2
- **3** - Mix 3 *(4 channel version only)*
- **4** - Mix 4 *(4 channel version only)*

1.4.3 {ZID} - Zones

The following zones is defined for the amplifier.

- **A** - Zone A
- **B** - Zone B
- **C** - Zone C *(4 channel version only)*
- **D** - Zone D *(4 channel version only)*

1.4.4 {ZP} - Zone Priorities

The following zone priority levels are defined for priority override.

- **2** - Priority 2 (Medium Priority)
- **3** - Priority 3 (Highest Priority)

1.4.5 {OID} - Output Channels

- **1** - Output 1
- **2** - Output 2
- **3** - Output 3 *(4 channel version only)*
- **4** - Output 4 *(4 channel version only)*

1.4.6 {RID} Output Route Channels

- **1** - Output 1
- **2** - Output 2
- **3** - Output 3 *(4 channel version only)*
- **4** - Output 4 *(4 channel version only)*

1.4.7 {SID} Input Source

The following input sources is defined for the amplifier.

- **0** - Unused Input (Silent)
- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1 *(Only for Dante Enabled Amplifiers)*
- **301** - Dante 2 *(Only for Dante Enabled Amplifiers)*
- **302** - Dante 3 *(Only for Dante Enabled Amplifiers)*
- **303** - Dante 4 *(Only for Dante Enabled Amplifiers)*
- **400** - Noise Generator
- **500** - Mix 1

- **501** - Mix 2
- **502** - Mix 3 *(Only for 4 Channel Amplifiers)*
- **503** - Mix 4 *(Only for 4 Channel Amplifiers)*

1.4.8 {RSID} Route Source

The following route sources is defined for the amplifier.

- **0** - Unused (Silent)
- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1
- **301** - Dante 2
- **302** - Dante 3
- **303** - Dante 4
- **500** - Mix A

- **501** - Mix B
- **502** - Mix C
- **503** - Mix D
- **1000** - Zone A
- **1001** - Zone B
- **1002** - Zone C
- **1003** - Zone D

1.4.9 {VID} Volume Controls

The following Volume Controls is defined for the amplifier.

- **0** - OFF
- **1** - GPIO PIN 4 Volume Control
- **2** - GPIO PIN 5 Volume Control
- **3** - GPIO PIN 6 Volume Control
- **4** - GPIO PIN 7 Volume Control

1.4.10 {EID} Input Equalizer Bands

The following Input Equalizer Bands (`IN-{IID}.EQ-{EID}.*`) are defined in the amplifier. The number of bands is model dependent; read the `IN.EQ.COUNT` register to discover the band count supported by a specific device. The current maximum is **5** bands.

- **1** - Input EQ Band 1
- **2** - Input EQ Band 2
- **3** - Input EQ Band 3
- **4** - Input EQ Band 4
- **5** - Input EQ Band 5

1.4.11 {OEID} Output Equalizer Bands

The following Output Equalizer Bands (`OUT-{OID}.EQ-{OEID}.*`) are defined in the amplifier. The number of bands is model dependent; read the `OUT.EQ.COUNT` register to discover the band count supported by a specific device. The current maximum is **10** bands.

- **1** - Output EQ Band 1
- **2** - Output EQ Band 2
- **3** - Output EQ Band 3
- **4** - Output EQ Band 4
- **5** - Output EQ Band 5
- **6** - Output EQ Band 6
- **7** - Output EQ Band 7
- **8** - Output EQ Band 8
- **9** - Output EQ Band 9
- **10** - Output EQ Band 10

1.4.12 {OSEID} Speaker Equalizer Bands

The following Speaker Equalizer Bands (`OUT-{OID}.SPEAKER_EQ-{OSEID}.*`) are defined in the amplifier. The number of bands is model dependent; read the `OUT.SPEAKER_EQ.COUNT` register to discover the band count supported by a specific device. The current maximum is **15** bands.

- **1** - Speaker EQ Band 1
- **2** - Speaker EQ Band 2
- **3** - Speaker EQ Band 3
- **4** - Speaker EQ Band 4
- **5** - Speaker EQ Band 5
- **6** - Speaker EQ Band 6
- **7** - Speaker EQ Band 7
- **8** - Speaker EQ Band 8
- **9** - Speaker EQ Band 9
- **10** - Speaker EQ Band 10
- **11** - Speaker EQ Band 11
- **12** - Speaker EQ Band 12
- **13** - Speaker EQ Band 13
- **14** - Speaker EQ Band 14
- **15** - Speaker EQ Band 15

1.4.13 Variable Types

- **Float** - Float format, delimited with '.'
- **Integer** - Normal integer
- **Enum** - Basically a string with a predefined set of options
- **String** - String. Might have limitations on number of characters. String values containing spaces must be enclosed in double-quotes.

2 API Endpoints

2.1 Raw Socket API

The Primary API in the amplifier is TCP Socket based (Port 7621) and is line based. That means every line is delimited by newline \n. Every line contains a single message. The API consists of 2 parts: a Command/Response interface and a Publish/Subscribe interface.

2.1.1 Ncat

Examples in documentation are based on Ncat (https://nmap.org/download.html). The specific syntax is PowerShell, but can easily be converted to bash for Linux.

Powershell style:

```
$> "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

bash style:

```
$> echo "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

2.2 WebSocket API

It is also possible to connect to the WebSocket based API in the amplifier. The syntax of commands and replies is exactly the same between the socket based API and the WebSocket based API, though a single WebSocket message might contain or return multiple lines of text, with each line containing a single message.

3 Command/Response

The Command/Response interface allows for Querying/Updating the registers in the amplifier and to execute commands.

To execute a command, send a websocket message with the command followed by newline.

- If the command executes successfully the response will be an asterisk followed by the command text.

```
$> "<COMMAND>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*<COMMAND>
```

- If the command fails the response is a single line beginning with a hash #, then the command that was sent (echoed verbatim), a pipe |, and a human-readable message: #<command>|<message>. The echoed command lets a client correlate the error with the request that caused it (useful when pipelining). See Error Handling for the full message catalog.

```
$> "GET IN-100.BOGUS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
#GET IN-100.BOGUS|E107: Unknown Parameter
```

- If the command returns data in form of registers the response will be:

```
$> "<COMMAND>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+<RESPONSE>
*<COMMAND>
```

3.1 Command Types

3.1.1 GET

Get value of amplifier register. The command supports wildcards.

Format:

```
"GET <REGISTER>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+<RESPONSE(s)>
*<COMMAND>
```

Example:

```
$> "GET IN-100.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.NAME "Analog 1"
*GET IN-100.NAME
```

```
"GET IN-*.NAME" | websocat -t -0 ws://192.168.64.100/ws
+IN-100.NAME "Analog 1"
+IN-101.NAME "Analog 2"
+IN-102.NAME "Analog 3"
+IN-103.NAME "Analog 4"
+IN-200.NAME "S/PDIF 1"
+IN-201.NAME "S/PDIF 1R"
+IN-400.NAME "Noise Generator"
*GET IN-*.NAME
```

3.1.2 SET

Set value in amplifier register. The command does not support wildcards!

Format:

```
"SET <REGISTER> <VALUE>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*<COMMAND>
```

Example

```
$> "SET IN-100.NAME ""Streamer""" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.NAME "Analog 1"
*SET IN-100.NAME "Streamer"
```

On a successful SET the firmware emits the changed register as a normal REG-topic update (`+<register> <value>`) to all REG subscribers, then sends the `*<command>` ack to the caller. A given client therefore sees the `+...` echo for its own SET only if it is itself subscribed to the REG topic; an unsubscribed client receives just the `*SET ...` ack. (For INC, by contrast, the caller is always sent the resulting `+<register> <value>` line before the ack.) See SUBSCRIBE for why `+...` lines and ack lines can interleave.

3.1.3 INC

Modifies the value in amplifier register by the amount specified in the command. The valus can be positive or negative. The command does not support wildcards!

Format:

```
"INC <REGISTER> <VALUE>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+<REGISTER> <MODIFIED VALUE>
*<COMMAND>
```

Example

```
$> "INC ZONE-A.GAIN -5 | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN -5.00
*INC ZONE-A.GAIN -5
```

3.1.4 SUBSCRIBE

Subscribe to changes in all registers and dynamics. The subscribe command does not support subscriptions to individual registers. *This might change in a later release*.

The register changes will stream to the websocket after subscription…

```
$> "SUBSCRIBE" | ncat 192.168.64.100 7621 --no-shutdown
...
+IN-100.DYN.SIGNAL -49.9777
+IN-100.DYN.CLIP 0
+IN-101.DYN.SIGNAL -49.3077
+IN-101.DYN.CLIP 0
+IN-102.DYN.SIGNAL -99.7209
+IN-102.DYN.CLIP 0
...
*SUBSCRIBE
```

3.1.5 SUBSCRIBE <BLANK|*|REG|DYN> <FREQ> {#sec:subscribe}

Subscription is organised into two topics (firmware `protocol_connection.cpp`):

- **REG** - Register-change updates only. Pushed whenever a register value changes (from any client, the web UI, GPIO, presets, etc.).

- **DYN** - Dynamic/telemetry updates only. This is the channel that carries the subscription-only metering registers (`IN-{IID}.DYN.SIGNAL/.CLIP`, `ZONE-{ZID}.DYN.SIGNAL`, `OUT-{OID}.DYN.SIGNAL`, `ROUT-{RID}.DYN.SIGNAL/.CLIP`), which are not GET-able and only appear over this topic.
- **"*"** or **BLANK** (argument omitted) - Subscribes to both REG and DYN.

<FREQ> is the maximum push rate in Hz (updates per second) for that topic, applied per-topic on the server side:

- `0` (the default when omitted) means unlimited: every change/sample is pushed as it happens.
- Otherwise the server enforces a minimum interval of `1 / FREQ` seconds between pushes for that topic. So `1` = at most 1 update/second, `0.5` = at most 1 update every 2 seconds, `2` = at most 1 update every 0.5 s.
- The accepted range is `0..1000`; values are clamped to that range.

Frequency limiting is most useful for the high-rate DYN metering. Re-issuing SUBSCRIBE for a topic replaces its frequency setting.

Subscribe to changes in all registers or dynamic updates. The subscribe command does not support subscriptions to individual registers. *This might change in a later release*.

> **Push lines look exactly like GET data lines.** Both a subscription push and a line in a GET reply are formatted as `+<register> <value>`. The only thing that delimits a *response* to a command is the trailing `*<command>` ack line; subscription pushes carry no such delimiter and can arrive at any time, interleaved with command responses. A robust client must therefore not assume that the next `+...` line it reads belongs to the command it just sent. It should track outstanding commands and treat any `+...` line that arrives before the matching `*<command>` (or an unsolicited `+...` between command exchanges) as subscription data. When a value is changed by another client, subscribers receive the resulting `+<register> <value>` line on the REG topic.

The register changes will stream to the socket/websocket after subscription…

Example: Subscribe to All updates

```
$> "SUBSCRIBE" | ncat 192.168.64.100 7621 --no-shutdown
...
+IN-100.DYN.SIGNAL -49.9777
+IN-100.DYN.CLIP 0
+IN-101.DYN.SIGNAL -49.3077
+IN-101.DYN.CLIP 0
+IN-102.DYN.SIGNAL -99.7209
+IN-102.DYN.CLIP 0
...
*SUBSCRIBE
```

Example: Subscribe to Register only updates

```
$> "SUBSCRIBE REG" | ncat 192.168.64.100 7621 --no-shutdown
...
+ZONE-A.GAIN -5.00
...
*SUBSCRIBE REG
```

Example: Subscribe to Dynamic updates, but limit frequency to 1 Hz

```
$> "SUBSCRIBE DYN 1" | ncat 192.168.64.100 7621 --no-shutdown
...
+IN-100.DYN.SIGNAL -49.9777
+IN-100.DYN.CLIP 0
+IN-101.DYN.SIGNAL -49.3077
+IN-101.DYN.CLIP 0
+IN-102.DYN.SIGNAL -99.7209
+IN-102.DYN.CLIP 0
...
*SUBSCRIBE DYN 1
```

3.1.6 UNSUBSCRIBE <BLANK|*|REG|DYN>

- **REG** - Register Updates Only
- **DYN** - Dynamic updates Only
- **"BLANK"** - IF EMPTY - Both dynamic and register updates

Unsubscribe from the previous subscription. The parameter must match a previous subscription. An UNSUBSCRIBE with a blank value (unsubscribe all) will not cancel a subscription made to register updates only.

3.1.7 POWER_ON

TYPE: Command

Methods: POWER_ON

Example:

```
$> "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

3.1.8 POWER_OFF

TYPE: Command

Methods: POWER_OFF

Example:

```
$> "POWER_OFF" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_OFF
```

POWER_ON/POWER_OFF are only valid in certain Power-On modes; in an incompatible mode the firmware returns `E122: Error - Power On/Off not supported for current PowerOn Mode`.

3.1.9 PING

TYPE: Command

Application-level keepalive. The amplifier immediately replies with a PONG data line and then the usual *PING ack. This is the recommended way to confirm that a raw TCP control session (port 7621) is still alive, since that socket has no idle timeout of its own (see Connection & Session Lifecycle).

```
$> "PING" | ncat 192.168.64.100 7621 --no-shutdown -i 1
PONG
*PING
```

3.1.10 REBOOT

TYPE: Command

Reboots the amplifier. The firmware acks the command and then schedules the restart a moment later (~1 s), so the connection drops shortly after the *REBOOT ack. Audio is interrupted.

```
$> "REBOOT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*REBOOT
```

3.1.11 COPY_EQ <IN|SPK|OUT> <SRC> <DEST>

TYPE: Command

Copies an entire equalizer block from one channel to another. The first argument selects which EQ block:

- **IN** - input EQ; <SRC>/<DEST> are input channel IDs (e.g. `100`, `101`).
- **SPK** - speaker (output) EQ; <SRC>/<DEST> are 1-based output indices.
- **OUT** - output EQ; <SRC>/<DEST> are 1-based output indices.

On success the affected destination registers are returned (and pushed to REG subscribers). Copying onto a locked speaker preset fails with `E118: Setting is Protected Speaker Preset`; an unknown direction yields `E104: Invalid Parameter Value`.

```
$> "COPY_EQ IN 100 101" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-101.EQ.BYPASS 0
...
*COPY_EQ IN 100 101
```

3.1.12 RESET_EQ <IN|SPK|OUT> <DEST>

TYPE: Command

Resets the equalizer block on the destination channel to its defaults (flat). Direction and indexing match COPY_EQ.

```
$> "RESET_EQ OUT 1" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ.BYPASS 0
...
*RESET_EQ OUT 1
```

3.1.13 COPY_XR <SRC> <DEST>

TYPE: Command

Copies the crossover (XR) settings from one output to another. <SRC>/<DEST> are 1-based output indices. Copying onto a preset with a locked crossover fails with `E118`.

```
$> "COPY_XR 1 2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-2.XR.HPF_TYPE ...
...
*COPY_XR 1 2
```

3.1.14 RESET_XR <DEST>

TYPE: Command

Resets the crossover on the destination output (1-based index) to its defaults.

```
$> "RESET_XR 2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-2.XR.HPF_TYPE ...
...
*RESET_XR 2
```

3.1.15 FACTORY_DEFAULTS

TYPE: Command

Restores the entire device setup to factory defaults, re-applies it to the DSP and (if present) resets Dante, then re-applies network configuration. This closes every other open control connection (the connection that issued the command is kept and receives the full post-reset register dump); other clients must reconnect. Use with care.

```
$> "FACTORY_DEFAULTS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+...full register dump...
*FACTORY_DEFAULTS
```

3.2 Registers

Supported Registers for General Use

3.2.1 Base Registers

Register Name	Type	Access	Unit	Range / Values
API_VERSION	String	Get		
SYSTEM.STATUS.STATE	Enum	Get		{INIT, STANDBY, STANDBY_FORCED, ON}
SYSTEM.STATUS.SIGNAL_IN	Enum	Get		{OFF, NO_SIGNAL, SIGNAL, CLIP}
SYSTEM.STATUS.SIGNAL_OUT	Enum	Get		{OFF, NO_SIGNAL, SIGNAL, CLIP, FAULT}
SYSTEM.STATUS.LAN	String	Get		
SYSTEM.STATUS.WIFI	String	Get		

3.2.2 Device Information Registers

Register Name	Type	Access	Unit	Range / Values
SYSTEM.DEVICE.SWID	Integer	Get		
SYSTEM.DEVICE.HWID	Integer	Get		
SYSTEM.DEVICE.VENDOR_NAME	String[32]	Get		Max 32
SYSTEM.DEVICE.MODEL_NAME	String[32]	Get		Max 32
SYSTEM.DEVICE.SERIAL	String	Get		
SYSTEM.DEVICE.FIRMWARE	String	Get		
SYSTEM.DEVICE.FIRMWARE_DATE	String	Get		
SYSTEM.DEVICE.MAC	String	Get		
SYSTEM.DEVICE.WIFI_MAC	String	Get		

3.2.3 System Information Registers

Register Name	Type	Access	Unit	Range / Values
SETUP.SYSTEM.DEVICE_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.VENUE_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.CUSTOMER_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.ASSET_TAG	String[32]	Get, Set		Max 32

Register Name	Type	Access	Unit	Range / Values
SETUP.SYSTEM.INSTALLER_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.CONTACT_INFO	String[32]	Get, Set		Max 32
SETUP.SYSTEM.INSTALL_DATE	String[32]	Get, Set		Max 32
SETUP.SYSTEM.INSTALL_NOTES	String[256]	Get, Set		Max 256
SETUP.SYSTEM.LOCATING	Boolean	Get, Set		
SETUP.SYSTEM.CUSTOM1	String[8192]	Get, Set		Max 8192
SETUP.SYSTEM.CUSTOM2	String[8192]	Get, Set		Max 8192
SETUP.SYSTEM.CUSTOM3	String[8192]	Get, Set		Max 8192

3.2.4 Generator Registers

Register Name	Type	Access	Unit	Range / Values
GENERATOR.TYPE	Enum	Get, Set		{PINK, SINE}
GENERATOR.SINE.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
GENERATOR.PINK.LPF_ENABLE	Boolean	Get, Set		
GENERATOR.PINK.LPF_FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
GENERATOR.PINK.HPF_ENABLE	Boolean	Get, Set		
GENERATOR.PINK.HPF_FREQ	Float	Get, Set	Hz	[20.0, 20000.0]

3.2.5 Input Registers

Register Name	Type	Access	Unit	Range / Values
IN.COUNT	Integer	Get		
IN-{IID}.NAME	String[32]	Get, Set		Max 32
IN-{IID}.SENS	Enum	Get, Set		{14DBU, 4DBU, -10DBV, MIC}
IN-{IID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
IN-{IID}.STEREO	Boolean	Get, Set		
IN-{IID}.HPF_ENABLE	Boolean	Get, Set		
IN-{IID}.DYN.SIGNAL	Float	Subscribe	dB	
IN-{IID}.DYN.CLIP	Boolean	Subscribe		

3.2.6 Input Eq Registers

Register Name	Type	Access	Unit	Range / Values
IN.EQ.COUNT	Integer	Get		
IN-{IID}.EQ.BYPASS	Boolean	Get, Set		
IN-{IID}.EQ-{EID}.TYPE	Enum	Get, Set		{PARAMETRIC, LOWPASS_12, HIGHPASS_12, LOW_SHELF_Q, HIGH_SHELF_Q}
IN-{IID}.EQ-{EID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
IN-{IID}.EQ-{EID}.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
IN-{IID}.EQ-{EID}.Q	Float	Get, Set		[0.4, 30.0]

Register Name	Type	Access	Unit	Range / Values
IN-{IID}.EQ-{EID}.BYPASS	Boolean	Get, Set		

3.2.7 Zone Registers

Register Name	Type	Access	Unit	Range / Values
ZONE.COUNT	Integer	Get		
ZONE-{ZID}.NAME	String[32]	Get, Set		Max 32
ZONE-{ZID}.PRIMARY_SRC	Integer	Get, Set		Valid {SID}
ZONE-{ZID}.PRIORITY_SRC	Integer	Get, Set		Valid {SID}
ZONE-{ZID}.GAIN	Float	Get, Set, Inc	dB	[-80.0, 0.0]
ZONE-{ZID}.GAIN_MIN	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.GAIN_MAX	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.STEREO	Boolean	Get, Set		
ZONE-{ZID}.GPIO_VC	Integer	Get, Set		Valid {VID}, 0 = OFF
ZONE-{ZID}.MUTE	Boolean	Get, Set		
ZONE-{ZID}.MUTE_ENABLE	Boolean	Get, Set		
ZONE-{ZID}.SRC-{IID}.ENABLED	Boolean	Get, Set		
ZONE-{ZID}.DYN.SIGNAL	Float	Subscribe	dB	[-144.0, 20.0]

3.2.8 Zone Ducker Registers

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.DUCK.MODE	Enum	Get, Set		{OFF, DUCKER, OVERRIDE}
ZONE-{ZID}.DUCK.AUTO	Boolean	Get, Set		
ZONE-{ZID}.DUCK.THRESHOLD	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.DUCK.DEPTH	Float	Get, Set	dB	[-144.0, 0.0]
ZONE-{ZID}.DUCK.ATTACK	Float	Get, Set	Sec	[0.001, 0.2]
ZONE-{ZID}.DUCK.RELEASE	Float	Get, Set	Sec	[0.01, 10.0]
ZONE-{ZID}.DUCK.HOLD	Float	Get, Set	Sec	[0.0, 10.0]
ZONE-{ZID}.DUCK.OVERRIDE_GAIN	Float	Get, Set	dB	[-144.0, 15.0]
ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE	Boolean	Get, Set		

3.2.9 Zone Priority Registers

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.PRIORITY-{ZP}.SRC	Integer	Get, Set		Valid {SID}
ZONE-{ZID}.PRIORITY-{ZP}.AUTO	Boolean	Get, Set		

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.PRIORITY-{ZP}.THRESHOLD	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.PRIORITY-{ZP}.HOLD	Float	Get, Set	Sec	[0.001, 10.0]
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN	Float	Get, Set	dB	[-144.0, 15.0]
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN_ENABLE	Boolean	Get, Set		

3.2.10 Zone Compressor Registers

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.COMPRESSOR.AUTO	Boolean	Get, Set		
ZONE-{ZID}.COMPRESSOR.THRESHOLD	Float	Get, Set	dB	[-40.0, 20.0]
ZONE-{ZID}.COMPRESSOR.ATTACK	Float	Get, Set	Sec	[0.0003, 0.05]
ZONE-{ZID}.COMPRESSOR.RELEASE	Float	Get, Set	Sec	[0.001, 1.0]
ZONE-{ZID}.COMPRESSOR.HOLD	Float	Get, Set	Sec	[0.0, 1.0]
ZONE-{ZID}.COMPRESSOR.RATIO	Float	Get, Set		[1.0, 50.0]
ZONE-{ZID}.COMPRESSOR.KNEE	Float	Get, Set	dB	[0.0, 12.0]
ZONE-{ZID}.COMPRESSOR.BYPASS	Boolean	Get, Set		

3.2.11 Output Registers

Register Name	Type	Access	Unit	Range / Values
OUT.COUNT	Integer	Get		
OUT-{OID}.NAME	String[32]	Get, Set		Max 32
OUT-{OID}.GAIN	Float	Get, Set, Inc	dB	[-30.0, 15.0]
OUT-{OID}.MUTE	Boolean	Get, Set		
OUT-{OID}.SRC	String[1]	Get, Set		Max 1
OUT-{OID}.SRC_CHANNEL	Enum	Get, Set		{L, R, S}
OUT-{OID}.POLARITY	Integer	Get, Set		{-1, 1}
OUT-{OID}.OUTPUT_MODE	Enum	Get, Set		{OFF, 4R, 8R, 70V, 100V, BTL}
OUT-{OID}.OUTPUT_HIGHPASS	Float	Get, Set	Hz	[0.0, 1000.0]
OUT-{OID}.DYN.SIGNAL	Float	Subscribe	dB	[-144.0, 20.0]
OUT-{OID}.DYN.CLIP	Boolean	Subscribe		

3.2.12 Output Delay Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.DELAY.TIME	Float	Get, Set	Sec	[0.0, 0.1]
OUT-{OID}.DELAY.BYPASS	Boolean	Get, Set		

3.3 Output Speaker Processing Registers

The following registers are available for advanced configuration and speaker processing. Many of these are automatically configured by speaker presets and should not be modified unless necessary.

Speaker presets define the DSP configuration for specific speakers. These registers are read-only and indicate the current preset status.

3.3.1 Output Preset Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.PRESET.NAME	String[64]	Get		Max 64
OUT-{OID}.PRESET.ID	String	Get		
OUT-{OID}.PRESET.LOCKED	Boolean	Get		
OUT-{OID}.PRESET.CUSTOMIZED	Boolean	Get		

Speaker delay provides time alignment for speaker arrays. May be protected by preset.

3.3.2 Output Speaker Delay Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.SPEAKER_DELAY.TIME	Float	Get, Set	Sec	[0.0, 0.01]
OUT-{OID}.SPEAKER_DELAY.BYPASS	Boolean	Get, Set		
OUT-{OID}.SPEAKER_DELAY.PROTECTED	Boolean	Get		

Peak limiter protects speakers from transient peaks.

3.3.3 Output Peak Limiter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.PEAK_LIMITER.BYPASS	Boolean	Get, Set		
OUT-{OID}.PEAK_LIMITER.AUTO	Boolean	Get, Set		
OUT-{OID}.PEAK_LIMITER.THRESHOLD	Float	Get, Set	Vpeak	[1.0, 200.0]

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.PEAK_LIMITER.ATTACK	Float	Get, Set	Sec	[0.0003, 0.1]
OUT-{OID}.PEAK_LIMITER.RELEASE	Float	Get, Set	Sec	[0.004, 2.0]
OUT-{OID}.PEAK_LIMITER.HOLD	Float	Get, Set	Sec	[0.0, 1.0]
OUT-{OID}.PEAK_LIMITER.KNEE	Float	Get, Set	dB	[0.0, 6.0]

RMS limiter protects speakers from sustained thermal damage.

3.3.4 Output RMS Limiter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.RMS_LIMITER.BYPASS	Boolean	Get, Set		
OUT-{OID}.RMS_LIMITER.THRESHOLD	Float	Get, Set	Vpeak	[1.0, 150.0]
OUT-{OID}.RMS_LIMITER.ATTACK	Float	Get, Set	Sec	[0.01, 30.0]
OUT-{OID}.RMS_LIMITER.RELEASE	Float	Get, Set	Sec	[0.01, 30.0]
OUT-{OID}.RMS_LIMITER.HOLD	Float	Get, Set	Sec	[0.0, 1.0]
OUT-{OID}.RMS_LIMITER.KNEE	Float	Get, Set	dB	[0.0, 6.0]

Clip limiter prevents hard clipping at the amplifier output stage.

3.3.5 Output Clip Limiter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.CLIP_LIMITER.BYPASS	Boolean	Get, Set		
OUT-{OID}.CLIP_LIMITER.MODE	Enum	Get, Set		{NORMAL, FAST}
OUT-{OID}.LIMITER.PROTECTED	Boolean	Get		

User-adjustable output EQ for system tuning.

3.3.6 Output EQ Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.EQ.BYPASS	Boolean	Get, Set		

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.EQ-{OEID}.TYPE	Enum	Get, Set		{PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}
OUT-{OID}.EQ-{OEID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
OUT-{OID}.EQ-{OEID}.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.EQ-{OEID}.Q	Float	Get, Set		[0.4, 30.0]
OUT-{OID}.EQ-{OEID}.BYPASS	Boolean	Get, Set		

Factory-configured speaker EQ. Usually protected by preset.

3.3.7 Output Speaker EQ Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.SPEAKER_EQ.BYPASS	Boolean	Get, Set		
OUT-{OID}.SPEAKER_EQ-{OSEID}.TYPE	Enum	Get, Set		{PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}
OUT-{OID}.SPEAKER_EQ-{OSEID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
OUT-{OID}.SPEAKER_EQ-{OSEID}.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.SPEAKER_EQ-{OSEID}.Q	Float	Get, Set		[0.4, 30.0]
OUT-{OID}.SPEAKER_EQ-{OSEID}.BYPASS	Boolean	Get, Set		
OUT-{OID}.SPEAKER_EQ.PROTECTED	Boolean	Get		

Crossover filters for multi-way speaker systems. Usually protected by preset.

3.3.8 Output Crossover (XR) Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.XR.BYPASS	Boolean	Get, Set		
OUT-{OID}.XR.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.XR.LOWPASS_TYPE	Enum	Get, Set		{OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}
OUT-{OID}.XR.LOWPASS_FREQUENCY	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.XR.HIGHPASS_TYPE	Enum	Get, Set		{OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}
OUT-{OID}.XR.HIGHPASS_FREQUENCY	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.XR.PROTECTED	Boolean	Get		

FIR filters for advanced speaker processing. Usually protected by preset.

3.3.9 Output FIR Filter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.FIR.BYPASS	Boolean	Get, Set		
OUT-{OID}.FIR.TAPS	Integer	Get		
OUT-{OID}.FIR.PROTECTED	Boolean	Get		

Direct routing for sending zone audio to specific outputs (advanced use).

3.3.10 Routing Registers

Register Name	Type	Access	Unit	Range / Values
ROUT-{RID}.SRC	Integer	Get, Set		Valid {RSID}
ROUT-{RID}.SRC_CHANNEL	Enum	Get, Set		{L, R, S}
ROUT-{RID}.GAIN	Float	Get, Set	dB	[-40.0, 20.0]
ROUT-{RID}.DYN.SIGNAL	Float	Subscribe	dB	
ROUT-{RID}.DYN.CLIP	Boolean	Subscribe		

External volume control inputs (GPIO wall plates).

3.3.11 Volume Control Registers

Register Name	Type	Access	Unit	Range / Values
VC.COUNT	Integer	Get		
VC-{VID}.NAME	String[32]	Get		Max 32
VC-{VID}.VALUE	Float	Get	Percent	[0.0, 100.0]

3.3.12 Power Management Registers

Register Name	Type	Access	Unit	Range / Values
SETUP.POWER.POWER_ON	Enum	Get, Set		{AUDIO, AUDIO_ECO, AUDIO_DSP, TRIGGER, TRIGGER_ECO, NETWORK}
SETUP.POWER.MUTE_TIME	Integer	Get, Set	Sec	[0, 3600]
SETUP.POWER.STANDBY_TIME	Integer	Get, Set	Sec	[0, 3600]

GPIO pins can be configured for various functions (status indication, triggers). The allowed values are per-pin; the Range / Values column lists each pin's accepted functions.

3.3.13 GPIO Configuration Registers

Register Name	Type	Access	Unit	Range / Values
SETUP.GPIO.PIN2	Enum	Get, Set		{OFF, STANDBY_NO, STANDBY_NC, MUTE_NO, MUTE_NC}
SETUP.GPIO.PIN4	Enum	Get, Set		{OFF, VOLUME_CONTROL}
SETUP.GPIO.PIN5	Enum	Get, Set		{OFF, VOLUME_CONTROL}
SETUP.GPIO.PIN6	Enum	Get, Set		{OFF, VOLUME_CONTROL, TRIGGER_12V_IN}
SETUP.GPIO.PIN7	Enum	Get, Set		{OFF, VOLUME_CONTROL, TRIGGER_12V_OUT}
SETUP.GPIO.PIN8	Enum	Get, Set		{VCC_3V3}

4 Wire-Protocol Reference

This chapter documents low-level behaviour of the control protocol that a third-party client must handle to be robust. Everything here is taken from the amplifier firmware.

4.1 Connection & Session Lifecycle

Two transports expose the identical line-based command/response protocol:

- **Raw TCP control socket** on port 7621. Plain TCP, line-based, each message terminated by a single newline `\n`. There is no idle timeout on this socket; a session stays open until the client closes it or the device reboots. Use PING/PONG to verify liveness.
- **WebSocket** at `ws://<ip>/ws` served on port 80 (the same HTTP server that serves the web UI). Commands and replies are byte-identical to the TCP socket, except a single WebSocket message may carry several `\n`-separated lines. The server sends a WebSocket PING control frame about every 3 seconds and applies a ~10-second idle timeout: if no frame is received within the timeout the server closes the connection. WebSocket clients must answer PINGs (most libraries auto-reply with PONG) or otherwise send traffic to stay connected.

Limits enforced by both transports (firmware `webserver.cpp`):

- The maximum received line length is 8192 + 128 bytes (`SOCKET_MAX_RX_LENGTH`). Two things protect against oversize input: the protocol engine rejects a command longer than that with `#<command>|Command too long`, and the session layer drops the whole connection if the unparsed receive buffer would exceed the limit (e.g. a line with no terminating newline). Keep individual lines well under 8 KB.
- The outgoing queue depth is 64 messages (`SOCKET_MAX_QUEUE_LENGTH`). If a client does not read fast enough and the server's per-connection TX queue fills, the server enters an overflow state, drops messages, and emits an overflow notice. The exact bytes differ by transport: the TCP socket sends `#|E0: TXQueue`

`Overflow – Messages dropped` and the WebSocket sends `#E0: TXQueue Overflow – Messages dropped`. The overflow state clears once the queue drains below half. A client that subscribes at high DYN rates must drain its socket promptly, or cap the rate with the <FREQ> argument.

4.2 Discovery & HTTP Endpoints

The TXT records and basic mDNS usage are covered under Discovery (mDNS). The driver-relevant detail: the advertised port is 80 (the HTTP/WebSocket port), and the amplifier does not advertise the 7621 control socket over mDNS. Three service types are registered (firmware `network_manager.cpp`): `_pasconnect._tcp` (the primary discovery service), `_http._tcp`, and `_rti._tcp` (for RTI controller integration, not needed by general API clients), all on port 80. A client should discover the device via `_pasconnect._tcp`, read its address, and then open the 7621 control socket (or the `/ws` WebSocket) by convention.

A small JSON-RPC 2.0 endpoint also exists at `POST /jrpc` on port 80. It exposes a single method, `reset_setup`, taking one boolean parameter `network` (whether to also reset the network configuration). It is used by the web UI; the line-based protocol described above is the intended integration surface for third-party clients.

4.3 Error Handling

A failed command produces exactly one line: `#<command>|<message>`, where <command> is the request echoed back verbatim and <message> is human-readable. Successful commands instead end with `*<command>`, and data is returned on `+...` lines (see Command/Response).

Most error messages begin with a stable `Exxx` code, but a client should treat the whole message string as the source of truth and not key solely on the number, because:

- some numbers are skipped (there is no `E113`, `E115`-`E117`, or `E125`);
- some numbers are reused for different conditions: `E110` covers both *Invalid IP address* and *Invalid Netmask address*, and `E126` covers both *Setting is not supported for non primary channel* and *Volume is controlled by GPIO volume control*;
- a few engine-level failures are reported without an `Exxx` code (see below).

4.3.1 Error Codes

The numbered codes (verbatim from firmware `protocol_error.h`):

Code	Message
E100	Channel is inactive (stereo)
E101	Invalid channel
E102	Invalid EQ band
E103	Invalid mix source
E104	Invalid Parameter Value
E105	Invalid number of arguments
E106	Unknown command
E107	Unknown Parameter
E108	No getter for parameter
E109	No setter for parameter
E110	Invalid IP address
E110	Invalid Netmask address
E111	No free slots

Code	Message
E112	Internal Error
E114	Invalid Password *(defined but never emitted - see Authentication)*
E118	Setting is Protected Speaker Preset
E119	String is too long
E120	String is invalid
E121	Limiter Error - Release must have higher value than attack
E122	Error - Power On/Off not supported for current PowerOn Mode
E123	Error - Limiter settings disabled due to Output Mode setting
E124	No INC for parameter
E126	Setting is not supported for non primary channel.
E126	Volume is controlled by GPIO volume control.
E127	No password is set.
E128	Mute disabled.
E129	Cannot Disable Primary Input.
E130	Source is Disabled.
E131	String is empty

In addition, the protocol engine and session layer can emit messages without an `Exxx` code (firmware `pas_protocol.cpp` / `webserver.cpp`):

Message	Meaning
Command Not Recognized	First token is not a known verb
Command too long	Command exceeded the 8192 + 128 byte limit
Parse Error	The line could not be tokenised (e.g. an unterminated quoted string)
Command Failed	The verb ran but reported failure (e.g. a register was not found)
E0: TXQueue Overflow - Messages dropped	Server TX queue overflowed; some pushes were dropped (delivered as `# | E0: ...` on the TCP socket and `#E0: ...` over WebSocket)

4.4 Value Encoding

Values on the wire (firmware `pas_protocol.cpp`, `protocol_helpers.h`) follow these rules; a client should parse leniently and emit canonically:

- **Booleans** are *returned* as `0` or `1`. On SET, the firmware also accepts the upper-case tokens `T`, `F`, `TRUE`, `FALSE` in addition to `1`/`0`. Lower-case forms are not accepted.
- **Floats** are formatted per-register with varying precision (anywhere from `%.1f` to `%0.5f` depending on the register), so the same numeric quantity may appear as `-5.00`, `1000.000`, or `0.50000`. Parse floats generically rather than assuming a fixed number of decimals; when sending, a plain decimal string is fine.
- **Strings** are *returned* double-quoted, e.g. `+IN-100.NAME "Analog 1"`. On SET, quotes are required only when the value contains spaces (the tokenizer otherwise splits on spaces); a single-word value may be sent unquoted. Values are UTF-8 and validated as such (invalid UTF-8 -> `E120`); over-length strings -> `E119`; an empty string where one is not allowed -> `E131`. There is no escape mechanism for a double-quote embedded inside a quoted string, so values cannot contain a `"` character.
- **Enums** are matched case-sensitively and exactly against the documented option set; an unrecognised token yields `E104`.

- Index-typed values are 1-based on the wire but 0-based internally. The firmware adds 1 when reporting an index value and subtracts 1 when parsing one (`add_index_response` / `get_value_as_index` / `parse_index`). So for an index register such as `ZONE.{ZID}.GPIO_VC`, the value the device reports and the value you send are the human-facing 1-based number; there is no need to compensate, but be aware the value is not a raw array offset and that 0-based assumptions will be off by one.

4.5 Authentication

The control protocol is unauthenticated. The registers `SYSTEM.SECURITY.PASSWORD_ENABLE` and `SYSTEM.SECURITY.PASSWORD_HASH` exist and the firmware stores them, but they are advisory only: the line-based protocol does not verify any password and gates no command behind one. The `E114: Invalid Password` code is defined in the firmware but is never thrown anywhere in the protocol path. Setting `PASSWORD_ENABLE` to true while no hash is stored returns `E127: No password is set.`, but this only guards the flag itself; it does not lock the socket. Treat any device reachable on port 7621 (or `/ws`) as fully controllable; enforce access control at the network layer.

5 Register Reference

5.1 API_VERSION

TYPE: Register

METHODS: Get

VALUES: [String]

API version string

Example:

```
$> "GET API_VERSION" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+API_VERSION "5.3"
*GET API_VERSION
```

5.2 GENERATOR.PINK.HPF_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable high-pass filter on pink noise

Example:

```
$> "GET GENERATOR.PINK.HPF_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.HPF_ENABLE 1
*GET GENERATOR.PINK.HPF_ENABLE

$> "SET GENERATOR.PINK.HPF_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.HPF_ENABLE 0
```

5.3 GENERATOR.PINK.HPF_FREQ

TYPE: Register

METHODS: Get, Set

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Pink noise high-pass filter frequency

Example:

```
$> "GET GENERATOR.PINK.HPF_FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.HPF_FREQ 1000.0
*GET GENERATOR.PINK.HPF_FREQ

$> "SET GENERATOR.PINK.HPF_FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.HPF_FREQ 2000.0
```

5.4 GENERATOR.PINK.LPF_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable low-pass filter on pink noise

Example:

```
$> "GET GENERATOR.PINK.LPF_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.LPF_ENABLE 1
*GET GENERATOR.PINK.LPF_ENABLE

$> "SET GENERATOR.PINK.LPF_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.LPF_ENABLE 0
```

5.5 GENERATOR.PINK.LPF_FREQ

TYPE: Register

METHODS: Get, Set

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Pink noise low-pass filter frequency

Example:

```
$> "GET GENERATOR.PINK.LPF_FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.LPF_FREQ 1000.0
*GET GENERATOR.PINK.LPF_FREQ

$> "SET GENERATOR.PINK.LPF_FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.LPF_FREQ 2000.0
```

5.6 GENERATOR.SINE.FREQ

TYPE: Register

METHODS: Get, Set

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Sine generator frequency

Example:

```
$> "GET GENERATOR.SINE.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.SINE.FREQ 1000.0
*GET GENERATOR.SINE.FREQ

$> "SET GENERATOR.SINE.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.SINE.FREQ 2000.0
```

5.7 GENERATOR.TYPE

TYPE: Register

METHODS: Get, Set

VALUES: Enum:

- **PINK** - Pink noise generator
- **SINE** - Sine wave generator

Generator type

Example:

```
$> "GET GENERATOR.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.TYPE PINK
*GET GENERATOR.TYPE

$> "SET GENERATOR.TYPE SINE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.TYPE SINE
```

5.8 IN-{IID}.DYN.CLIP

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Input clip indicator (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+IN-100.DYN.CLIP 1
```

5.9 IN-{IID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Float] (dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Input signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+IN-100.DYN.SIGNAL -12.00
```

5.10 IN-{IID}.EQ-{EID}.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Boolean]

Bypass individual EQ band

Example:

```
$> "GET IN-100.EQ-1.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.BYPASS 1
*GET IN-100.EQ-1.BYPASS

$> "SET IN-100.EQ-1.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.BYPASS 0
```

5.11 IN-{IID}.EQ-{EID}.FREQ

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

EQ band center frequency

Example:

```
$> "GET IN-100.EQ-1.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.FREQ 1000.0
*GET IN-100.EQ-1.FREQ

$> "SET IN-100.EQ-1.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.FREQ 2000.0
```

5.12 IN-{IID}.EQ-{EID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Float] (Range: -15.0 to 15.0 dB)

EQ band gain

Example:

```
$> "GET IN-100.EQ-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.GAIN -6.00
*GET IN-100.EQ-1.GAIN

$> "SET IN-100.EQ-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.GAIN -12.00
```

5.13 IN-{IID}.EQ-{EID}.Q

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Float] (Range: 0.4 to 30.0)

NOTES: Max Q 30 for PARAMETRIC/LOWPASS/HIGHPASS/NOTCH; 10 for shelf types

EQ band Q factor

Example:

```
$> "GET IN-100.EQ-1.Q" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.Q 1.0
*GET IN-100.EQ-1.Q

$> "SET IN-100.EQ-1.Q 2.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.Q 2.0
```

5.14 IN-{IID}.EQ-{EID}.TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: Enum:

- **PARAMETRIC** - Parametric EQ
- **LOWPASS_12** - 12 dB/octave low-pass filter
- **HIGHPASS_12** - 12 dB/octave high-pass filter
- **LOW_SHELF_Q** - Low shelf with Q control
- **HIGH_SHELF_Q** - High shelf with Q control

EQ band filter type

Example:

```
$> "GET IN-100.EQ-1.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.TYPE PARAMETRIC
*GET IN-100.EQ-1.TYPE

$> "SET IN-100.EQ-1.TYPE LOWPASS_12" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.TYPE LOWPASS_12
```

5.15 IN-{IID}.EQ.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

Bypass input EQ

Example:

```
$> "GET IN-100.EQ.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ.BYPASS 1
*GET IN-100.EQ.BYPASS

$> "SET IN-100.EQ.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ.BYPASS 0
```

5.16 IN-{IID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Float] (Range: -15.0 to 15.0 dB)

NOTES: Range is [-48.0, 0.0] for the Noise Generator input (IID 400)

Input channel gain

Example:

```
$> "GET IN-100.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.GAIN -6.00
*GET IN-100.GAIN

$> "SET IN-100.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.GAIN -12.00
```

5.17 IN-{IID}.HPF_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

High-pass filter enable

Example:

```
$> "GET IN-100.HPF_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.HPF_ENABLE 1
*GET IN-100.HPF_ENABLE

$> "SET IN-100.HPF_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.HPF_ENABLE 0
```

5.18 IN-{IID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [String] (Max Length 32 chars)

Input channel name

Example:

```
$> "GET IN-100.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.NAME "Mic 1"
*GET IN-100.NAME

$> "SET IN-100.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.NAME "New Name"
```

5.19 IN-{IID}.SENS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: Enum:

- **14DBU** - +14 dBu professional line level
- **4DBU** - +4 dBu professional line level
- **-10DBV** - -10 dBV consumer line level
- **MIC** - Microphone level input

Input sensitivity

Example:

```
$> "GET IN-100.SENS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.SENS 14DBU
*GET IN-100.SENS

$> "SET IN-100.SENS 4DBU" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.SENS 4DBU
```

5.20 IN-{IID}.STEREO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

Stereo link with next channel

Example:

```
$> "GET IN-100.STEREO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.STEREO 1
*GET IN-100.STEREO

$> "SET IN-100.STEREO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.STEREO 0
```

5.21 IN.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of input channels

Example:

```
$> "GET IN.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN.COUNT 4
*GET IN.COUNT
```

5.22 IN.EQ.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of EQ bands per input

Example:

```
$> "GET IN.EQ.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN.EQ.COUNT 4
*GET IN.EQ.COUNT
```

5.23 MIX-{MID}.GAIN-{IID}

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{MID}**: See paragraph Mix Channels
- **{IID}**: See paragraph Input Channels

VALUES: [Float] (Range: -144.0 to 0.0 dB)

Mix input gain for specific input channel

Example:

```
$> "GET MIX-1.GAIN-100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+MIX-1.GAIN-100 -6.00
*GET MIX-1.GAIN-100

$> "SET MIX-1.GAIN-100 -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET MIX-1.GAIN-100 -12.00
```

5.24 MIX-{MID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{MID}**: See paragraph Mix Channels

VALUES: [String] (Max Length 32 chars)

Mix channel name

Example:

```
$> "GET MIX-1.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+MIX-1.NAME "Mix A"
*GET MIX-1.NAME

$> "SET MIX-1.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET MIX-1.NAME "New Name"
```

5.25 MIX.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of mix channels

Example:

```
$> "GET MIX.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+MIX.COUNT 4
*GET MIX.COUNT
```

5.26 OUT-{OID}.CLIP_LIMITER.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass clip limiter

Example:

```
$> "GET OUT-1.CLIP_LIMITER.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.CLIP_LIMITER.BYPASS 1
*GET OUT-1.CLIP_LIMITER.BYPASS

$> "SET OUT-1.CLIP_LIMITER.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.CLIP_LIMITER.BYPASS 0
```

5.27 OUT-{OID}.CLIP_LIMITER.MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum {NORMAL, FAST}

Clip limiter mode

Example:

```
$> "GET OUT-1.CLIP_LIMITER.MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.CLIP_LIMITER.MODE NORMAL
*GET OUT-1.CLIP_LIMITER.MODE

$> "SET OUT-1.CLIP_LIMITER.MODE FAST" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.CLIP_LIMITER.MODE FAST
```

5.28 OUT-{OID}.DELAY.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass output delay

Example:

```
$> "GET OUT-1.DELAY.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.DELAY.BYPASS 1
*GET OUT-1.DELAY.BYPASS

$> "SET OUT-1.DELAY.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.DELAY.BYPASS 0
```

5.29 OUT-{OID}.DELAY.TIME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 0.1 Sec)

Output delay time

Example:

```
$> "GET OUT-1.DELAY.TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.DELAY.TIME 0.05
*GET OUT-1.DELAY.TIME

$> "SET OUT-1.DELAY.TIME 0.05" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.DELAY.TIME 0.05
```

5.30 OUT-{OID}.DYN.CLIP

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

NOTES: Planned — present in firmware but not yet streamed (commented out at system_manager.cpp:1276)

Output clip indicator (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+OUT-1.DYN.CLIP 1
```

5.31 OUT-{OID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: -144.0 to 20.0 dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Output signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+OUT-1.DYN.SIGNAL -62.00
```

5.32 OUT-{OID}.EQ-{OEID}.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Boolean]

Bypass individual output EQ band

Example:

```
$> "GET OUT-1.EQ-1.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.BYPASS 1
*GET OUT-1.EQ-1.BYPASS

$> "SET OUT-1.EQ-1.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.BYPASS 0
```

5.33 OUT-{OID}.EQ-{OEID}.FREQ

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Output EQ band center frequency

Example:

```
$> "GET OUT-1.EQ-1.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.FREQ 1000.0
*GET OUT-1.EQ-1.FREQ

$> "SET OUT-1.EQ-1.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.FREQ 2000.0
```

5.34 OUT-{OID}.EQ-{OEID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Float] (Range: -15.0 to 15.0 dB)

Output EQ band gain

Example:

```
$> "GET OUT-1.EQ-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.GAIN -6.00
*GET OUT-1.EQ-1.GAIN

$> "SET OUT-1.EQ-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.GAIN -12.00
```

5.35 OUT-{OID}.EQ-{OEID}.Q

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Float] (Range: 0.4 to 30.0)

Output EQ band Q factor

Example:

```
$> "GET OUT-1.EQ-1.Q" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.Q 1.0
*GET OUT-1.EQ-1.Q

$> "SET OUT-1.EQ-1.Q 2.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.Q 2.0
```

5.36 OUT-{OID}.EQ-{OEID}.TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: Enum {PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}

NOTES: TILT_SHELF added in 2026.x firmware

Output EQ band filter type

Example:

```
$> "GET OUT-1.EQ-1.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.TYPE PARAMETRIC
*GET OUT-1.EQ-1.TYPE

$> "SET OUT-1.EQ-1.TYPE LOWPASS_6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.TYPE LOWPASS_6
```

5.37 OUT-{OID}.EQ.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass output EQ

Example:

```
$> "GET OUT-1.EQ.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ.BYPASS 1
*GET OUT-1.EQ.BYPASS

$> "SET OUT-1.EQ.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ.BYPASS 0
```

5.38 OUT-{OID}.FIR.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass FIR filter

Example:

```
$> "GET OUT-1.FIR.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.FIR.BYPASS 1
*GET OUT-1.FIR.BYPASS

$> "SET OUT-1.FIR.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.FIR.BYPASS 0
```

5.39 OUT-{OID}.FIR.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

FIR filter settings are protected by preset

Example:

```
$> "GET OUT-1.FIR.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.FIR.PROTECTED 1
*GET OUT-1.FIR.PROTECTED
```

5.40 OUT-{OID}.FIR.TAPS

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Integer]

Number of FIR filter taps loaded

Example:

```
$> "GET OUT-1.FIR.TAPS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.FIR.TAPS 512
*GET OUT-1.FIR.TAPS
```

5.41 OUT-{OID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: -30.0 to 15.0 dB)

NOTES: INC is bounded to [-15.0, 15.0]

Output gain

Example:

```
$> "GET OUT-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.GAIN -6.00
*GET OUT-1.GAIN

$> "SET OUT-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.GAIN -12.00
```

5.42 OUT-{OID}.LIMITER.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Limiter settings are protected by preset

Example:

```
$> "GET OUT-1.LIMITER.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.LIMITER.PROTECTED 1
*GET OUT-1.LIMITER.PROTECTED
```

5.43 OUT-{OID}.MUTE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Output mute

Example:

```
$> "GET OUT-1.MUTE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.MUTE 1
*GET OUT-1.MUTE

$> "SET OUT-1.MUTE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.MUTE 0
```

5.44 OUT-{OID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String] (Max Length 32 chars)

Output channel name

Example:

```
$> "GET OUT-1.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.NAME "Main L"
*GET OUT-1.NAME

$> "SET OUT-1.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.NAME "New Name"
```

5.45 OUT-{OID}.OUTPUT_HIGHPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 1000.0 Hz)

Output high-pass filter frequency (0 = disabled)

Example:

```
$> "GET OUT-1.OUTPUT_HIGHPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.OUTPUT_HIGHPASS 500.00
*GET OUT-1.OUTPUT_HIGHPASS

$> "SET OUT-1.OUTPUT_HIGHPASS 500.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.OUTPUT_HIGHPASS 500.00
```

5.46 OUT-{OID}.OUTPUT_MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum:

- **OFF** - Output disabled
- **4R** - 4 ohm low-impedance mode (reported only, not settable)
- **8R** - 8 ohm low-impedance mode
- **70V** - 70V distributed audio mode

- **100V** - 100V distributed audio mode
- **BTL** - Bridge-tied load mode

NOTES: 4R is reported by GET but not settable; 70V/100V/BTL apply to the primary channel only (BTL requires hardware support)

Output mode

Example:

```
$> "GET OUT-1.OUTPUT_MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.OUTPUT_MODE OFF
*GET OUT-1.OUTPUT_MODE

$> "SET OUT-1.OUTPUT_MODE 4R" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.OUTPUT_MODE 4R
```

5.47 OUT-{OID}.OUTPUT_MODE.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Output mode setting is protected by preset

Example:

```
$> "GET OUT-1.OUTPUT_MODE.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.OUTPUT_MODE.PROTECTED 1
*GET OUT-1.OUTPUT_MODE.PROTECTED
```

5.48 OUT-{OID}.PEAK_LIMITER.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0003 to 0.1 Sec)

Peak limiter attack time

Example:

```
$> "GET OUT-1.PEAK_LIMITER.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.ATTACK 0.05
*GET OUT-1.PEAK_LIMITER.ATTACK

$> "SET OUT-1.PEAK_LIMITER.ATTACK 0.05" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.ATTACK 0.05
```

5.49 OUT-{OID}.PEAK_LIMITER.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Enable automatic peak limiter makeup gain

Example:

```
$> "GET OUT-1.PEAK_LIMITER.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.AUTO 1
*GET OUT-1.PEAK_LIMITER.AUTO

$> "SET OUT-1.PEAK_LIMITER.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.AUTO 0
```

5.50 OUT-{OID}.PEAK_LIMITER.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass peak limiter

Example:

```
$> "GET OUT-1.PEAK_LIMITER.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.BYPASS 1
*GET OUT-1.PEAK_LIMITER.BYPASS

$> "SET OUT-1.PEAK_LIMITER.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.BYPASS 0
```

5.51 OUT-{OID}.PEAK_LIMITER.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 1.0 Sec)

Peak limiter hold time

Example:

```
$> "GET OUT-1.PEAK_LIMITER.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.HOLD 0.50
*GET OUT-1.PEAK_LIMITER.HOLD

$> "SET OUT-1.PEAK_LIMITER.HOLD 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.HOLD 0.50
```

5.52 OUT-{OID}.PEAK_LIMITER.KNEE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 6.0 dB)

Peak limiter knee width

Example:

```
$> "GET OUT-1.PEAK_LIMITER.KNEE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.KNEE 3.00
*GET OUT-1.PEAK_LIMITER.KNEE

$> "SET OUT-1.PEAK_LIMITER.KNEE 3.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.KNEE 3.00
```

5.53 OUT-{OID}.PEAK_LIMITER.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.004 to 2.0 Sec)

Peak limiter release time

Example:

```
$> "GET OUT-1.PEAK_LIMITER.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.RELEASE 1.00
*GET OUT-1.PEAK_LIMITER.RELEASE

$> "SET OUT-1.PEAK_LIMITER.RELEASE 1.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.RELEASE 1.00
```

5.54 OUT-{OID}.PEAK_LIMITER.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 1.0 to 200.0 Vpeak)

Peak limiter threshold (peak volts)

Example:

```
$> "GET OUT-1.PEAK_LIMITER.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.THRESHOLD -20.00
*GET OUT-1.PEAK_LIMITER.THRESHOLD

$> "SET OUT-1.PEAK_LIMITER.THRESHOLD 100.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.THRESHOLD 100.50
```

5.55 OUT-{OID}.POLARITY

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Integer]

Output polarity (1 = normal, -1 = inverted)

Example:

```
$> "GET OUT-1.POLARITY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.POLARITY 1
*GET OUT-1.POLARITY

$> "SET OUT-1.POLARITY 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.POLARITY 100
```

5.56 OUT-{OID}.POLARITY.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Polarity setting is protected by preset

Example:

```
$> "GET OUT-1.POLARITY.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.POLARITY.PROTECTED 1
*GET OUT-1.POLARITY.PROTECTED
```

5.57 OUT-{OID}.PRESET.CUSTOMIZED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Preset has been customized from factory settings

Example:

```
$> "GET OUT-1.PRESET.CUSTOMIZED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.CUSTOMIZED 1
*GET OUT-1.PRESET.CUSTOMIZED
```

5.58 OUT-{OID}.PRESET.ID

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String]

Active speaker preset ID

Example:

```
$> "GET OUT-1.PRESET.ID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.ID ""
*GET OUT-1.PRESET.ID
```

5.59 OUT-{OID}.PRESET.LOCKED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Preset is locked (cannot be modified)

Example:

```
$> "GET OUT-1.PRESET.LOCKED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.LOCKED 1
*GET OUT-1.PRESET.LOCKED
```

5.60 OUT-{OID}.PRESET.NAME

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String] (Max Length 64 chars)

Active speaker preset name

Example:

```
$> "GET OUT-1.PRESET.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.NAME "Factory Preset"
*GET OUT-1.PRESET.NAME
```

5.61 OUT-{OID}.RMS_LIMITER.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.01 to 30.0 Sec)

RMS limiter attack time

Example:

```
$> "GET OUT-1.RMS_LIMITER.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.ATTACK 15.01
*GET OUT-1.RMS_LIMITER.ATTACK

$> "SET OUT-1.RMS_LIMITER.ATTACK 15.01" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.ATTACK 15.01
```

5.62 OUT-{OID}.RMS_LIMITER.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass RMS limiter

Example:

```
$> "GET OUT-1.RMS_LIMITER.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.BYPASS 1
*GET OUT-1.RMS_LIMITER.BYPASS

$> "SET OUT-1.RMS_LIMITER.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.BYPASS 0
```

5.63 OUT-{OID}.RMS_LIMITER.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 1.0 Sec)

RMS limiter hold time

Example:

```
$> "GET OUT-1.RMS_LIMITER.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.HOLD 0.50
*GET OUT-1.RMS_LIMITER.HOLD

$> "SET OUT-1.RMS_LIMITER.HOLD 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.HOLD 0.50
```

5.64 OUT-{OID}.RMS_LIMITER.KNEE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 6.0 dB)

RMS limiter knee width

Example:

```
$> "GET OUT-1.RMS_LIMITER.KNEE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.KNEE 3.00
*GET OUT-1.RMS_LIMITER.KNEE

$> "SET OUT-1.RMS_LIMITER.KNEE 3.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.KNEE 3.00
```

5.65 OUT-{OID}.RMS_LIMITER.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.01 to 30.0 Sec)

RMS limiter release time

Example:

```
$> "GET OUT-1.RMS_LIMITER.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.RELEASE 15.01
*GET OUT-1.RMS_LIMITER.RELEASE

$> "SET OUT-1.RMS_LIMITER.RELEASE 15.01" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.RELEASE 15.01
```

5.66 OUT-{OID}.RMS_LIMITER.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 1.0 to 150.0 Vpeak)

RMS limiter threshold (peak volts)

Example:

```
$> "GET OUT-1.RMS_LIMITER.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.THRESHOLD -20.00
*GET OUT-1.RMS_LIMITER.THRESHOLD

$> "SET OUT-1.RMS_LIMITER.THRESHOLD 75.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.THRESHOLD 75.50
```

5.67 OUT-{OID}.SPEAKER_DELAY.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass speaker delay

Example:

```
$> "GET OUT-1.SPEAKER_DELAY.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_DELAY.BYPASS 1
*GET OUT-1.SPEAKER_DELAY.BYPASS

$> "SET OUT-1.SPEAKER_DELAY.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_DELAY.BYPASS 0
```

5.68 OUT-{OID}.SPEAKER_DELAY.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Speaker delay settings are protected by preset

Example:

```
$> "GET OUT-1.SPEAKER_DELAY.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_DELAY.PROTECTED 1
*GET OUT-1.SPEAKER_DELAY.PROTECTED
```

5.69 OUT-{OID}.SPEAKER_DELAY.TIME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 0.01 Sec)

Speaker delay time for time alignment

Example:

```
$> "GET OUT-1.SPEAKER_DELAY.TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_DELAY.TIME 0.01
*GET OUT-1.SPEAKER_DELAY.TIME

$> "SET OUT-1.SPEAKER_DELAY.TIME 0.01" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_DELAY.TIME 0.01
```

5.70 OUT-{OID}.SPEAKER_EQ-{OSEID}.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Boolean]

Bypass individual speaker EQ band

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.BYPASS 1
*GET OUT-1.SPEAKER_EQ-1.BYPASS

$> "SET OUT-1.SPEAKER_EQ-1.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.BYPASS 0
```

5.71 OUT-{OID}.SPEAKER_EQ-{OSEID}.FREQ

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Speaker EQ band center frequency

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.FREQ 1000.0
*GET OUT-1.SPEAKER_EQ-1.FREQ

$> "SET OUT-1.SPEAKER_EQ-1.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.FREQ 2000.0
```

5.72 OUT-{OID}.SPEAKER_EQ-{OSEID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Float] (Range: -15.0 to 15.0 dB)

Speaker EQ band gain

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.GAIN -6.00
*GET OUT-1.SPEAKER_EQ-1.GAIN

$> "SET OUT-1.SPEAKER_EQ-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.GAIN -12.00
```

5.73 OUT-{OID}.SPEAKER_EQ-{OSEID}.Q

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Float] (Range: 0.4 to 30.0)

Speaker EQ band Q factor

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.Q" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.Q 1.0
*GET OUT-1.SPEAKER_EQ-1.Q

$> "SET OUT-1.SPEAKER_EQ-1.Q 2.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.Q 2.0
```

5.74 OUT-{OID}.SPEAKER_EQ-{OSEID}.TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: Enum {PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}

Speaker EQ band filter type

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.TYPE PARAMETRIC
*GET OUT-1.SPEAKER_EQ-1.TYPE

$> "SET OUT-1.SPEAKER_EQ-1.TYPE LOWPASS_6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.TYPE LOWPASS_6
```

5.75 OUT-{OID}.SPEAKER_EQ.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass speaker EQ

Example:

```
$> "GET OUT-1.SPEAKER_EQ.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ.BYPASS 1
*GET OUT-1.SPEAKER_EQ.BYPASS

$> "SET OUT-1.SPEAKER_EQ.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ.BYPASS 0
```

5.76 OUT-{OID}.SPEAKER_EQ.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Speaker EQ settings are protected by preset

Example:

```
$> "GET OUT-1.SPEAKER_EQ.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ.PROTECTED 1
*GET OUT-1.SPEAKER_EQ.PROTECTED
```

5.77 OUT-{OID}.SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String] (Max Length 1 chars)

Output source zone (ZID)

Example:

```
$> "GET OUT-1.SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SRC ""
*GET OUT-1.SRC

$> "SET OUT-1.SRC "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SRC "New Name"
```

5.78 OUT-{OID}.SRC_CHANNEL

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum:

- **L** - Left channel
- **R** - Right channel
- **S** - Summed mono (L+R)

Output source channel (Left, Right, Summed)

Example:

```
$> "GET OUT-1.SRC_CHANNEL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SRC_CHANNEL L
*GET OUT-1.SRC_CHANNEL

$> "SET OUT-1.SRC_CHANNEL R" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SRC_CHANNEL R
```

5.79 OUT-{OID}.XR.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass crossover filters

Example:

```
$> "GET OUT-1.XR.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.BYPASS 1
*GET OUT-1.XR.BYPASS

$> "SET OUT-1.XR.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.BYPASS 0
```

5.80 OUT-{OID}.XR.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: -15.0 to 15.0 dB)

Crossover section gain

Example:

```
$> "GET OUT-1.XR.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.GAIN -6.00
*GET OUT-1.XR.GAIN

$> "SET OUT-1.XR.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.GAIN -12.00
```

5.81 OUT-{OID}.XR.HIGHPASS_FREQUENCY

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

High-pass crossover frequency

Example:

```
$> "GET OUT-1.XR.HIGHPASS_FREQUENCY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.HIGHPASS_FREQUENCY 1000.0
*GET OUT-1.XR.HIGHPASS_FREQUENCY

$> "SET OUT-1.XR.HIGHPASS_FREQUENCY 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.HIGHPASS_FREQUENCY 2000.0
```

5.82 OUT-{OID}.XR.HIGHPASS_TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum {OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}

High-pass crossover filter type

Example:

```
$> "GET OUT-1.XR.HIGHPASS_TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.HIGHPASS_TYPE OFF
*GET OUT-1.XR.HIGHPASS_TYPE

$> "SET OUT-1.XR.HIGHPASS_TYPE BUT6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.HIGHPASS_TYPE BUT6
```

5.83 OUT-{OID}.XR.LOWPASS_FREQUENCY

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Low-pass crossover frequency

Example:

```
$> "GET OUT-1.XR.LOWPASS_FREQUENCY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.LOWPASS_FREQUENCY 1000.0
*GET OUT-1.XR.LOWPASS_FREQUENCY

$> "SET OUT-1.XR.LOWPASS_FREQUENCY 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.LOWPASS_FREQUENCY 2000.0
```

5.84 OUT-{OID}.XR.LOWPASS_TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum {OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}

Low-pass crossover filter type

Example:

```
$> "GET OUT-1.XR.LOWPASS_TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.LOWPASS_TYPE OFF
*GET OUT-1.XR.LOWPASS_TYPE

$> "SET OUT-1.XR.LOWPASS_TYPE BUT6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.LOWPASS_TYPE BUT6
```

5.85 OUT-{OID}.XR.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Crossover settings are protected by preset

Example:

```
$> "GET OUT-1.XR.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.PROTECTED 1
*GET OUT-1.XR.PROTECTED
```

5.86 OUT.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of output channels

Example:

```
$> "GET OUT.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT.COUNT 4
*GET OUT.COUNT
```

5.87 OUT.EQ.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of output EQ bands

Example:

```
$> "GET OUT.EQ.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT.EQ.COUNT 4
*GET OUT.EQ.COUNT
```

5.88 OUT.SPEAKER_EQ.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of speaker EQ bands

Example:

```
$> "GET OUT.SPEAKER_EQ.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT.SPEAKER_EQ.COUNT 4
*GET OUT.SPEAKER_EQ.COUNT
```

5.89 ROUT-{RID}.DYN.CLIP

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Boolean]

NOTES: Subscription only — streamed on SUBSCRIBE DYN

Route clip indicator (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+ROUT-1.DYN.CLIP 1
```

5.90 ROUT-{RID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Float] (dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN

Route signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+ROUT-1.DYN.SIGNAL -12.00
```

5.91 ROUT-{RID}.GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Float] (Range: -40.0 to 20.0 dB)

Routing gain level

Example:

```
$> "GET ROUT-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ROUT-1.GAIN -6.00
*GET ROUT-1.GAIN

$> "SET ROUT-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ROUT-1.GAIN -12.00
```

5.92 ROUT-{RID}.SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Integer]

NOTES: See {RSID} Route Source definitions

Routing source (RSID value)

Example:

```
$> "GET ROUT-1.SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ROUT-1.SRC 1
*GET ROUT-1.SRC

$> "SET ROUT-1.SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ROUT-1.SRC 100
```

5.93 ROUT-{RID}.SRC_CHANNEL

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: Enum {L, R, S}

Routing source channel (Left, Right, Summed)

Example:

```
$> "GET ROUT-1.SRC_CHANNEL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ROUT-1.SRC_CHANNEL L
*GET ROUT-1.SRC_CHANNEL

$> "SET ROUT-1.SRC_CHANNEL R" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ROUT-1.SRC_CHANNEL R
```

5.94 SETUP.GPIO.PIN2

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, STANDBY_NO, STANDBY_NC, MUTE_NO, MUTE_NC}

GPIO Pin 2 function

Example:

```
$> "GET SETUP.GPIO.PIN2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN2 OFF
*GET SETUP.GPIO.PIN2

$> "SET SETUP.GPIO.PIN2 STANDBY_NO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN2 STANDBY_NO
```

5.95 SETUP.GPIO.PIN4

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL}

GPIO Pin 4 function

Example:

```
$> "GET SETUP.GPIO.PIN4" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN4 OFF
*GET SETUP.GPIO.PIN4

$> "SET SETUP.GPIO.PIN4 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN4 VOLUME_CONTROL
```

5.96 SETUP.GPIO.PIN5

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL}

GPIO Pin 5 function

Example:

```
$> "GET SETUP.GPIO.PIN5" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN5 OFF
*GET SETUP.GPIO.PIN5

$> "SET SETUP.GPIO.PIN5 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN5 VOLUME_CONTROL
```

5.97 SETUP.GPIO.PIN6

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL, TRIGGER_12V_IN}

GPIO Pin 6 function

Example:

```
$> "GET SETUP.GPIO.PIN6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN6 OFF
*GET SETUP.GPIO.PIN6

$> "SET SETUP.GPIO.PIN6 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN6 VOLUME_CONTROL
```

5.98 SETUP.GPIO.PIN7

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL, TRIGGER_12V_OUT}

GPIO Pin 7 function

Example:

```
$> "GET SETUP.GPIO.PIN7" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN7 OFF
*GET SETUP.GPIO.PIN7

$> "SET SETUP.GPIO.PIN7 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN7 VOLUME_CONTROL
```

5.99 SETUP.GPIO.PIN8

TYPE: Register

METHODS: Get, Set

VALUES: Enum {VCC_3V3}

GPIO Pin 8 function

Example:

```
$> "GET SETUP.GPIO.PIN8" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN8 VCC_3V3
*GET SETUP.GPIO.PIN8

$> "SET SETUP.GPIO.PIN8 VCC_3V3" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN8 VCC_3V3
```

5.100 SETUP.LAN.DNS1

TYPE: Register

METHODS: Get

VALUES: [String]

Primary DNS server

Example:

```
$> "GET SETUP.LAN.DNS1" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.DNS1 ""
*GET SETUP.LAN.DNS1
```

5.101 SETUP.LAN.DNS2

TYPE: Register

METHODS: Get

VALUES: [String]

Secondary DNS server

Example:

```
$> "GET SETUP.LAN.DNS2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.DNS2 ""
*GET SETUP.LAN.DNS2
```

5.102 SETUP.LAN.GATEWAY

TYPE: Register

METHODS: Get

VALUES: [String]

LAN gateway address

Example:

```
$> "GET SETUP.LAN.GATEWAY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.GATEWAY ""
*GET SETUP.LAN.GATEWAY
```

5.103 SETUP.LAN.IP

TYPE: Register

METHODS: Get

VALUES: [String]

LAN IP address

Example:

```
$> "GET SETUP.LAN.IP" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.IP "192.168.1.100"
*GET SETUP.LAN.IP
```

5.104 SETUP.LAN.MASK

TYPE: Register

METHODS: Get

VALUES: [String]

LAN subnet mask

Example:

```
$> "GET SETUP.LAN.MASK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.MASK ""
*GET SETUP.LAN.MASK
```

5.105 SETUP.LAN.NETWORK_MODE

TYPE: Register

METHODS: Get

VALUES: Enum:

- **STATIC** - Static IP configuration
- **DHCP** - Dynamic IP via DHCP

Network configuration mode

Example:

```
$> "GET SETUP.LAN.NETWORK_MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.NETWORK_MODE STATIC
*GET SETUP.LAN.NETWORK_MODE
```

5.106 SETUP.POWER.MUTE_TIME

TYPE: Register

METHODS: Get, Set

VALUES: [Integer] (Range: 0 to 3600 Sec)

Mute time after power on

Example:

```
$> "GET SETUP.POWER.MUTE_TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.POWER.MUTE_TIME 1800
*GET SETUP.POWER.MUTE_TIME

$> "SET SETUP.POWER.MUTE_TIME 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.POWER.MUTE_TIME 100
```

5.107 SETUP.POWER.POWER_ON

TYPE: Register

METHODS: Get, Set

VALUES: Enum {AUDIO, AUDIO_ECO, AUDIO_DSP, TRIGGER, TRIGGER_ECO, NETWORK}

Power-on / standby behaviour mode

Example:

```
$> "GET SETUP.POWER.POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.POWER.POWER_ON AUDIO
*GET SETUP.POWER.POWER_ON

$> "SET SETUP.POWER.POWER_ON AUDIO_ECO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.POWER.POWER_ON AUDIO_ECO
```

5.108 SETUP.POWER.STANDBY_TIME

TYPE: Register

METHODS: Get, Set

VALUES: [Integer] (Range: 0 to 3600 Sec)

Auto standby after no signal timeout (0 = disabled)

Example:

```
$> "GET SETUP.POWER.STANDBY_TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.POWER.STANDBY_TIME 1800
*GET SETUP.POWER.STANDBY_TIME

$> "SET SETUP.POWER.STANDBY_TIME 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.POWER.STANDBY_TIME 100
```

5.109 SETUP.SYSTEM.ASSET_TAG

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Asset tag identifier

Example:

```
$> "GET SETUP.SYSTEM.ASSET_TAG" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.ASSET_TAG ""
*GET SETUP.SYSTEM.ASSET_TAG

$> "SET SETUP.SYSTEM.ASSET_TAG "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.ASSET_TAG "New Name"
```

5.110 SETUP.SYSTEM.CONTACT_INFO

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Contact information

Example:

```
$> "GET SETUP.SYSTEM.CONTACT_INFO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CONTACT_INFO ""
*GET SETUP.SYSTEM.CONTACT_INFO

$> "SET SETUP.SYSTEM.CONTACT_INFO "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.CONTACT_INFO "New Name"
```

5.111 SETUP.SYSTEM.CUSTOM1

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 8192 chars)

Custom field 1 for integrator use

Example:

```
$> "GET SETUP.SYSTEM.CUSTOM1" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOM1 ""
*GET SETUP.SYSTEM.CUSTOM1

$> "SET SETUP.SYSTEM.CUSTOM1 "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.CUSTOM1 "New Name"
```

5.112 SETUP.SYSTEM.CUSTOM2

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 8192 chars)

Custom field 2 for integrator use

Example:

```
$> "GET SETUP.SYSTEM.CUSTOM2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOM2 ""
*GET SETUP.SYSTEM.CUSTOM2

$> "SET SETUP.SYSTEM.CUSTOM2 "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.CUSTOM2 "New Name"
```

5.113 SETUP.SYSTEM.CUSTOM3

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 8192 chars)

Custom field 3 for integrator use

Example:

```
$> "GET SETUP.SYSTEM.CUSTOM3" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOM3 ""
*GET SETUP.SYSTEM.CUSTOM3

$> "SET SETUP.SYSTEM.CUSTOM3 "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.CUSTOM3 "New Name"
```

5.114 SETUP.SYSTEM.CUSTOMER_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Customer name

Example:

```
$> "GET SETUP.SYSTEM.CUSTOMER_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOMER_NAME "Acme Corp"
*GET SETUP.SYSTEM.CUSTOMER_NAME

$> "SET SETUP.SYSTEM.CUSTOMER_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.CUSTOMER_NAME "New Name"
```

5.115 SETUP.SYSTEM.DEVICE_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

User-defined device name

Example:

```
$> "GET SETUP.SYSTEM.DEVICE_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.DEVICE_NAME "Amp-01"
*GET SETUP.SYSTEM.DEVICE_NAME

$> "SET SETUP.SYSTEM.DEVICE_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.DEVICE_NAME "New Name"
```

5.116 SETUP.SYSTEM.INSTALLER_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Installer name

Example:

```
$> "GET SETUP.SYSTEM.INSTALLER_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.INSTALLER_NAME "Channel 1"
*GET SETUP.SYSTEM.INSTALLER_NAME

$> "SET SETUP.SYSTEM.INSTALLER_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET SETUP.SYSTEM.INSTALLER_NAME "New Name"
```

5.117 SETUP.SYSTEM.INSTALL_DATE

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Installation date

Example:

```
$> "GET SETUP.SYSTEM.INSTALL_DATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.INSTALL_DATE ""
*GET SETUP.SYSTEM.INSTALL_DATE

$> "SET SETUP.SYSTEM.INSTALL_DATE "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.INSTALL_DATE "New Name"
```

5.118 SETUP.SYSTEM.INSTALL_NOTES

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 256 chars)

Installation notes

Example:

```
$> "GET SETUP.SYSTEM.INSTALL_NOTES" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.INSTALL_NOTES ""
*GET SETUP.SYSTEM.INSTALL_NOTES

$> "SET SETUP.SYSTEM.INSTALL_NOTES "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.INSTALL_NOTES "New Name"
```

5.119 SETUP.SYSTEM.LOCATING

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable locating mode (flashing LEDs)

Example:

```
$> "GET SETUP.SYSTEM.LOCATING" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.LOCATING 1
*GET SETUP.SYSTEM.LOCATING

$> "SET SETUP.SYSTEM.LOCATING 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.LOCATING 0
```

5.120 SETUP.SYSTEM.VENUE_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Installation venue name

Example:

```
$> "GET SETUP.SYSTEM.VENUE_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.VENUE_NAME "Main Hall"
*GET SETUP.SYSTEM.VENUE_NAME

$> "SET SETUP.SYSTEM.VENUE_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.VENUE_NAME "New Name"
```

5.121 SETUP.WIFI.AP_PASS

TYPE: Register

METHODS: Get

VALUES: [String]

Access Point password

Example:

```
$> "GET SETUP.WIFI.AP_PASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.AP_PASS ""
*GET SETUP.WIFI.AP_PASS
```

5.122 SETUP.WIFI.AP_SSID

TYPE: Register

METHODS: Get

VALUES: [String]

Access Point SSID

Example:

```
$> "GET SETUP.WIFI.AP_SSID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.AP_SSID ""
*GET SETUP.WIFI.AP_SSID
```

5.123 SETUP.WIFI.DISABLE_AFTER

TYPE: Register

METHODS: Get

VALUES: [Float] (Range: 0.0 to 3600.0 Sec)

Disable WiFi after timeout (0 = never)

Example:

```
$> "GET SETUP.WIFI.DISABLE_AFTER" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.DISABLE_AFTER 1800.00
*GET SETUP.WIFI.DISABLE_AFTER
```

5.124 SETUP.WIFI.DISABLE_LAN_CONNECTED

TYPE: Register

METHODS: Get

VALUES: [Boolean]

Disable WiFi when LAN connected

Example:

```
$> "GET SETUP.WIFI.DISABLE_LAN_CONNECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.DISABLE_LAN_CONNECTED 1
*GET SETUP.WIFI.DISABLE_LAN_CONNECTED
```

5.125 SETUP.WIFI.ENABLE

TYPE: Register

METHODS: Get

VALUES: [Boolean]

WiFi enabled

Example:

```
$> "GET SETUP.WIFI.ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.ENABLE 1
*GET SETUP.WIFI.ENABLE
```

5.126 SETUP.WIFI.MODE

TYPE: Register

METHODS: Get

VALUES: Enum:

- **AP** - Access Point mode
- **STA** - Station (client) mode

WiFi mode (Access Point or Station)

Example:

```
$> "GET SETUP.WIFI.MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.MODE AP
*GET SETUP.WIFI.MODE
```

5.127 SETUP.WIFI.STA_PASS

TYPE: Register

METHODS: Get

VALUES: [String]

Station mode password

Example:

```
$> "GET SETUP.WIFI.STA_PASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.STA_PASS ""
*GET SETUP.WIFI.STA_PASS
```

5.128 SETUP.WIFI.STA_SSID

TYPE: Register

METHODS: Get

VALUES: [String]

Station mode SSID

Example:

```
$> "GET SETUP.WIFI.STA_SSID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.STA_SSID ""
*GET SETUP.WIFI.STA_SSID
```

5.129 SYSTEM.DANTE.AES67_ENABLED

TYPE: Register

METHODS: Get

VALUES: [Boolean]

AES67 mode enabled

Example:

```
$> "GET SYSTEM.DANTE.AES67_ENABLED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.AES67_ENABLED 1
*GET SYSTEM.DANTE.AES67_ENABLED
```

5.130 SYSTEM.DANTE.CLOCK_STATE

TYPE: Register

METHODS: Get

VALUES: Enum {NO_CLOCK, LOCKING, LOCKED}

Dante clock synchronization state

Example:

```
$> "GET SYSTEM.DANTE.CLOCK_STATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.CLOCK_STATE NO_CLOCK
*GET SYSTEM.DANTE.CLOCK_STATE
```

5.131 SYSTEM.DANTE.DEVICE_NAME

TYPE: Register

METHODS: Get

VALUES: [String] (Max Length 32 chars)

Dante device name

Example:

```
$> "GET SYSTEM.DANTE.DEVICE_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.DEVICE_NAME "Pascal-01"
*GET SYSTEM.DANTE.DEVICE_NAME
```

5.132 SYSTEM.DANTE.ENCODING

TYPE: Register

METHODS: Get

VALUES: Enum {PCM16, PCM24, PCM32}

Dante audio encoding format

Example:

```
$> "GET SYSTEM.DANTE.ENCODING" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.ENCODING PCM16
*GET SYSTEM.DANTE.ENCODING
```

5.133 SYSTEM.DANTE.FIRMWARE_VERSION

TYPE: Register

METHODS: Get

VALUES: [String]

Dante firmware version

Example:

```
$> "GET SYSTEM.DANTE.FIRMWARE_VERSION" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.FIRMWARE_VERSION "5.3"
*GET SYSTEM.DANTE.FIRMWARE_VERSION
```

5.134 SYSTEM.DANTE.IP

TYPE: Register

METHODS: Get

VALUES: [String]

Dante network IP address

Example:

```
$> "GET SYSTEM.DANTE.IP" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.IP "192.168.1.100"
*GET SYSTEM.DANTE.IP
```

5.135 SYSTEM.DANTE.LINK_SPEED

TYPE: Register

METHODS: Get

VALUES: [Integer]

Dante network link speed

Example:

```
$> "GET SYSTEM.DANTE.LINK_SPEED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.LINK_SPEED 1
*GET SYSTEM.DANTE.LINK_SPEED
```

5.136 SYSTEM.DANTE.MAC

TYPE: Register

METHODS: Get

VALUES: [String]

Dante MAC address

Example:

```
$> "GET SYSTEM.DANTE.MAC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.MAC "00:1A:2B:3C:4D:5E"
*GET SYSTEM.DANTE.MAC
```

5.137 SYSTEM.DANTE.MUTE_STATE

TYPE: Register

METHODS: Get

VALUES: [String]

Dante mute state

Example:

```
$> "GET SYSTEM.DANTE.MUTE_STATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.MUTE_STATE ""
*GET SYSTEM.DANTE.MUTE_STATE
```

5.138 SYSTEM.DANTE.SAMPLE_RATE

TYPE: Register

METHODS: Get

VALUES: Enum {44100, 48000, 96000}

Dante sample rate

Example:

```
$> "GET SYSTEM.DANTE.SAMPLE_RATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.SAMPLE_RATE 44100
*GET SYSTEM.DANTE.SAMPLE_RATE
```

5.139 SYSTEM.DANTE.SOFTWARE_VERSION

TYPE: Register

METHODS: Get

VALUES: [String]

Dante software version

Example:

```
$> "GET SYSTEM.DANTE.SOFTWARE_VERSION" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.SOFTWARE_VERSION "5.3"
*GET SYSTEM.DANTE.SOFTWARE_VERSION
```

5.140 SYSTEM.DEVICE.FIRMWARE

TYPE: Register

METHODS: Get

VALUES: [String]

Firmware version

Example:

```
$> "GET SYSTEM.DEVICE.FIRMWARE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.FIRMWARE "1.8.0"
*GET SYSTEM.DEVICE.FIRMWARE
```

5.141 SYSTEM.DEVICE.FIRMWARE_DATE

TYPE: Register

METHODS: Get

VALUES: [String]

Firmware build date

Example:

```
$> "GET SYSTEM.DEVICE.FIRMWARE_DATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.FIRMWARE_DATE "1.8.0"
*GET SYSTEM.DEVICE.FIRMWARE_DATE
```

5.142 SYSTEM.DEVICE.HWID

TYPE: Register

METHODS: Get

VALUES: [Integer]

Hardware ID

Example:

```
$> "GET SYSTEM.DEVICE.HWID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.HWID 1
*GET SYSTEM.DEVICE.HWID
```

5.143 SYSTEM.DEVICE.MAC

TYPE: Register

METHODS: Get

VALUES: [String]

LAN MAC address

Example:

```
$> "GET SYSTEM.DEVICE.MAC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.MAC "00:1A:2B:3C:4D:5E"
*GET SYSTEM.DEVICE.MAC
```

5.144 SYSTEM.DEVICE.MODEL_NAME

TYPE: Register

METHODS: Get

VALUES: [String] (Max Length 32 chars)

Product model name

Example:

```
$> "GET SYSTEM.DEVICE.MODEL_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.MODEL_NAME "BLAZE-4"
*GET SYSTEM.DEVICE.MODEL_NAME
```

5.145 SYSTEM.DEVICE.SERIAL

TYPE: Register

METHODS: Get

VALUES: [String]

Serial number

Example:

```
$> "GET SYSTEM.DEVICE.SERIAL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.SERIAL "SN12345678"
*GET SYSTEM.DEVICE.SERIAL
```

5.146 SYSTEM.DEVICE.SWID

TYPE: Register

METHODS: Get

VALUES: [Integer]

Software ID

Example:

```
$> "GET SYSTEM.DEVICE.SWID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.SWID 1
*GET SYSTEM.DEVICE.SWID
```

5.147 SYSTEM.DEVICE.VENDOR_NAME

TYPE: Register

METHODS: Get

VALUES: [String] (Max Length 32 chars)

Manufacturer name

Example:

```
$> "GET SYSTEM.DEVICE.VENDOR_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.VENDOR_NAME "Pascal"
*GET SYSTEM.DEVICE.VENDOR_NAME
```

5.148 SYSTEM.DEVICE.WIFI_MAC

TYPE: Register

METHODS: Get

VALUES: [String]

WiFi MAC address

Example:

```
$> "GET SYSTEM.DEVICE.WIFI_MAC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.WIFI_MAC "00:1A:2B:3C:4D:5E"
*GET SYSTEM.DEVICE.WIFI_MAC
```

5.149 SYSTEM.SECURITY.PASSWORD_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable password protection

Example:

```
$> "GET SYSTEM.SECURITY.PASSWORD_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.SECURITY.PASSWORD_ENABLE 1
*GET SYSTEM.SECURITY.PASSWORD_ENABLE

$> "SET SYSTEM.SECURITY.PASSWORD_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SYSTEM.SECURITY.PASSWORD_ENABLE 0
```

5.150 SYSTEM.SECURITY.PASSWORD_HASH

TYPE: Register

METHODS: Get, Set

VALUES: [String]

Password hash (write-only for security)

Example:

```
$> "GET SYSTEM.SECURITY.PASSWORD_HASH" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.SECURITY.PASSWORD_HASH ""
*GET SYSTEM.SECURITY.PASSWORD_HASH

$> "SET SYSTEM.SECURITY.PASSWORD_HASH "New Name"" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET SYSTEM.SECURITY.PASSWORD_HASH "New Name"
```

5.151 SYSTEM.STATUS.LAN

TYPE: Register

METHODS: Get

VALUES: [String]

LAN connection status - IP address or empty if not connected

Example:

```
$> "GET SYSTEM.STATUS.LAN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.LAN ""
*GET SYSTEM.STATUS.LAN
```

5.152 SYSTEM.STATUS.SIGNAL_IN

TYPE: Register

METHODS: Get

VALUES: Enum:

- **OFF** - Input(s) is off
- **NO_SIGNAL** - Input(s) has no signal (below threshold)
- **SIGNAL** - Input(s) has signal (above threshold)
- **CLIP** - Input(s) is clipping ADC - please decrease sensitivity

Input signal status

Example:

```
$> "GET SYSTEM.STATUS.SIGNAL_IN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.SIGNAL_IN OFF
*GET SYSTEM.STATUS.SIGNAL_IN
```

5.153 SYSTEM.STATUS.SIGNAL_OUT

TYPE: Register

METHODS: Get

VALUES: Enum:

- **OFF** - Output(s) is off
- **NO_SIGNAL** - Output(s) has no signal (below threshold)
- **SIGNAL** - Output(s) has signal (above threshold)
- **CLIP** - Output(s) is clipping in amplifier - please decrease volume
- **FAULT** - Output(s) has unspecified fault

Output signal status

Example:

```
$> "GET SYSTEM.STATUS.SIGNAL_OUT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.SIGNAL_OUT OFF
*GET SYSTEM.STATUS.SIGNAL_OUT
```

5.154 SYSTEM.STATUS.STATE

TYPE: Register

METHODS: Get

VALUES: Enum:

- **INIT** - Amplifier is initializing
- **STANDBY** - Amplifier is in standby
- **STANDBY_FORCED** - Amplifier forced into standby (protection/over-temperature)
- **ON** - Amplifier is on

Current system state

Example:

```
$> "GET SYSTEM.STATUS.STATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.STATE INIT
*GET SYSTEM.STATUS.STATE
```

5.155 SYSTEM.STATUS.WIFI

TYPE: Register

METHODS: Get

VALUES: [String]

WiFi connection status

Example:

```
$> "GET SYSTEM.STATUS.WIFI" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.WIFI ""
*GET SYSTEM.STATUS.WIFI
```

5.156 VC-{VID}.NAME

TYPE: Register

METHODS: Get

PARAMS:

- **{VID}**: See paragraph Volume Control Channels

VALUES: [String] (Max Length 32 chars)

Volume control channel name

Example:

```
$> "GET VC-1.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+VC-1.NAME "Wall VC 1"
*GET VC-1.NAME
```

5.157 VC-{VID}.VALUE

TYPE: Register

METHODS: Get

PARAMS:

- **{VID}**: See paragraph Volume Control Channels

VALUES: [Float] (Range: 0.0 to 100.0 Percent)

Volume control value (0-100%)

Example:

```
$> "GET VC-1.VALUE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+VC-1.VALUE 50.00
*GET VC-1.VALUE
```

5.158 VC.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of volume control channels

Example:

```
$> "GET VC.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+VC.COUNT 4
*GET VC.COUNT
```

5.159 ZONE-{ZID}.COMPRESSOR.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0003 to 0.05 Sec)

Compressor attack time

Example:

```
$> "GET ZONE-A.COMPRESSOR.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.ATTACK 0.03
*GET ZONE-A.COMPRESSOR.ATTACK

$> "SET ZONE-A.COMPRESSOR.ATTACK 0.03" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.ATTACK 0.03
```

5.160 ZONE-{ZID}.COMPRESSOR.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable automatic compressor makeup gain

Example:

```
$> "GET ZONE-A.COMPRESSOR.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.AUTO 1
*GET ZONE-A.COMPRESSOR.AUTO

$> "SET ZONE-A.COMPRESSOR.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.AUTO 0
```

5.161 ZONE-{ZID}.COMPRESSOR.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Bypass zone compressor

Example:

```
$> "GET ZONE-A.COMPRESSOR.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.BYPASS 1
*GET ZONE-A.COMPRESSOR.BYPASS

$> "SET ZONE-A.COMPRESSOR.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.BYPASS 0
```

5.162 ZONE-{ZID}.COMPRESSOR.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0 to 1.0 Sec)

Compressor hold time

Example:

```
$> "GET ZONE-A.COMPRESSOR.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.HOLD 0.50
*GET ZONE-A.COMPRESSOR.HOLD

$> "SET ZONE-A.COMPRESSOR.HOLD 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.HOLD 0.50
```

5.163 ZONE-{ZID}.COMPRESSOR.KNEE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0 to 12.0 dB)

Compressor knee width

Example:

```
$> "GET ZONE-A.COMPRESSOR.KNEE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.KNEE 6.00
*GET ZONE-A.COMPRESSOR.KNEE

$> "SET ZONE-A.COMPRESSOR.KNEE 6.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.KNEE 6.00
```

5.164 ZONE-{ZID}.COMPRESSOR.RATIO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 1.0 to 50.0)

Compressor ratio

Example:

```
$> "GET ZONE-A.COMPRESSOR.RATIO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.RATIO 25.50
*GET ZONE-A.COMPRESSOR.RATIO

$> "SET ZONE-A.COMPRESSOR.RATIO 4.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.RATIO 4.0
```

5.165 ZONE-{ZID}.COMPRESSOR.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.001 to 1.0 Sec)

Compressor release time

Example:

```
$> "GET ZONE-A.COMPRESSOR.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.RELEASE 0.50
*GET ZONE-A.COMPRESSOR.RELEASE

$> "SET ZONE-A.COMPRESSOR.RELEASE 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.RELEASE 0.50
```

5.166 ZONE-{ZID}.COMPRESSOR.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -40.0 to 20.0 dB)

Compressor threshold

Example:

```
$> "GET ZONE-A.COMPRESSOR.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.THRESHOLD -20.00
*GET ZONE-A.COMPRESSOR.THRESHOLD

$> "SET ZONE-A.COMPRESSOR.THRESHOLD -30.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.THRESHOLD -30.00
```

5.167 ZONE-{ZID}.DUCK.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.001 to 0.2 Sec)

Ducker attack time

Example:

```
$> "GET ZONE-A.DUCK.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.ATTACK 0.10
*GET ZONE-A.DUCK.ATTACK

$> "SET ZONE-A.DUCK.ATTACK 0.10" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.ATTACK 0.10
```

5.168 ZONE-{ZID}.DUCK.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable automatic ducking

Example:

```
$> "GET ZONE-A.DUCK.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.AUTO 1
*GET ZONE-A.DUCK.AUTO

$> "SET ZONE-A.DUCK.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.AUTO 0
```

5.169 ZONE-{ZID}.DUCK.DEPTH

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -144.0 to 0.0 dB)

Ducking depth (attenuation amount)

Example:

```
$> "GET ZONE-A.DUCK.DEPTH" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.DEPTH -72.00
*GET ZONE-A.DUCK.DEPTH

$> "SET ZONE-A.DUCK.DEPTH -72.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.DEPTH -72.00
```

5.170 ZONE-{ZID}.DUCK.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0 to 10.0 Sec)

Ducker hold time

Example:

```
$> "GET ZONE-A.DUCK.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.HOLD 5.00
*GET ZONE-A.DUCK.HOLD

$> "SET ZONE-A.DUCK.HOLD 5.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.HOLD 5.00
```

5.171 ZONE-{ZID}.DUCK.MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: Enum:

- **OFF** - Ducking disabled
- **DUCKER** - Standard ducking mode
- **OVERRIDE** - Override mode - priority source replaces primary

Ducker mode

Example:

```
$> "GET ZONE-A.DUCK.MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.MODE OFF
*GET ZONE-A.DUCK.MODE

$> "SET ZONE-A.DUCK.MODE DUCKER" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.MODE DUCKER
```

5.172 ZONE-{ZID}.DUCK.OVERRIDE_GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -144.0 to 15.0 dB)

Override gain for primary source during ducking

Example:

```
$> "GET ZONE-A.DUCK.OVERRIDE_GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.OVERRIDE_GAIN -6.00
*GET ZONE-A.DUCK.OVERRIDE_GAIN

$> "SET ZONE-A.DUCK.OVERRIDE_GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.OVERRIDE_GAIN -12.00
```

5.173 ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable override gain during ducking

Example:

```
$> "GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 1
*GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE

$> "SET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 0
```

5.174 ZONE-{ZID}.DUCK.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.01 to 10.0 Sec)

Ducker release time

Example:

```
$> "GET ZONE-A.DUCK.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.RELEASE 5.00
*GET ZONE-A.DUCK.RELEASE

$> "SET ZONE-A.DUCK.RELEASE 5.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.RELEASE 5.00
```

5.175 ZONE-{ZID}.DUCK.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

Ducker trigger threshold

Example:

```
$> "GET ZONE-A.DUCK.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.THRESHOLD -20.00
*GET ZONE-A.DUCK.THRESHOLD

$> "SET ZONE-A.DUCK.THRESHOLD -30.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.THRESHOLD -30.00
```

5.176 ZONE-{ZID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -144.0 to 20.0 dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Zone signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+ZONE-A.DYN.SIGNAL -62.00
```

5.177 ZONE-{ZID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

NOTES: Read-Only if ZONE-{ZID}.GPIO_VC is set on zone. Actual range is [ZONE-{ZID}.GAIN_MIN, ZONE-{ZID}.GAIN_MAX]

Zone gain level

Example:

```
$> "GET ZONE-A.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN -6.00
*GET ZONE-A.GAIN

$> "SET ZONE-A.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GAIN -12.00
```

5.178 ZONE-{ZID}.GAIN_MAX

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

NOTES: Lower bound is the current ZONE-{ZID}.GAIN_MIN; ceiling is 0.0

Maximum gain limit

Example:

```
$> "GET ZONE-A.GAIN_MAX" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN_MAX -6.00
*GET ZONE-A.GAIN_MAX

$> "SET ZONE-A.GAIN_MAX -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GAIN_MAX -12.00
```

5.179 ZONE-{ZID}.GAIN_MIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

NOTES: Upper bound is the current ZONE-{ZID}.GAIN_MAX

Minimum gain limit

Example:

```
$> "GET ZONE-A.GAIN_MIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN_MIN -6.00
*GET ZONE-A.GAIN_MIN

$> "SET ZONE-A.GAIN_MIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GAIN_MIN -12.00
```

5.180 ZONE-{ZID}.GPIO_VC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Integer]

GPIO volume control assignment (VID, 0 for OFF)

Example:

```
$> "GET ZONE-A.GPIO_VC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GPIO_VC 1
*GET ZONE-A.GPIO_VC

$> "SET ZONE-A.GPIO_VC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GPIO_VC 100
```

5.181 ZONE-{ZID}.MUTE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Zone mute state

Example:

```
$> "GET ZONE-A.MUTE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.MUTE 1
*GET ZONE-A.MUTE

$> "SET ZONE-A.MUTE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.MUTE 0
```

5.182 ZONE-{ZID}.MUTE_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable mute functionality for zone

Example:

```
$> "GET ZONE-A.MUTE_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.MUTE_ENABLE 1
*GET ZONE-A.MUTE_ENABLE

$> "SET ZONE-A.MUTE_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.MUTE_ENABLE 0
```

5.183 ZONE-{ZID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [String] (Max Length 32 chars)

Zone name

Example:

```
$> "GET ZONE-A.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.NAME "Lobby"
*GET ZONE-A.NAME

$> "SET ZONE-A.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.NAME "New Name"
```

5.184 ZONE-{ZID}.PRIMARY_SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Integer]

NOTES: See {SID} Input Source definitions

Primary source input (SID value)

Example:

```
$> "GET ZONE-A.PRIMARY_SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIMARY_SRC 1
*GET ZONE-A.PRIMARY_SRC

$> "SET ZONE-A.PRIMARY_SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIMARY_SRC 100
```

5.185 ZONE-{ZID}.PRIORITY-{ZP}.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Boolean]

Enable automatic priority override

Example:

```
$> "GET ZONE-A.PRIORITY-2.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.AUTO 1
*GET ZONE-A.PRIORITY-2.AUTO

$> "SET ZONE-A.PRIORITY-2.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.AUTO 0
```

5.186 ZONE-{ZID}.PRIORITY-{ZP}.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Float] (Range: 0.001 to 10.0 Sec)

Priority hold time

Example:

```
$> "GET ZONE-A.PRIORITY-2.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.HOLD 5.00
*GET ZONE-A.PRIORITY-2.HOLD

$> "SET ZONE-A.PRIORITY-2.HOLD 5.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.HOLD 5.00
```

5.187 ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Float] (Range: -144.0 to 15.0 dB)

Override gain level for primary source

Example:

```
$> "GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.OVERRIDE_GAIN -6.00
*GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN
```

```
$> "SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN -12.00
```

5.188 ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Boolean]

Enable override gain adjustment

Example:

```
$> "GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
+ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE 1
*GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE

$> "SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE 0
```

5.189 ZONE-{ZID}.PRIORITY-{ZP}.SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Integer]

NOTES: See {SID} Input Source definitions

Priority source input (SID value)

Example:

```
$> "GET ZONE-A.PRIORITY-2.SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.SRC 1
*GET ZONE-A.PRIORITY-2.SRC

$> "SET ZONE-A.PRIORITY-2.SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.SRC 100
```

5.190 ZONE-{ZID}.PRIORITY-{ZP}.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Float] (Range: -80.0 to 0.0 dB)

Priority trigger threshold

Example:

```
$> "GET ZONE-A.PRIORITY-2.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.THRESHOLD -20.00
*GET ZONE-A.PRIORITY-2.THRESHOLD

$> "SET ZONE-A.PRIORITY-2.THRESHOLD -30.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.THRESHOLD -30.00
```

5.191 ZONE-{ZID}.PRIORITY_SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Integer]

NOTES: Valid SID value

Priority source input (legacy, use PRIORITY-{ZP}.SRC for FW 1.8+)

Example:

```
$> "GET ZONE-A.PRIORITY_SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY_SRC 1
*GET ZONE-A.PRIORITY_SRC

$> "SET ZONE-A.PRIORITY_SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY_SRC 100
```

5.192 ZONE-{ZID}.SRC-{IID}.ENABLED

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

Enable input source for zone mixing

Example:

```
$> "GET ZONE-A.SRC-100.ENABLED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.SRC-100.ENABLED 1
*GET ZONE-A.SRC-100.ENABLED

$> "SET ZONE-A.SRC-100.ENABLED 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.SRC-100.ENABLED 0
```

5.193 ZONE-{ZID}.STEREO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Stereo link zone with next zone

Example:

```
$> "GET ZONE-A.STEREO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.STEREO 1
*GET ZONE-A.STEREO

$> "SET ZONE-A.STEREO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.STEREO 0
```

5.194 ZONE.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of zones

Example:

```
$> "GET ZONE.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE.COUNT 4
*GET ZONE.COUNT
```

Contents

1 Introduction

Applies to: API 1.5 · firmware 1.8+ (incl. 2026.x).

1.1 Revision History

> From edition `2026.x` onward the document uses calendar-versioned editions (derived from the release tag); earlier `R5.x` rows are retained for history.
>
> The **API Version** column is the value the firmware's `API_VERSION` register reports. It has stayed `1.5` since firmware 1.5 even though registers were later added (Zone Priority in 1.8, `TILT_SHELF` in 2026.x) — so it is not a reliable freshness signal.

Revision	Date	Changed By	API Version	Firmware
2026.24.1	09/06-2026	MAM	1.5	1.8+
5.4	08/01-2026	MAM	1.5	1.8+
5.3	19/03-2025	MAM	1.5	1.8+
5.2	21/11-2023	MAM	1.5	1.6+
5.1	24/08-2023	MAM	1.5	1.5+
5	27/06-2023	MAM	1.5	1.5+
4	16/01-2023	MAM	1.4	1.4+
3	05/09-2022	MAM	1.3	1.3+
2	09/03-2022	MAM	1.2	1.2+
1	16/12-2021	MAM	1.1	1.1+
0	11/11-2021	MAM	1.0	1.0+

1.1.1 Edition 2026.24.1: Firmware-Accurate Rebuild

- First calendar-versioned edition (supersedes the unreleased R5.x revisions).
- Corrected register definitions — value ranges, types, and enumerations — to match the firmware exactly; many had drifted (limiter thresholds, EQ and crossover filter types, gain ranges, GPIO functions, power-on mode, polarity).
- Added a full **Wire-Protocol Reference**: error-line format and error-code catalog, all command verbs, connection/session limits and keep-alive, value encoding, and subscription semantics.
- Documented route signal/clip metering (`ROUT-{RID}.DYN.*`) and clarified that `*.DYN.*` registers are subscription-only (streamed on `SUBSCRIBE DYN`).
- Corrected EQ band counts (input 5, output 10, speaker 15) and the crossover / clip-limiter value sets.

1.1.2 Revision 5.4: Documentation Update

- Updated documentation tooling and build process
- Renamed "Advanced Registers" section to "Output Speaker Processing Registers"
- Auto-generated register reference from YAML schema

1.1.3 Revision 5.3: Firmware 1.8 Changes

- Added: 4 Zone Priority Override Registers (`ZONE-{ZID}.PRIORITY-{ZP}.*`)
- Added: {ZP} Zone Priority variable definition

1.1.3.1 Registers Added (R5.3)

Register Name
ZONE-{ZID}.PRIORITY-{ZP}.SRC
ZONE-{ZID}.PRIORITY-{ZP}.AUTO
ZONE-{ZID}.PRIORITY-{ZP}.THRESHOLD
ZONE-{ZID}.PRIORITY-{ZP}.HOLD
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN_ENABLE

1.1.4 Revision 5.2: Firmware 1.6 Changes

- Updated: Input Channels updated to 8 channels. See Input Channels.

1.1.5 Revision 5.1

Clarifications and documentation updates - No API Changes

1.1.6 Revision 5: API 1.5 Changes

- Added: Dante Info Registers (SYSTEM.DANTE.*)
- Updated: Input Channels updated with Dante. See Input Channels.
- Updated: Input Sources updated with Dante. See Input Sources.
- Updated: Added Mix to Output Route Sources. See Output Route Sources.

1.1.6.1 Registers Added (R5)

Register Name
SYSTEM.DANTE.SOFTWARE_VERSION
SYSTEM.DANTE.FIRMWARE_VERSION
SYSTEM.DANTE.IP
SYSTEM.DANTE.MAC
SYSTEM.DANTE.LINK_SPEED
SYSTEM.DANTE.AES67_ENABLED
SYSTEM.DANTE.DEVICE_NAME
SYSTEM.DANTE.ENCODING
SYSTEM.DANTE.SAMPLE_RATE
SYSTEM.DANTE.CLOCK_STATE
SYSTEM.DANTE.MUTE_STATE

1.1.6.2 Registers Modified (R5)

- Added Mixes to Route Sources
- Added PowerMode AUDIO_DSP to register SETUP.POWER.POWER_ON

1.1.7 Revision 4: API 1.4 Changes

- Added: Input HPF
- Added: 5-Band Input EQ
- Added: Mixes as Zone Primary Src.
- Added: Zone Priority Src for Zone
- Added: Option to disable Mute to Zone
- Added: Zone Ducker
- Added: Option to limit zone sources (Wall Controller Specific)
- Added: Option to select output SPDIF source
- Added: Bandwidth limitation for Pink Noise Generator
- Added: Sine Generator
- Removed: Generator cannot be disabled

1.1.7.1 Registers Added (R4)

Register Name
IN.EQ.COUNT
IN-{IID}.HPF_ENABLE
IN-{IID}.EQ.BYPASS
IN-{IID}.EQ-{EID}.TYPE
IN-{IID}.EQ-{EID}.GAIN
IN-{IID}.EQ-{EID}.FREQ
IN-{IID}.EQ-{EID}.Q
IN-{IID}.EQ-{EID}.BYPASS
ZONE-{ZID}.PRIORITY_SRC
ZONE-{ZID}.MUTE_ENABLE
ZONE-{ZID}.SRC-{IID}.ENABLED
ZONE-{ZID}.DUCK.MODE
ZONE-{ZID}.DUCK.AUTO
ZONE-{ZID}.DUCK.THRESHOLD
ZONE-{ZID}.DUCK.ATTACK
ZONE-{ZID}.DUCK.RELEASE
ZONE-{ZID}.DUCK.HOLD
ZONE-{ZID}.DUCK.OVERRIDE_GAIN
ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE
GENERATOR.TYPE
GENERATOR.SINE.FREQ
GENERATOR.PINK.LPF_ENABLE
GENERATOR.PINK.LPF_FREQ
GENERATOR.PINK.HPF_ENABLE
GENERATOR.PINK.HPF_FREQ
MIX.COUNT
MIX-{MID}.NAME

Register Name
MIX-{MID}.GAIN-{IID}
ROUT-{RID}.SRC
ROUT-{RID}.SRC_CHANNEL
ROUT-{RID}.GAIN

1.1.7.2 Registers Removed (R4) GENERATOR.ENABLE |

1.1.8 Revision 3: API 1.3 Changes

- Added: Output Gain
- Added: Clip Limiter Mode
- Added: Security Registers for WebPage Security
- Added: Input Gain Min + Input Gain Max
- Added: Analog Volume Control Value register as Value
- Remove: Analog Volume Control Volume register
- Update: Input Gain - Range increase from [-10, 10] to [-15, 15] dB
- Update: Zone Gain - when using Analog Volume Control
- Update: SETUP.LAN and SETUP.WIFI registers as readonly

1.1.8.1 Registers Added (R3)

Register Name
ZONE-{ZID}.GAIN_MIN
ZONE-{ZID}.GAIN_MAX
OUT-{OID}.GAIN
OUT-{OID}.CLIP_LIMITER.MODE
VC-{VID}.VALUE
SYSTEM.SECURITY.PASSWORD_ENABLE
SYSTEM.SECURITY.PASSWORD_HASH

1.1.8.2 Registers Updated (R3)

Register Name	Change
IN-{IID}.GAIN	Limits
ZONE-{ZID}.GAIN	Limits, more
SETUP.LAN.NETWORK_MODE	Read Only
SETUP.LAN.IP	Read Only
SETUP.LAN.MASK	Read Only
SETUP.LAN.GATEWAY	Read Only
SETUP.LAN.DNS1	Read Only
SETUP.LAN.DNS2	Read Only
SETUP.WIFI.ENABLE	Read Only
SETUP.WIFI.DISABLE_LAN_CONNECTED	Read Only

Register Name	Change
SETUP.WIFI.DISABLE_AFTER	Read Only
SETUP.WIFI.MODE	Read Only
SETUP.WIFI.AP_SSID	Read Only
SETUP.WIFI.AP_PASS	Read Only
SETUP.WIFI.STA_SSID	Read Only
SETUP.WIFI.STA_PASS	Read Only

1.1.8.3 Registers Removed (R3)

Register Name
VC-{VID}.VOLUME

1.1.9 Revision 2: API 1.2 Changes

- INC Command support for Input Gain
- Added Frequency parameter for SUBSCRIBE command
- Pink NoiseGenerator

1.1.9.1 Registers Added (R2)

Register Name
SETUP.DEVICE.SERIAL
SETUP.DEVICE.FIRMWARE
SETUP.DEVICE.FIRMWARE
SETUP.DEVICE.MAC
SETUP.DEVICE.WIFI_MAC
OUT-{OID}.LIMITER.AUTO
OUT-{OID}.LIMITER.THRESHOLD
OUT-{OID}.LIMITER.ATTACK
OUT-{OID}.LIMITER.RELEASE
OUT-{OID}.LIMITER.HOLD

1.1.9.2 Revision 1: Registers Added (R1)

Register Name
ZONE-{ZID}.COMPRESSOR.HOLD
OUT-{OID}.PRESET.NAME
OUT-{OID}.PRESET.ID
OUT-{OID}.PRESET.LOCKED
OUT-{OID}.POLARITY.PROTECTED
OUT-{OID}.OUTPUT_MODE.PROTECTED

Register Name
OUT-{OID}.SPEAKER_DELAY.PROTECTED
OUT-{OID}.LIMITER.PROTECTED
OUT-{OID}.SPEAKER_EQ.PROTECTED
OUT-{OID}.XR.PROTECTED
OUT-{OID}.FIR.PROTECTED
OUT-{OID}.PEAK_LIMITER.BYPASS
OUT-{OID}.PEAK_LIMITER.KNEE
OUT-{OID}.RMS_LIMITER.BYPASS
OUT-{OID}.RMS_LIMITER.THRESHOLD
OUT-{OID}.RMS_LIMITER.ATTACK
OUT-{OID}.RMS_LIMITER.RELEASE
OUT-{OID}.RMS_LIMITER.HOLD
OUT-{OID}.RMS_LIMITER.KNEE
OUT-{OID}.CLIP_LIMITER.BYPASS
OUT-{OID}.FIR.BYPASS
OUT-{OID}.FIR.TAPS

1.2 Connecting to the Amplifier

Out of the Box the amplifier is hard-coded with the Ethernet Address 192.168.64.100. It is also possible to connect to the amplifier using Wifi. Connect to the Wifi AP (SSID) and connect using the default IP address of 192.168.4.1.

1.3 Discovery (mDNS)

If the application requires the amplifier to have a dynamic IP address, it is possible to use mDNS to locate the amplifier.

The service type is: `_pasconnect._tcp`

The following properties is defined:

- **api_version** - the api version of the device
- **device_type** - the device type. For amplifiers this will always be `PasAmpControl`
- **model** - the model name of the device
- **software_id** - software id of the amplifier (Manufacturer and Model Specific)
- **hardware_id** - hardware id of the amplifier (Model ID)

Example (Avahi for Linux):

```
$> avahi-browse -t -r _pasconnect._tcp
+ enp0s8 IPv4 PASCAL-IP1252-2122-00031.local                    _pasconnect._tcp
 ↪  local
= enp0s8 IPv4 PASCAL-IP1252-2122-00031.local                    _pasconnect._tcp
 ↪  local
   hostname = [PASCAL-IP1252-2122-00031.local.local]
```

```
address = [192.168.64.100]
port = [80]
txt = ["api_version=1.5" "device_type=PasAmpControl" "manufacturer=Pascal Audio"
 ↪  "model=IP 125.2" "software_id=2" "model_id=2"]
```

1.4 Definitions

1.4.1 {IID} - Input Channels

The following input channels is defined for the amplifier.

- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **104** - Analog Input 5 *(8 channel version only)*
- **105** - Analog Input 6 *(8 channel version only)*
- **106** - Analog Input 7 *(8 channel version only)*
- **107** - Analog Input 8 *(8 channel version only)*
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1 *(Only for Dante Enabled Amplifiers)*
- **301** - Dante 2 *(Only for Dante Enabled Amplifiers)*
- **302** - Dante 3 *(Only for Dante Enabled Amplifiers)*
- **303** - Dante 4 *(Only for Dante Enabled Amplifiers)*
- **400** - Noise Generator

1.4.2 {MID} - Mix Channels

The following mix channels is defined for the amplifier.

- **1** - Mix 1
- **2** - Mix 2
- **3** - Mix 3 *(4 channel version only)*
- **4** - Mix 4 *(4 channel version only)*

1.4.3 {ZID} - Zones

The following zones is defined for the amplifier.

- **A** - Zone A
- **B** - Zone B
- **C** - Zone C *(4 channel version only)*
- **D** - Zone D *(4 channel version only)*

1.4.4 {ZP} - Zone Priorities

The following zone priority levels are defined for priority override.

- **2** - Priority 2 (Medium Priority)
- **3** - Priority 3 (Highest Priority)

1.4.5 {OID} - Output Channels

- **1** - Output 1
- **2** - Output 2
- **3** - Output 3 *(4 channel version only)*
- **4** - Output 4 *(4 channel version only)*

1.4.6 {RID} Output Route Channels

- **1** - Output 1
- **2** - Output 2
- **3** - Output 3 *(4 channel version only)*
- **4** - Output 4 *(4 channel version only)*

1.4.7 {SID} Input Source

The following input sources is defined for the amplifier.

- **0** - Unused Input (Silent)
- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1 *(Only for Dante Enabled Amplifiers)*
- **301** - Dante 2 *(Only for Dante Enabled Amplifiers)*
- **302** - Dante 3 *(Only for Dante Enabled Amplifiers)*
- **303** - Dante 4 *(Only for Dante Enabled Amplifiers)*
- **400** - Noise Generator
- **500** - Mix 1

- **501** - Mix 2
- **502** - Mix 3 *(Only for 4 Channel Amplifiers)*
- **503** - Mix 4 *(Only for 4 Channel Amplifiers)*

1.4.8 {RSID} Route Source

The following route sources is defined for the amplifier.

- **0** - Unused (Silent)
- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1
- **301** - Dante 2
- **302** - Dante 3
- **303** - Dante 4
- **500** - Mix A

- **501** - Mix B
- **502** - Mix C
- **503** - Mix D
- **1000** - Zone A
- **1001** - Zone B
- **1002** - Zone C
- **1003** - Zone D

1.4.9 {VID} Volume Controls

The following Volume Controls is defined for the amplifier.

- **0** - OFF
- **1** - GPIO PIN 4 Volume Control
- **2** - GPIO PIN 5 Volume Control
- **3** - GPIO PIN 6 Volume Control
- **4** - GPIO PIN 7 Volume Control

1.4.10 {EID} Input Equalizer Bands

The following Input Equalizer Bands (`IN-{IID}.EQ-{EID}.*`) are defined in the amplifier. The number of bands is model dependent; read the `IN.EQ.COUNT` register to discover the band count supported by a specific device. The current maximum is **5** bands.

- **1** - Input EQ Band 1
- **2** - Input EQ Band 2
- **3** - Input EQ Band 3
- **4** - Input EQ Band 4
- **5** - Input EQ Band 5

1.4.11 {OEID} Output Equalizer Bands

The following Output Equalizer Bands (`OUT-{OID}.EQ-{OEID}.*`) are defined in the amplifier. The number of bands is model dependent; read the `OUT.EQ.COUNT` register to discover the band count supported by a specific device. The current maximum is **10** bands.

- **1** - Output EQ Band 1
- **2** - Output EQ Band 2
- **3** - Output EQ Band 3
- **4** - Output EQ Band 4
- **5** - Output EQ Band 5
- **6** - Output EQ Band 6
- **7** - Output EQ Band 7
- **8** - Output EQ Band 8
- **9** - Output EQ Band 9
- **10** - Output EQ Band 10

1.4.12 {OSEID} Speaker Equalizer Bands

The following Speaker Equalizer Bands (`OUT-{OID}.SPEAKER_EQ-{OSEID}.*`) are defined in the amplifier. The number of bands is model dependent; read the `OUT.SPEAKER_EQ.COUNT` register to discover the band count supported by a specific device. The current maximum is **15** bands.

- **1** - Speaker EQ Band 1
- **2** - Speaker EQ Band 2
- **3** - Speaker EQ Band 3
- **4** - Speaker EQ Band 4
- **5** - Speaker EQ Band 5
- **6** - Speaker EQ Band 6
- **7** - Speaker EQ Band 7
- **8** - Speaker EQ Band 8
- **9** - Speaker EQ Band 9
- **10** - Speaker EQ Band 10
- **11** - Speaker EQ Band 11
- **12** - Speaker EQ Band 12
- **13** - Speaker EQ Band 13
- **14** - Speaker EQ Band 14
- **15** - Speaker EQ Band 15

1.4.13 Variable Types

- **Float** - Float format, delimited with '.'
- **Integer** - Normal integer
- **Enum** - Basically a string with a predefined set of options
- **String** - String. Might have limitations on number of characters. String values containing spaces must be enclosed in double-quotes.

2 API Endpoints

2.1 Raw Socket API

The Primary API in the amplifier is TCP Socket based (Port 7621) and is line based. That means every line is delimited by newline \n. Every line contains a single message. The API consists of 2 parts: a Command/Response interface and a Publish/Subscribe interface.

2.1.1 Ncat

Examples in documentation are based on Ncat (https://nmap.org/download.html). The specific syntax is PowerShell, but can easily be converted to bash for Linux.

Powershell style:

```
$> "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

bash style:

```
$> echo "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

2.2 WebSocket API

It is also possible to connect to the WebSocket based API in the amplifier. The syntax of commands and replies is exactly the same between the socket based API and the WebSocket based API, though a single WebSocket message might contain or return multiple lines of text, with each line containing a single message.

3 Command/Response

The Command/Response interface allows for Querying/Updating the registers in the amplifier and to execute commands.

To execute a command, send a websocket message with the command followed by newline.

- If the command executes successfully the response will be an asterisk followed by the command text.

```
$> "<COMMAND>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*<COMMAND>
```

- If the command fails the response is a single line beginning with a hash #, then the command that was sent (echoed verbatim), a pipe |, and a human-readable message: #<command>|<message>. The echoed command lets a client correlate the error with the request that caused it (useful when pipelining). See Error Handling for the full message catalog.

```
$> "GET IN-100.BOGUS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
#GET IN-100.BOGUS|E107: Unknown Parameter
```

- If the command returns data in form of registers the response will be:

```
$> "<COMMAND>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+<RESPONSE>
*<COMMAND>
```

3.1 Command Types

3.1.1 GET

Get value of amplifier register. The command supports wildcards.

Format:

```
"GET <REGISTER>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+<RESPONSE(s)>
*<COMMAND>
```

Example:

```
$> "GET IN-100.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.NAME "Analog 1"
*GET IN-100.NAME
```

```
"GET IN-*.NAME" | websocat -t -0 ws://192.168.64.100/ws
+IN-100.NAME "Analog 1"
+IN-101.NAME "Analog 2"
+IN-102.NAME "Analog 3"
+IN-103.NAME "Analog 4"
+IN-200.NAME "S/PDIF 1"
+IN-201.NAME "S/PDIF 1R"
+IN-400.NAME "Noise Generator"
*GET IN-*.NAME
```

3.1.2 SET

Set value in amplifier register. The command does not support wildcards!

Format:

```
"SET <REGISTER> <VALUE>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*<COMMAND>
```

Example

```
$> "SET IN-100.NAME ""Streamer""" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.NAME "Analog 1"
*SET IN-100.NAME "Streamer"
```

On a successful SET the firmware emits the changed register as a normal REG-topic update (`+<register> <value>`) to all REG subscribers, then sends the `*<command>` ack to the caller. A given client therefore sees the `+...` echo for its own SET only if it is itself subscribed to the REG topic; an unsubscribed client receives just the `*SET ...` ack. (For INC, by contrast, the caller is always sent the resulting `+<register> <value>` line before the ack.) See SUBSCRIBE for why `+...` lines and ack lines can interleave.

3.1.3 INC

Modifies the value in amplifier register by the amount specified in the command. The valus can be positive or negative. The command does not support wildcards!

Format:

```
"INC <REGISTER> <VALUE>" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+<REGISTER> <MODIFIED VALUE>
*<COMMAND>
```

Example

```
$> "INC ZONE-A.GAIN -5 | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN -5.00
*INC ZONE-A.GAIN -5
```

3.1.4 SUBSCRIBE

Subscribe to changes in all registers and dynamics. The subscribe command does not support subscriptions to individual registers. *This might change in a later release*.

The register changes will stream to the websocket after subscription…

```
$> "SUBSCRIBE" | ncat 192.168.64.100 7621 --no-shutdown
...
+IN-100.DYN.SIGNAL -49.9777
+IN-100.DYN.CLIP 0
+IN-101.DYN.SIGNAL -49.3077
+IN-101.DYN.CLIP 0
+IN-102.DYN.SIGNAL -99.7209
+IN-102.DYN.CLIP 0
...
*SUBSCRIBE
```

3.1.5 SUBSCRIBE <BLANK|*|REG|DYN> <FREQ> {#sec:subscribe}

Subscription is organised into two topics (firmware `protocol_connection.cpp`):

- **REG** - Register-change updates only. Pushed whenever a register value changes (from any client, the web UI, GPIO, presets, etc.).

- **DYN** - Dynamic/telemetry updates only. This is the channel that carries the subscription-only metering registers (`IN-{IID}.DYN.SIGNAL/.CLIP`, `ZONE-{ZID}.DYN.SIGNAL`, `OUT-{OID}.DYN.SIGNAL`, `ROUT-{RID}.DYN.SIGNAL/.CLIP`), which are not GET-able and only appear over this topic.
- **"*"** or **BLANK** (argument omitted) - Subscribes to both REG and DYN.

<FREQ> is the maximum push rate in Hz (updates per second) for that topic, applied per-topic on the server side:

- `0` (the default when omitted) means unlimited: every change/sample is pushed as it happens.
- Otherwise the server enforces a minimum interval of `1 / FREQ` seconds between pushes for that topic. So `1` = at most 1 update/second, `0.5` = at most 1 update every 2 seconds, `2` = at most 1 update every 0.5 s.
- The accepted range is `0..1000`; values are clamped to that range.

Frequency limiting is most useful for the high-rate DYN metering. Re-issuing SUBSCRIBE for a topic replaces its frequency setting.

Subscribe to changes in all registers or dynamic updates. The subscribe command does not support subscriptions to individual registers. *This might change in a later release*.

> **Push lines look exactly like GET data lines.** Both a subscription push and a line in a GET reply are formatted as `+<register> <value>`. The only thing that delimits a *response* to a command is the trailing `*<command>` ack line; subscription pushes carry no such delimiter and can arrive at any time, interleaved with command responses. A robust client must therefore not assume that the next `+...` line it reads belongs to the command it just sent. It should track outstanding commands and treat any `+...` line that arrives before the matching `*<command>` (or an unsolicited `+...` between command exchanges) as subscription data. When a value is changed by another client, subscribers receive the resulting `+<register> <value>` line on the REG topic.

The register changes will stream to the socket/websocket after subscription…

Example: Subscribe to All updates

```
$> "SUBSCRIBE" | ncat 192.168.64.100 7621 --no-shutdown
...
+IN-100.DYN.SIGNAL -49.9777
+IN-100.DYN.CLIP 0
+IN-101.DYN.SIGNAL -49.3077
+IN-101.DYN.CLIP 0
+IN-102.DYN.SIGNAL -99.7209
+IN-102.DYN.CLIP 0
...
*SUBSCRIBE
```

Example: Subscribe to Register only updates

```
$> "SUBSCRIBE REG" | ncat 192.168.64.100 7621 --no-shutdown
...
+ZONE-A.GAIN -5.00
...
*SUBSCRIBE REG
```

Example: Subscribe to Dynamic updates, but limit frequency to 1 Hz

```
$> "SUBSCRIBE DYN 1" | ncat 192.168.64.100 7621 --no-shutdown
...
+IN-100.DYN.SIGNAL -49.9777
+IN-100.DYN.CLIP 0
+IN-101.DYN.SIGNAL -49.3077
+IN-101.DYN.CLIP 0
+IN-102.DYN.SIGNAL -99.7209
+IN-102.DYN.CLIP 0
...
*SUBSCRIBE DYN 1
```

3.1.6 UNSUBSCRIBE <BLANK|*|REG|DYN>

- **REG** - Register Updates Only
- **DYN** - Dynamic updates Only
- **"BLANK"** - IF EMPTY - Both dynamic and register updates

Unsubscribe from the previous subscription. The parameter must match a previous subscription. An UNSUBSCRIBE with a blank value (unsubscribe all) will not cancel a subscription made to register updates only.

3.1.7 POWER_ON

TYPE: Command

Methods: POWER_ON

Example:

```
$> "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

3.1.8 POWER_OFF

TYPE: Command

Methods: POWER_OFF

Example:

```
$> "POWER_OFF" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_OFF
```

POWER_ON/POWER_OFF are only valid in certain Power-On modes; in an incompatible mode the firmware returns `E122: Error - Power On/Off not supported for current PowerOn Mode`.

3.1.9 PING

TYPE: Command

Application-level keepalive. The amplifier immediately replies with a PONG data line and then the usual *PING ack. This is the recommended way to confirm that a raw TCP control session (port 7621) is still alive, since that socket has no idle timeout of its own (see Connection & Session Lifecycle).

```
$> "PING" | ncat 192.168.64.100 7621 --no-shutdown -i 1
PONG
*PING
```

3.1.10 REBOOT

TYPE: Command

Reboots the amplifier. The firmware acks the command and then schedules the restart a moment later (~1 s), so the connection drops shortly after the *REBOOT ack. Audio is interrupted.

```
$> "REBOOT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*REBOOT
```

3.1.11 COPY_EQ <IN|SPK|OUT> <SRC> <DEST>

TYPE: Command

Copies an entire equalizer block from one channel to another. The first argument selects which EQ block:

- **IN** - input EQ; <SRC>/<DEST> are input channel IDs (e.g. `100`, `101`).
- **SPK** - speaker (output) EQ; <SRC>/<DEST> are 1-based output indices.
- **OUT** - output EQ; <SRC>/<DEST> are 1-based output indices.

On success the affected destination registers are returned (and pushed to REG subscribers). Copying onto a locked speaker preset fails with `E118: Setting is Protected Speaker Preset`; an unknown direction yields `E104: Invalid Parameter Value`.

```
$> "COPY_EQ IN 100 101" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-101.EQ.BYPASS 0
...
*COPY_EQ IN 100 101
```

3.1.12 RESET_EQ <IN|SPK|OUT> <DEST>

TYPE: Command

Resets the equalizer block on the destination channel to its defaults (flat). Direction and indexing match COPY_EQ.

```
$> "RESET_EQ OUT 1" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ.BYPASS 0
...
*RESET_EQ OUT 1
```

3.1.13 COPY_XR <SRC> <DEST>

TYPE: Command

Copies the crossover (XR) settings from one output to another. <SRC>/<DEST> are 1-based output indices. Copying onto a preset with a locked crossover fails with `E118`.

```
$> "COPY_XR 1 2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-2.XR.HPF_TYPE ...
...
*COPY_XR 1 2
```

3.1.14 RESET_XR <DEST>

TYPE: Command

Resets the crossover on the destination output (1-based index) to its defaults.

```
$> "RESET_XR 2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-2.XR.HPF_TYPE ...
...
*RESET_XR 2
```

3.1.15 FACTORY_DEFAULTS

TYPE: Command

Restores the entire device setup to factory defaults, re-applies it to the DSP and (if present) resets Dante, then re-applies network configuration. This closes every other open control connection (the connection that issued the command is kept and receives the full post-reset register dump); other clients must reconnect. Use with care.

```
$> "FACTORY_DEFAULTS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+...full register dump...
*FACTORY_DEFAULTS
```

3.2 Registers

Supported Registers for General Use

3.2.1 Base Registers

Register Name	Type	Access	Unit	Range / Values
API_VERSION	String	Get		
SYSTEM.STATUS.STATE	Enum	Get		{INIT, STANDBY, STANDBY_FORCED, ON}
SYSTEM.STATUS.SIGNAL_IN	Enum	Get		{OFF, NO_SIGNAL, SIGNAL, CLIP}
SYSTEM.STATUS.SIGNAL_OUT	Enum	Get		{OFF, NO_SIGNAL, SIGNAL, CLIP, FAULT}
SYSTEM.STATUS.LAN	String	Get		
SYSTEM.STATUS.WIFI	String	Get		

3.2.2 Device Information Registers

Register Name	Type	Access	Unit	Range / Values
SYSTEM.DEVICE.SWID	Integer	Get		
SYSTEM.DEVICE.HWID	Integer	Get		
SYSTEM.DEVICE.VENDOR_NAME	String[32]	Get		Max 32
SYSTEM.DEVICE.MODEL_NAME	String[32]	Get		Max 32
SYSTEM.DEVICE.SERIAL	String	Get		
SYSTEM.DEVICE.FIRMWARE	String	Get		
SYSTEM.DEVICE.FIRMWARE_DATE	String	Get		
SYSTEM.DEVICE.MAC	String	Get		
SYSTEM.DEVICE.WIFI_MAC	String	Get		

3.2.3 System Information Registers

Register Name	Type	Access	Unit	Range / Values
SETUP.SYSTEM.DEVICE_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.VENUE_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.CUSTOMER_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.ASSET_TAG	String[32]	Get, Set		Max 32

Register Name	Type	Access	Unit	Range / Values
SETUP.SYSTEM.INSTALLER_NAME	String[32]	Get, Set		Max 32
SETUP.SYSTEM.CONTACT_INFO	String[32]	Get, Set		Max 32
SETUP.SYSTEM.INSTALL_DATE	String[32]	Get, Set		Max 32
SETUP.SYSTEM.INSTALL_NOTES	String[256]	Get, Set		Max 256
SETUP.SYSTEM.LOCATING	Boolean	Get, Set		
SETUP.SYSTEM.CUSTOM1	String[8192]	Get, Set		Max 8192
SETUP.SYSTEM.CUSTOM2	String[8192]	Get, Set		Max 8192
SETUP.SYSTEM.CUSTOM3	String[8192]	Get, Set		Max 8192

3.2.4 Generator Registers

Register Name	Type	Access	Unit	Range / Values
GENERATOR.TYPE	Enum	Get, Set		{PINK, SINE}
GENERATOR.SINE.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
GENERATOR.PINK.LPF_ENABLE	Boolean	Get, Set		
GENERATOR.PINK.LPF_FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
GENERATOR.PINK.HPF_ENABLE	Boolean	Get, Set		
GENERATOR.PINK.HPF_FREQ	Float	Get, Set	Hz	[20.0, 20000.0]

3.2.5 Input Registers

Register Name	Type	Access	Unit	Range / Values
IN.COUNT	Integer	Get		
IN-{IID}.NAME	String[32]	Get, Set		Max 32
IN-{IID}.SENS	Enum	Get, Set		{14DBU, 4DBU, -10DBV, MIC}
IN-{IID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
IN-{IID}.STEREO	Boolean	Get, Set		
IN-{IID}.HPF_ENABLE	Boolean	Get, Set		
IN-{IID}.DYN.SIGNAL	Float	Subscribe	dB	
IN-{IID}.DYN.CLIP	Boolean	Subscribe		

3.2.6 Input Eq Registers

Register Name	Type	Access	Unit	Range / Values
IN.EQ.COUNT	Integer	Get		
IN-{IID}.EQ.BYPASS	Boolean	Get, Set		
IN-{IID}.EQ-{EID}.TYPE	Enum	Get, Set		{PARAMETRIC, LOWPASS_12, HIGHPASS_12, LOW_SHELF_Q, HIGH_SHELF_Q}
IN-{IID}.EQ-{EID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
IN-{IID}.EQ-{EID}.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
IN-{IID}.EQ-{EID}.Q	Float	Get, Set		[0.4, 30.0]

Register Name	Type	Access	Unit	Range / Values
IN-{IID}.EQ-{EID}.BYPASS	Boolean	Get, Set		

3.2.7 Zone Registers

Register Name	Type	Access	Unit	Range / Values
ZONE.COUNT	Integer	Get		
ZONE-{ZID}.NAME	String[32]	Get, Set		Max 32
ZONE-{ZID}.PRIMARY_SRC	Integer	Get, Set		Valid {SID}
ZONE-{ZID}.PRIORITY_SRC	Integer	Get, Set		Valid {SID}
ZONE-{ZID}.GAIN	Float	Get, Set, Inc	dB	[-80.0, 0.0]
ZONE-{ZID}.GAIN_MIN	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.GAIN_MAX	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.STEREO	Boolean	Get, Set		
ZONE-{ZID}.GPIO_VC	Integer	Get, Set		Valid {VID}, 0 = OFF
ZONE-{ZID}.MUTE	Boolean	Get, Set		
ZONE-{ZID}.MUTE_ENABLE	Boolean	Get, Set		
ZONE-{ZID}.SRC-{IID}.ENABLED	Boolean	Get, Set		
ZONE-{ZID}.DYN.SIGNAL	Float	Subscribe	dB	[-144.0, 20.0]

3.2.8 Zone Ducker Registers

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.DUCK.MODE	Enum	Get, Set		{OFF, DUCKER, OVERRIDE}
ZONE-{ZID}.DUCK.AUTO	Boolean	Get, Set		
ZONE-{ZID}.DUCK.THRESHOLD	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.DUCK.DEPTH	Float	Get, Set	dB	[-144.0, 0.0]
ZONE-{ZID}.DUCK.ATTACK	Float	Get, Set	Sec	[0.001, 0.2]
ZONE-{ZID}.DUCK.RELEASE	Float	Get, Set	Sec	[0.01, 10.0]
ZONE-{ZID}.DUCK.HOLD	Float	Get, Set	Sec	[0.0, 10.0]
ZONE-{ZID}.DUCK.OVERRIDE_GAIN	Float	Get, Set	dB	[-144.0, 15.0]
ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE	Boolean	Get, Set		

3.2.9 Zone Priority Registers

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.PRIORITY-{ZP}.SRC	Integer	Get, Set		Valid {SID}
ZONE-{ZID}.PRIORITY-{ZP}.AUTO	Boolean	Get, Set		

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.PRIORITY-{ZP}.THRESHOLD	Float	Get, Set	dB	[-80.0, 0.0]
ZONE-{ZID}.PRIORITY-{ZP}.HOLD	Float	Get, Set	Sec	[0.001, 10.0]
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN	Float	Get, Set	dB	[-144.0, 15.0]
ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN_ENABLE	Boolean	Get, Set		

3.2.10 Zone Compressor Registers

Register Name	Type	Access	Unit	Range / Values
ZONE-{ZID}.COMPRESSOR.AUTO	Boolean	Get, Set		
ZONE-{ZID}.COMPRESSOR.THRESHOLD	Float	Get, Set	dB	[-40.0, 20.0]
ZONE-{ZID}.COMPRESSOR.ATTACK	Float	Get, Set	Sec	[0.0003, 0.05]
ZONE-{ZID}.COMPRESSOR.RELEASE	Float	Get, Set	Sec	[0.001, 1.0]
ZONE-{ZID}.COMPRESSOR.HOLD	Float	Get, Set	Sec	[0.0, 1.0]
ZONE-{ZID}.COMPRESSOR.RATIO	Float	Get, Set		[1.0, 50.0]
ZONE-{ZID}.COMPRESSOR.KNEE	Float	Get, Set	dB	[0.0, 12.0]
ZONE-{ZID}.COMPRESSOR.BYPASS	Boolean	Get, Set		

3.2.11 Output Registers

Register Name	Type	Access	Unit	Range / Values
OUT.COUNT	Integer	Get		
OUT-{OID}.NAME	String[32]	Get, Set		Max 32
OUT-{OID}.GAIN	Float	Get, Set, Inc	dB	[-30.0, 15.0]
OUT-{OID}.MUTE	Boolean	Get, Set		
OUT-{OID}.SRC	String[1]	Get, Set		Max 1
OUT-{OID}.SRC_CHANNEL	Enum	Get, Set		{L, R, S}
OUT-{OID}.POLARITY	Integer	Get, Set		{-1, 1}
OUT-{OID}.OUTPUT_MODE	Enum	Get, Set		{OFF, 4R, 8R, 70V, 100V, BTL}
OUT-{OID}.OUTPUT_HIGHPASS	Float	Get, Set	Hz	[0.0, 1000.0]
OUT-{OID}.DYN.SIGNAL	Float	Subscribe	dB	[-144.0, 20.0]
OUT-{OID}.DYN.CLIP	Boolean	Subscribe		

3.2.12 Output Delay Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.DELAY.TIME	Float	Get, Set	Sec	[0.0, 0.1]
OUT-{OID}.DELAY.BYPASS	Boolean	Get, Set		

3.3 Output Speaker Processing Registers

The following registers are available for advanced configuration and speaker processing. Many of these are automatically configured by speaker presets and should not be modified unless necessary.

Speaker presets define the DSP configuration for specific speakers. These registers are read-only and indicate the current preset status.

3.3.1 Output Preset Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.PRESET.NAME	String[64]	Get		Max 64
OUT-{OID}.PRESET.ID	String	Get		
OUT-{OID}.PRESET.LOCKED	Boolean	Get		
OUT-{OID}.PRESET.CUSTOMIZED	Boolean	Get		

Speaker delay provides time alignment for speaker arrays. May be protected by preset.

3.3.2 Output Speaker Delay Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.SPEAKER_DELAY.TIME	Float	Get, Set	Sec	[0.0, 0.01]
OUT-{OID}.SPEAKER_DELAY.BYPASS	Boolean	Get, Set		
OUT-{OID}.SPEAKER_DELAY.PROTECTED	Boolean	Get		

Peak limiter protects speakers from transient peaks.

3.3.3 Output Peak Limiter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.PEAK_LIMITER.BYPASS	Boolean	Get, Set		
OUT-{OID}.PEAK_LIMITER.AUTO	Boolean	Get, Set		
OUT-{OID}.PEAK_LIMITER.THRESHOLD	Float	Get, Set	Vpeak	[1.0, 200.0]

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.PEAK_LIMITER.ATTACK	Float	Get, Set	Sec	[0.0003, 0.1]
OUT-{OID}.PEAK_LIMITER.RELEASE	Float	Get, Set	Sec	[0.004, 2.0]
OUT-{OID}.PEAK_LIMITER.HOLD	Float	Get, Set	Sec	[0.0, 1.0]
OUT-{OID}.PEAK_LIMITER.KNEE	Float	Get, Set	dB	[0.0, 6.0]

RMS limiter protects speakers from sustained thermal damage.

3.3.4 Output RMS Limiter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.RMS_LIMITER.BYPASS	Boolean	Get, Set		
OUT-{OID}.RMS_LIMITER.THRESHOLD	Float	Get, Set	Vpeak	[1.0, 150.0]
OUT-{OID}.RMS_LIMITER.ATTACK	Float	Get, Set	Sec	[0.01, 30.0]
OUT-{OID}.RMS_LIMITER.RELEASE	Float	Get, Set	Sec	[0.01, 30.0]
OUT-{OID}.RMS_LIMITER.HOLD	Float	Get, Set	Sec	[0.0, 1.0]
OUT-{OID}.RMS_LIMITER.KNEE	Float	Get, Set	dB	[0.0, 6.0]

Clip limiter prevents hard clipping at the amplifier output stage.

3.3.5 Output Clip Limiter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.CLIP_LIMITER.BYPASS	Boolean	Get, Set		
OUT-{OID}.CLIP_LIMITER.MODE	Enum	Get, Set		{NORMAL, FAST}
OUT-{OID}.LIMITER.PROTECTED	Boolean	Get		

User-adjustable output EQ for system tuning.

3.3.6 Output EQ Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.EQ.BYPASS	Boolean	Get, Set		

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.EQ-{OEID}.TYPE	Enum	Get, Set		{PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}
OUT-{OID}.EQ-{OEID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
OUT-{OID}.EQ-{OEID}.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.EQ-{OEID}.Q	Float	Get, Set		[0.4, 30.0]
OUT-{OID}.EQ-{OEID}.BYPASS	Boolean	Get, Set		

Factory-configured speaker EQ. Usually protected by preset.

3.3.7 Output Speaker EQ Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.SPEAKER_EQ.BYPASS	Boolean	Get, Set		
OUT-{OID}.SPEAKER_EQ-{OSEID}.TYPE	Enum	Get, Set		{PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}
OUT-{OID}.SPEAKER_EQ-{OSEID}.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]
OUT-{OID}.SPEAKER_EQ-{OSEID}.FREQ	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.SPEAKER_EQ-{OSEID}.Q	Float	Get, Set		[0.4, 30.0]
OUT-{OID}.SPEAKER_EQ-{OSEID}.BYPASS	Boolean	Get, Set		
OUT-{OID}.SPEAKER_EQ.PROTECTED	Boolean	Get		

Crossover filters for multi-way speaker systems. Usually protected by preset.

3.3.8 Output Crossover (XR) Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.XR.BYPASS	Boolean	Get, Set		
OUT-{OID}.XR.GAIN	Float	Get, Set, Inc	dB	[-15.0, 15.0]

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.XR.LOWPASS_TYPE	Enum	Get, Set		{OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}
OUT-{OID}.XR.LOWPASS_FREQUENCY	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.XR.HIGHPASS_TYPE	Enum	Get, Set		{OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}
OUT-{OID}.XR.HIGHPASS_FREQUENCY	Float	Get, Set	Hz	[20.0, 20000.0]
OUT-{OID}.XR.PROTECTED	Boolean	Get		

FIR filters for advanced speaker processing. Usually protected by preset.

3.3.9 Output FIR Filter Registers

Register Name	Type	Access	Unit	Range / Values
OUT-{OID}.FIR.BYPASS	Boolean	Get, Set		
OUT-{OID}.FIR.TAPS	Integer	Get		
OUT-{OID}.FIR.PROTECTED	Boolean	Get		

Direct routing for sending zone audio to specific outputs (advanced use).

3.3.10 Routing Registers

Register Name	Type	Access	Unit	Range / Values
ROUT-{RID}.SRC	Integer	Get, Set		Valid {RSID}
ROUT-{RID}.SRC_CHANNEL	Enum	Get, Set		{L, R, S}
ROUT-{RID}.GAIN	Float	Get, Set	dB	[-40.0, 20.0]
ROUT-{RID}.DYN.SIGNAL	Float	Subscribe	dB	
ROUT-{RID}.DYN.CLIP	Boolean	Subscribe		

External volume control inputs (GPIO wall plates).

3.3.11 Volume Control Registers

Register Name	Type	Access	Unit	Range / Values
VC.COUNT	Integer	Get		
VC-{VID}.NAME	String[32]	Get		Max 32
VC-{VID}.VALUE	Float	Get	Percent	[0.0, 100.0]

3.3.12 Power Management Registers

Register Name	Type	Access	Unit	Range / Values
SETUP.POWER.POWER_ON	Enum	Get, Set		{AUDIO, AUDIO_ECO, AUDIO_DSP, TRIGGER, TRIGGER_ECO, NETWORK}
SETUP.POWER.MUTE_TIME	Integer	Get, Set	Sec	[0, 3600]
SETUP.POWER.STANDBY_TIME	Integer	Get, Set	Sec	[0, 3600]

GPIO pins can be configured for various functions (status indication, triggers). The allowed values are per-pin; the Range / Values column lists each pin's accepted functions.

3.3.13 GPIO Configuration Registers

Register Name	Type	Access	Unit	Range / Values
SETUP.GPIO.PIN2	Enum	Get, Set		{OFF, STANDBY_NO, STANDBY_NC, MUTE_NO, MUTE_NC}
SETUP.GPIO.PIN4	Enum	Get, Set		{OFF, VOLUME_CONTROL}
SETUP.GPIO.PIN5	Enum	Get, Set		{OFF, VOLUME_CONTROL}
SETUP.GPIO.PIN6	Enum	Get, Set		{OFF, VOLUME_CONTROL, TRIGGER_12V_IN}
SETUP.GPIO.PIN7	Enum	Get, Set		{OFF, VOLUME_CONTROL, TRIGGER_12V_OUT}
SETUP.GPIO.PIN8	Enum	Get, Set		{VCC_3V3}

4 Wire-Protocol Reference

This chapter documents low-level behaviour of the control protocol that a third-party client must handle to be robust. Everything here is taken from the amplifier firmware.

4.1 Connection & Session Lifecycle

Two transports expose the identical line-based command/response protocol:

- **Raw TCP control socket** on port 7621. Plain TCP, line-based, each message terminated by a single newline `\n`. There is no idle timeout on this socket; a session stays open until the client closes it or the device reboots. Use PING/PONG to verify liveness.
- **WebSocket** at `ws://<ip>/ws` served on port 80 (the same HTTP server that serves the web UI). Commands and replies are byte-identical to the TCP socket, except a single WebSocket message may carry several `\n`-separated lines. The server sends a WebSocket PING control frame about every 3 seconds and applies a ~10-second idle timeout: if no frame is received within the timeout the server closes the connection. WebSocket clients must answer PINGs (most libraries auto-reply with PONG) or otherwise send traffic to stay connected.

Limits enforced by both transports (firmware `webserver.cpp`):

- The maximum received line length is 8192 + 128 bytes (`SOCKET_MAX_RX_LENGTH`). Two things protect against oversize input: the protocol engine rejects a command longer than that with `#<command>|Command too long`, and the session layer drops the whole connection if the unparsed receive buffer would exceed the limit (e.g. a line with no terminating newline). Keep individual lines well under 8 KB.
- The outgoing queue depth is 64 messages (`SOCKET_MAX_QUEUE_LENGTH`). If a client does not read fast enough and the server's per-connection TX queue fills, the server enters an overflow state, drops messages, and emits an overflow notice. The exact bytes differ by transport: the TCP socket sends `#|E0: TXQueue`

`Overflow – Messages dropped` and the WebSocket sends `#E0: TXQueue Overflow – Messages dropped`. The overflow state clears once the queue drains below half. A client that subscribes at high DYN rates must drain its socket promptly, or cap the rate with the <FREQ> argument.

4.2 Discovery & HTTP Endpoints

The TXT records and basic mDNS usage are covered under Discovery (mDNS). The driver-relevant detail: the advertised port is 80 (the HTTP/WebSocket port), and the amplifier does not advertise the 7621 control socket over mDNS. Three service types are registered (firmware `network_manager.cpp`): `_pasconnect._tcp` (the primary discovery service), `_http._tcp`, and `_rti._tcp` (for RTI controller integration, not needed by general API clients), all on port 80. A client should discover the device via `_pasconnect._tcp`, read its address, and then open the 7621 control socket (or the `/ws` WebSocket) by convention.

A small JSON-RPC 2.0 endpoint also exists at `POST /jrpc` on port 80. It exposes a single method, `reset_setup`, taking one boolean parameter `network` (whether to also reset the network configuration). It is used by the web UI; the line-based protocol described above is the intended integration surface for third-party clients.

4.3 Error Handling

A failed command produces exactly one line: `#<command>|<message>`, where <command> is the request echoed back verbatim and <message> is human-readable. Successful commands instead end with `*<command>`, and data is returned on `+...` lines (see Command/Response).

Most error messages begin with a stable `Exxx` code, but a client should treat the whole message string as the source of truth and not key solely on the number, because:

- some numbers are skipped (there is no `E113`, `E115`-`E117`, or `E125`);
- some numbers are reused for different conditions: `E110` covers both *Invalid IP address* and *Invalid Netmask address*, and `E126` covers both *Setting is not supported for non primary channel* and *Volume is controlled by GPIO volume control*;
- a few engine-level failures are reported without an `Exxx` code (see below).

4.3.1 Error Codes

The numbered codes (verbatim from firmware `protocol_error.h`):

Code	Message
E100	Channel is inactive (stereo)
E101	Invalid channel
E102	Invalid EQ band
E103	Invalid mix source
E104	Invalid Parameter Value
E105	Invalid number of arguments
E106	Unknown command
E107	Unknown Parameter
E108	No getter for parameter
E109	No setter for parameter
E110	Invalid IP address
E110	Invalid Netmask address
E111	No free slots

Code	Message
E112	Internal Error
E114	Invalid Password *(defined but never emitted - see Authentication)*
E118	Setting is Protected Speaker Preset
E119	String is too long
E120	String is invalid
E121	Limiter Error - Release must have higher value than attack
E122	Error - Power On/Off not supported for current PowerOn Mode
E123	Error - Limiter settings disabled due to Output Mode setting
E124	No INC for parameter
E126	Setting is not supported for non primary channel.
E126	Volume is controlled by GPIO volume control.
E127	No password is set.
E128	Mute disabled.
E129	Cannot Disable Primary Input.
E130	Source is Disabled.
E131	String is empty

In addition, the protocol engine and session layer can emit messages without an `Exxx` code (firmware `pas_protocol.cpp` / `webserver.cpp`):

Message	Meaning
Command Not Recognized	First token is not a known verb
Command too long	Command exceeded the 8192 + 128 byte limit
Parse Error	The line could not be tokenised (e.g. an unterminated quoted string)
Command Failed	The verb ran but reported failure (e.g. a register was not found)
E0: TXQueue Overflow - Messages dropped	Server TX queue overflowed; some pushes were dropped (delivered as `# | E0: ...` on the TCP socket and `#E0: ...` over WebSocket)

4.4 Value Encoding

Values on the wire (firmware `pas_protocol.cpp`, `protocol_helpers.h`) follow these rules; a client should parse leniently and emit canonically:

- **Booleans** are *returned* as `0` or `1`. On SET, the firmware also accepts the upper-case tokens `T`, `F`, `TRUE`, `FALSE` in addition to `1`/`0`. Lower-case forms are not accepted.
- **Floats** are formatted per-register with varying precision (anywhere from `%.1f` to `%0.5f` depending on the register), so the same numeric quantity may appear as `-5.00`, `1000.000`, or `0.50000`. Parse floats generically rather than assuming a fixed number of decimals; when sending, a plain decimal string is fine.
- **Strings** are *returned* double-quoted, e.g. `+IN-100.NAME "Analog 1"`. On SET, quotes are required only when the value contains spaces (the tokenizer otherwise splits on spaces); a single-word value may be sent unquoted. Values are UTF-8 and validated as such (invalid UTF-8 -> `E120`); over-length strings -> `E119`; an empty string where one is not allowed -> `E131`. There is no escape mechanism for a double-quote embedded inside a quoted string, so values cannot contain a `"` character.
- **Enums** are matched case-sensitively and exactly against the documented option set; an unrecognised token yields `E104`.

- Index-typed values are 1-based on the wire but 0-based internally. The firmware adds 1 when reporting an index value and subtracts 1 when parsing one (`add_index_response` / `get_value_as_index` / `parse_index`). So for an index register such as `ZONE.{ZID}.GPIO_VC`, the value the device reports and the value you send are the human-facing 1-based number; there is no need to compensate, but be aware the value is not a raw array offset and that 0-based assumptions will be off by one.

4.5 Authentication

The control protocol is unauthenticated. The registers `SYSTEM.SECURITY.PASSWORD_ENABLE` and `SYSTEM.SECURITY.PASSWORD_HASH` exist and the firmware stores them, but they are advisory only: the line-based protocol does not verify any password and gates no command behind one. The `E114: Invalid Password` code is defined in the firmware but is never thrown anywhere in the protocol path. Setting `PASSWORD_ENABLE` to true while no hash is stored returns `E127: No password is set.`, but this only guards the flag itself; it does not lock the socket. Treat any device reachable on port 7621 (or `/ws`) as fully controllable; enforce access control at the network layer.

5 Register Reference

5.1 API_VERSION

TYPE: Register

METHODS: Get

VALUES: [String]

API version string

Example:

```
$> "GET API_VERSION" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+API_VERSION "5.3"
*GET API_VERSION
```

5.2 GENERATOR.PINK.HPF_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable high-pass filter on pink noise

Example:

```
$> "GET GENERATOR.PINK.HPF_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.HPF_ENABLE 1
*GET GENERATOR.PINK.HPF_ENABLE

$> "SET GENERATOR.PINK.HPF_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.HPF_ENABLE 0
```

5.3 GENERATOR.PINK.HPF_FREQ

TYPE: Register

METHODS: Get, Set

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Pink noise high-pass filter frequency

Example:

```
$> "GET GENERATOR.PINK.HPF_FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.HPF_FREQ 1000.0
*GET GENERATOR.PINK.HPF_FREQ

$> "SET GENERATOR.PINK.HPF_FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.HPF_FREQ 2000.0
```

5.4 GENERATOR.PINK.LPF_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable low-pass filter on pink noise

Example:

```
$> "GET GENERATOR.PINK.LPF_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.LPF_ENABLE 1
*GET GENERATOR.PINK.LPF_ENABLE

$> "SET GENERATOR.PINK.LPF_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.LPF_ENABLE 0
```

5.5 GENERATOR.PINK.LPF_FREQ

TYPE: Register

METHODS: Get, Set

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Pink noise low-pass filter frequency

Example:

```
$> "GET GENERATOR.PINK.LPF_FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.PINK.LPF_FREQ 1000.0
*GET GENERATOR.PINK.LPF_FREQ

$> "SET GENERATOR.PINK.LPF_FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.PINK.LPF_FREQ 2000.0
```

5.6 GENERATOR.SINE.FREQ

TYPE: Register

METHODS: Get, Set

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Sine generator frequency

Example:

```
$> "GET GENERATOR.SINE.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.SINE.FREQ 1000.0
*GET GENERATOR.SINE.FREQ

$> "SET GENERATOR.SINE.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.SINE.FREQ 2000.0
```

5.7 GENERATOR.TYPE

TYPE: Register

METHODS: Get, Set

VALUES: Enum:

- **PINK** - Pink noise generator
- **SINE** - Sine wave generator

Generator type

Example:

```
$> "GET GENERATOR.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+GENERATOR.TYPE PINK
*GET GENERATOR.TYPE

$> "SET GENERATOR.TYPE SINE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET GENERATOR.TYPE SINE
```

5.8 IN-{IID}.DYN.CLIP

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Input clip indicator (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+IN-100.DYN.CLIP 1
```

5.9 IN-{IID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Float] (dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Input signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+IN-100.DYN.SIGNAL -12.00
```

5.10 IN-{IID}.EQ-{EID}.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Boolean]

Bypass individual EQ band

Example:

```
$> "GET IN-100.EQ-1.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.BYPASS 1
*GET IN-100.EQ-1.BYPASS

$> "SET IN-100.EQ-1.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.BYPASS 0
```

5.11 IN-{IID}.EQ-{EID}.FREQ

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

EQ band center frequency

Example:

```
$> "GET IN-100.EQ-1.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.FREQ 1000.0
*GET IN-100.EQ-1.FREQ

$> "SET IN-100.EQ-1.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.FREQ 2000.0
```

5.12 IN-{IID}.EQ-{EID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Float] (Range: -15.0 to 15.0 dB)

EQ band gain

Example:

```
$> "GET IN-100.EQ-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.GAIN -6.00
*GET IN-100.EQ-1.GAIN

$> "SET IN-100.EQ-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.GAIN -12.00
```

5.13 IN-{IID}.EQ-{EID}.Q

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: [Float] (Range: 0.4 to 30.0)

NOTES: Max Q 30 for PARAMETRIC/LOWPASS/HIGHPASS/NOTCH; 10 for shelf types

EQ band Q factor

Example:

```
$> "GET IN-100.EQ-1.Q" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.Q 1.0
*GET IN-100.EQ-1.Q

$> "SET IN-100.EQ-1.Q 2.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.Q 2.0
```

5.14 IN-{IID}.EQ-{EID}.TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels
- **{EID}**: See paragraph Equalizer Bands

VALUES: Enum:

- **PARAMETRIC** - Parametric EQ
- **LOWPASS_12** - 12 dB/octave low-pass filter
- **HIGHPASS_12** - 12 dB/octave high-pass filter
- **LOW_SHELF_Q** - Low shelf with Q control
- **HIGH_SHELF_Q** - High shelf with Q control

EQ band filter type

Example:

```
$> "GET IN-100.EQ-1.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ-1.TYPE PARAMETRIC
*GET IN-100.EQ-1.TYPE

$> "SET IN-100.EQ-1.TYPE LOWPASS_12" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ-1.TYPE LOWPASS_12
```

5.15 IN-{IID}.EQ.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

Bypass input EQ

Example:

```
$> "GET IN-100.EQ.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.EQ.BYPASS 1
*GET IN-100.EQ.BYPASS

$> "SET IN-100.EQ.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.EQ.BYPASS 0
```

5.16 IN-{IID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Float] (Range: -15.0 to 15.0 dB)

NOTES: Range is [-48.0, 0.0] for the Noise Generator input (IID 400)

Input channel gain

Example:

```
$> "GET IN-100.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.GAIN -6.00
*GET IN-100.GAIN

$> "SET IN-100.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.GAIN -12.00
```

5.17 IN-{IID}.HPF_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

High-pass filter enable

Example:

```
$> "GET IN-100.HPF_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.HPF_ENABLE 1
*GET IN-100.HPF_ENABLE

$> "SET IN-100.HPF_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.HPF_ENABLE 0
```

5.18 IN-{IID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [String] (Max Length 32 chars)

Input channel name

Example:

```
$> "GET IN-100.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.NAME "Mic 1"
*GET IN-100.NAME

$> "SET IN-100.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.NAME "New Name"
```

5.19 IN-{IID}.SENS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: Enum:

- **14DBU** - +14 dBu professional line level
- **4DBU** - +4 dBu professional line level
- **-10DBV** - -10 dBV consumer line level
- **MIC** - Microphone level input

Input sensitivity

Example:

```
$> "GET IN-100.SENS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.SENS 14DBU
*GET IN-100.SENS

$> "SET IN-100.SENS 4DBU" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.SENS 4DBU
```

5.20 IN-{IID}.STEREO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

Stereo link with next channel

Example:

```
$> "GET IN-100.STEREO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN-100.STEREO 1
*GET IN-100.STEREO

$> "SET IN-100.STEREO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET IN-100.STEREO 0
```

5.21 IN.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of input channels

Example:

```
$> "GET IN.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN.COUNT 4
*GET IN.COUNT
```

5.22 IN.EQ.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of EQ bands per input

Example:

```
$> "GET IN.EQ.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+IN.EQ.COUNT 4
*GET IN.EQ.COUNT
```

5.23 MIX-{MID}.GAIN-{IID}

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{MID}**: See paragraph Mix Channels
- **{IID}**: See paragraph Input Channels

VALUES: [Float] (Range: -144.0 to 0.0 dB)

Mix input gain for specific input channel

Example:

```
$> "GET MIX-1.GAIN-100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+MIX-1.GAIN-100 -6.00
*GET MIX-1.GAIN-100

$> "SET MIX-1.GAIN-100 -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET MIX-1.GAIN-100 -12.00
```

5.24 MIX-{MID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{MID}**: See paragraph Mix Channels

VALUES: [String] (Max Length 32 chars)

Mix channel name

Example:

```
$> "GET MIX-1.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+MIX-1.NAME "Mix A"
*GET MIX-1.NAME

$> "SET MIX-1.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET MIX-1.NAME "New Name"
```

5.25 MIX.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of mix channels

Example:

```
$> "GET MIX.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+MIX.COUNT 4
*GET MIX.COUNT
```

5.26 OUT-{OID}.CLIP_LIMITER.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass clip limiter

Example:

```
$> "GET OUT-1.CLIP_LIMITER.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.CLIP_LIMITER.BYPASS 1
*GET OUT-1.CLIP_LIMITER.BYPASS

$> "SET OUT-1.CLIP_LIMITER.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.CLIP_LIMITER.BYPASS 0
```

5.27 OUT-{OID}.CLIP_LIMITER.MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum {NORMAL, FAST}

Clip limiter mode

Example:

```
$> "GET OUT-1.CLIP_LIMITER.MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.CLIP_LIMITER.MODE NORMAL
*GET OUT-1.CLIP_LIMITER.MODE

$> "SET OUT-1.CLIP_LIMITER.MODE FAST" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.CLIP_LIMITER.MODE FAST
```

5.28 OUT-{OID}.DELAY.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass output delay

Example:

```
$> "GET OUT-1.DELAY.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.DELAY.BYPASS 1
*GET OUT-1.DELAY.BYPASS

$> "SET OUT-1.DELAY.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.DELAY.BYPASS 0
```

5.29 OUT-{OID}.DELAY.TIME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 0.1 Sec)

Output delay time

Example:

```
$> "GET OUT-1.DELAY.TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.DELAY.TIME 0.05
*GET OUT-1.DELAY.TIME

$> "SET OUT-1.DELAY.TIME 0.05" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.DELAY.TIME 0.05
```

5.30 OUT-{OID}.DYN.CLIP

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

NOTES: Planned — present in firmware but not yet streamed (commented out at system_manager.cpp:1276)

Output clip indicator (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+OUT-1.DYN.CLIP 1
```

5.31 OUT-{OID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: -144.0 to 20.0 dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Output signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+OUT-1.DYN.SIGNAL -62.00
```

5.32 OUT-{OID}.EQ-{OEID}.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Boolean]

Bypass individual output EQ band

Example:

```
$> "GET OUT-1.EQ-1.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.BYPASS 1
*GET OUT-1.EQ-1.BYPASS

$> "SET OUT-1.EQ-1.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.BYPASS 0
```

5.33 OUT-{OID}.EQ-{OEID}.FREQ

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Output EQ band center frequency

Example:

```
$> "GET OUT-1.EQ-1.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.FREQ 1000.0
*GET OUT-1.EQ-1.FREQ

$> "SET OUT-1.EQ-1.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.FREQ 2000.0
```

5.34 OUT-{OID}.EQ-{OEID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Float] (Range: -15.0 to 15.0 dB)

Output EQ band gain

Example:

```
$> "GET OUT-1.EQ-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.GAIN -6.00
*GET OUT-1.EQ-1.GAIN

$> "SET OUT-1.EQ-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.GAIN -12.00
```

5.35 OUT-{OID}.EQ-{OEID}.Q

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: [Float] (Range: 0.4 to 30.0)

Output EQ band Q factor

Example:

```
$> "GET OUT-1.EQ-1.Q" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.Q 1.0
*GET OUT-1.EQ-1.Q

$> "SET OUT-1.EQ-1.Q 2.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.Q 2.0
```

5.36 OUT-{OID}.EQ-{OEID}.TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OEID}**: See paragraph Output Equalizer Bands

VALUES: Enum {PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}

NOTES: TILT_SHELF added in 2026.x firmware

Output EQ band filter type

Example:

```
$> "GET OUT-1.EQ-1.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ-1.TYPE PARAMETRIC
*GET OUT-1.EQ-1.TYPE

$> "SET OUT-1.EQ-1.TYPE LOWPASS_6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ-1.TYPE LOWPASS_6
```

5.37 OUT-{OID}.EQ.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass output EQ

Example:

```
$> "GET OUT-1.EQ.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.EQ.BYPASS 1
*GET OUT-1.EQ.BYPASS

$> "SET OUT-1.EQ.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.EQ.BYPASS 0
```

5.38 OUT-{OID}.FIR.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass FIR filter

Example:

```
$> "GET OUT-1.FIR.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.FIR.BYPASS 1
*GET OUT-1.FIR.BYPASS

$> "SET OUT-1.FIR.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.FIR.BYPASS 0
```

5.39 OUT-{OID}.FIR.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

FIR filter settings are protected by preset

Example:

```
$> "GET OUT-1.FIR.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.FIR.PROTECTED 1
*GET OUT-1.FIR.PROTECTED
```

5.40 OUT-{OID}.FIR.TAPS

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Integer]

Number of FIR filter taps loaded

Example:

```
$> "GET OUT-1.FIR.TAPS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.FIR.TAPS 512
*GET OUT-1.FIR.TAPS
```

5.41 OUT-{OID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: -30.0 to 15.0 dB)

NOTES: INC is bounded to [-15.0, 15.0]

Output gain

Example:

```
$> "GET OUT-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.GAIN -6.00
*GET OUT-1.GAIN

$> "SET OUT-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.GAIN -12.00
```

5.42 OUT-{OID}.LIMITER.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Limiter settings are protected by preset

Example:

```
$> "GET OUT-1.LIMITER.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.LIMITER.PROTECTED 1
*GET OUT-1.LIMITER.PROTECTED
```

5.43 OUT-{OID}.MUTE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Output mute

Example:

```
$> "GET OUT-1.MUTE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.MUTE 1
*GET OUT-1.MUTE

$> "SET OUT-1.MUTE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.MUTE 0
```

5.44 OUT-{OID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String] (Max Length 32 chars)

Output channel name

Example:

```
$> "GET OUT-1.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.NAME "Main L"
*GET OUT-1.NAME

$> "SET OUT-1.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.NAME "New Name"
```

5.45 OUT-{OID}.OUTPUT_HIGHPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 1000.0 Hz)

Output high-pass filter frequency (0 = disabled)

Example:

```
$> "GET OUT-1.OUTPUT_HIGHPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.OUTPUT_HIGHPASS 500.00
*GET OUT-1.OUTPUT_HIGHPASS

$> "SET OUT-1.OUTPUT_HIGHPASS 500.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.OUTPUT_HIGHPASS 500.00
```

5.46 OUT-{OID}.OUTPUT_MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum:

- **OFF** - Output disabled
- **4R** - 4 ohm low-impedance mode (reported only, not settable)
- **8R** - 8 ohm low-impedance mode
- **70V** - 70V distributed audio mode

- **100V** - 100V distributed audio mode
- **BTL** - Bridge-tied load mode

NOTES: 4R is reported by GET but not settable; 70V/100V/BTL apply to the primary channel only (BTL requires hardware support)

Output mode

Example:

```
$> "GET OUT-1.OUTPUT_MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.OUTPUT_MODE OFF
*GET OUT-1.OUTPUT_MODE

$> "SET OUT-1.OUTPUT_MODE 4R" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.OUTPUT_MODE 4R
```

5.47 OUT-{OID}.OUTPUT_MODE.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Output mode setting is protected by preset

Example:

```
$> "GET OUT-1.OUTPUT_MODE.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.OUTPUT_MODE.PROTECTED 1
*GET OUT-1.OUTPUT_MODE.PROTECTED
```

5.48 OUT-{OID}.PEAK_LIMITER.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0003 to 0.1 Sec)

Peak limiter attack time

Example:

```
$> "GET OUT-1.PEAK_LIMITER.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.ATTACK 0.05
*GET OUT-1.PEAK_LIMITER.ATTACK

$> "SET OUT-1.PEAK_LIMITER.ATTACK 0.05" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.ATTACK 0.05
```

5.49 OUT-{OID}.PEAK_LIMITER.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Enable automatic peak limiter makeup gain

Example:

```
$> "GET OUT-1.PEAK_LIMITER.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.AUTO 1
*GET OUT-1.PEAK_LIMITER.AUTO

$> "SET OUT-1.PEAK_LIMITER.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.AUTO 0
```

5.50 OUT-{OID}.PEAK_LIMITER.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass peak limiter

Example:

```
$> "GET OUT-1.PEAK_LIMITER.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.BYPASS 1
*GET OUT-1.PEAK_LIMITER.BYPASS

$> "SET OUT-1.PEAK_LIMITER.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.BYPASS 0
```

5.51 OUT-{OID}.PEAK_LIMITER.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 1.0 Sec)

Peak limiter hold time

Example:

```
$> "GET OUT-1.PEAK_LIMITER.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.HOLD 0.50
*GET OUT-1.PEAK_LIMITER.HOLD

$> "SET OUT-1.PEAK_LIMITER.HOLD 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.HOLD 0.50
```

5.52 OUT-{OID}.PEAK_LIMITER.KNEE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 6.0 dB)

Peak limiter knee width

Example:

```
$> "GET OUT-1.PEAK_LIMITER.KNEE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.KNEE 3.00
*GET OUT-1.PEAK_LIMITER.KNEE

$> "SET OUT-1.PEAK_LIMITER.KNEE 3.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.KNEE 3.00
```

5.53 OUT-{OID}.PEAK_LIMITER.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.004 to 2.0 Sec)

Peak limiter release time

Example:

```
$> "GET OUT-1.PEAK_LIMITER.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.RELEASE 1.00
*GET OUT-1.PEAK_LIMITER.RELEASE

$> "SET OUT-1.PEAK_LIMITER.RELEASE 1.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.RELEASE 1.00
```

5.54 OUT-{OID}.PEAK_LIMITER.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 1.0 to 200.0 Vpeak)

Peak limiter threshold (peak volts)

Example:

```
$> "GET OUT-1.PEAK_LIMITER.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PEAK_LIMITER.THRESHOLD -20.00
*GET OUT-1.PEAK_LIMITER.THRESHOLD

$> "SET OUT-1.PEAK_LIMITER.THRESHOLD 100.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.PEAK_LIMITER.THRESHOLD 100.50
```

5.55 OUT-{OID}.POLARITY

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Integer]

Output polarity (1 = normal, -1 = inverted)

Example:

```
$> "GET OUT-1.POLARITY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.POLARITY 1
*GET OUT-1.POLARITY

$> "SET OUT-1.POLARITY 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.POLARITY 100
```

5.56 OUT-{OID}.POLARITY.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Polarity setting is protected by preset

Example:

```
$> "GET OUT-1.POLARITY.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.POLARITY.PROTECTED 1
*GET OUT-1.POLARITY.PROTECTED
```

5.57 OUT-{OID}.PRESET.CUSTOMIZED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Preset has been customized from factory settings

Example:

```
$> "GET OUT-1.PRESET.CUSTOMIZED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.CUSTOMIZED 1
*GET OUT-1.PRESET.CUSTOMIZED
```

5.58 OUT-{OID}.PRESET.ID

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String]

Active speaker preset ID

Example:

```
$> "GET OUT-1.PRESET.ID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.ID ""
*GET OUT-1.PRESET.ID
```

5.59 OUT-{OID}.PRESET.LOCKED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Preset is locked (cannot be modified)

Example:

```
$> "GET OUT-1.PRESET.LOCKED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.LOCKED 1
*GET OUT-1.PRESET.LOCKED
```

5.60 OUT-{OID}.PRESET.NAME

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String] (Max Length 64 chars)

Active speaker preset name

Example:

```
$> "GET OUT-1.PRESET.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.PRESET.NAME "Factory Preset"
*GET OUT-1.PRESET.NAME
```

5.61 OUT-{OID}.RMS_LIMITER.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.01 to 30.0 Sec)

RMS limiter attack time

Example:

```
$> "GET OUT-1.RMS_LIMITER.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.ATTACK 15.01
*GET OUT-1.RMS_LIMITER.ATTACK

$> "SET OUT-1.RMS_LIMITER.ATTACK 15.01" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.ATTACK 15.01
```

5.62 OUT-{OID}.RMS_LIMITER.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass RMS limiter

Example:

```
$> "GET OUT-1.RMS_LIMITER.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.BYPASS 1
*GET OUT-1.RMS_LIMITER.BYPASS

$> "SET OUT-1.RMS_LIMITER.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.BYPASS 0
```

5.63 OUT-{OID}.RMS_LIMITER.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 1.0 Sec)

RMS limiter hold time

Example:

```
$> "GET OUT-1.RMS_LIMITER.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.HOLD 0.50
*GET OUT-1.RMS_LIMITER.HOLD

$> "SET OUT-1.RMS_LIMITER.HOLD 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.HOLD 0.50
```

5.64 OUT-{OID}.RMS_LIMITER.KNEE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 6.0 dB)

RMS limiter knee width

Example:

```
$> "GET OUT-1.RMS_LIMITER.KNEE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.KNEE 3.00
*GET OUT-1.RMS_LIMITER.KNEE

$> "SET OUT-1.RMS_LIMITER.KNEE 3.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.KNEE 3.00
```

5.65 OUT-{OID}.RMS_LIMITER.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.01 to 30.0 Sec)

RMS limiter release time

Example:

```
$> "GET OUT-1.RMS_LIMITER.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.RELEASE 15.01
*GET OUT-1.RMS_LIMITER.RELEASE

$> "SET OUT-1.RMS_LIMITER.RELEASE 15.01" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.RELEASE 15.01
```

5.66 OUT-{OID}.RMS_LIMITER.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 1.0 to 150.0 Vpeak)

RMS limiter threshold (peak volts)

Example:

```
$> "GET OUT-1.RMS_LIMITER.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.RMS_LIMITER.THRESHOLD -20.00
*GET OUT-1.RMS_LIMITER.THRESHOLD

$> "SET OUT-1.RMS_LIMITER.THRESHOLD 75.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.RMS_LIMITER.THRESHOLD 75.50
```

5.67 OUT-{OID}.SPEAKER_DELAY.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass speaker delay

Example:

```
$> "GET OUT-1.SPEAKER_DELAY.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_DELAY.BYPASS 1
*GET OUT-1.SPEAKER_DELAY.BYPASS

$> "SET OUT-1.SPEAKER_DELAY.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_DELAY.BYPASS 0
```

5.68 OUT-{OID}.SPEAKER_DELAY.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Speaker delay settings are protected by preset

Example:

```
$> "GET OUT-1.SPEAKER_DELAY.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_DELAY.PROTECTED 1
*GET OUT-1.SPEAKER_DELAY.PROTECTED
```

5.69 OUT-{OID}.SPEAKER_DELAY.TIME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 0.0 to 0.01 Sec)

Speaker delay time for time alignment

Example:

```
$> "GET OUT-1.SPEAKER_DELAY.TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_DELAY.TIME 0.01
*GET OUT-1.SPEAKER_DELAY.TIME

$> "SET OUT-1.SPEAKER_DELAY.TIME 0.01" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_DELAY.TIME 0.01
```

5.70 OUT-{OID}.SPEAKER_EQ-{OSEID}.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Boolean]

Bypass individual speaker EQ band

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.BYPASS 1
*GET OUT-1.SPEAKER_EQ-1.BYPASS

$> "SET OUT-1.SPEAKER_EQ-1.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.BYPASS 0
```

5.71 OUT-{OID}.SPEAKER_EQ-{OSEID}.FREQ

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Speaker EQ band center frequency

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.FREQ" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.FREQ 1000.0
*GET OUT-1.SPEAKER_EQ-1.FREQ

$> "SET OUT-1.SPEAKER_EQ-1.FREQ 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.FREQ 2000.0
```

5.72 OUT-{OID}.SPEAKER_EQ-{OSEID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Float] (Range: -15.0 to 15.0 dB)

Speaker EQ band gain

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.GAIN -6.00
*GET OUT-1.SPEAKER_EQ-1.GAIN

$> "SET OUT-1.SPEAKER_EQ-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.GAIN -12.00
```

5.73 OUT-{OID}.SPEAKER_EQ-{OSEID}.Q

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: [Float] (Range: 0.4 to 30.0)

Speaker EQ band Q factor

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.Q" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.Q 1.0
*GET OUT-1.SPEAKER_EQ-1.Q

$> "SET OUT-1.SPEAKER_EQ-1.Q 2.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.Q 2.0
```

5.74 OUT-{OID}.SPEAKER_EQ-{OSEID}.TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels
- **{OSEID}**: See paragraph Speaker Equalizer Bands

VALUES: Enum {PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2, TILT_SHELF}

Speaker EQ band filter type

Example:

```
$> "GET OUT-1.SPEAKER_EQ-1.TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ-1.TYPE PARAMETRIC
*GET OUT-1.SPEAKER_EQ-1.TYPE

$> "SET OUT-1.SPEAKER_EQ-1.TYPE LOWPASS_6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ-1.TYPE LOWPASS_6
```

5.75 OUT-{OID}.SPEAKER_EQ.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass speaker EQ

Example:

```
$> "GET OUT-1.SPEAKER_EQ.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ.BYPASS 1
*GET OUT-1.SPEAKER_EQ.BYPASS

$> "SET OUT-1.SPEAKER_EQ.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SPEAKER_EQ.BYPASS 0
```

5.76 OUT-{OID}.SPEAKER_EQ.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Speaker EQ settings are protected by preset

Example:

```
$> "GET OUT-1.SPEAKER_EQ.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SPEAKER_EQ.PROTECTED 1
*GET OUT-1.SPEAKER_EQ.PROTECTED
```

5.77 OUT-{OID}.SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [String] (Max Length 1 chars)

Output source zone (ZID)

Example:

```
$> "GET OUT-1.SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SRC ""
*GET OUT-1.SRC

$> "SET OUT-1.SRC "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SRC "New Name"
```

5.78 OUT-{OID}.SRC_CHANNEL

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum:

- **L** - Left channel
- **R** - Right channel
- **S** - Summed mono (L+R)

Output source channel (Left, Right, Summed)

Example:

```
$> "GET OUT-1.SRC_CHANNEL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.SRC_CHANNEL L
*GET OUT-1.SRC_CHANNEL

$> "SET OUT-1.SRC_CHANNEL R" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.SRC_CHANNEL R
```

5.79 OUT-{OID}.XR.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Bypass crossover filters

Example:

```
$> "GET OUT-1.XR.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.BYPASS 1
*GET OUT-1.XR.BYPASS

$> "SET OUT-1.XR.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.BYPASS 0
```

5.80 OUT-{OID}.XR.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: -15.0 to 15.0 dB)

Crossover section gain

Example:

```
$> "GET OUT-1.XR.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.GAIN -6.00
*GET OUT-1.XR.GAIN

$> "SET OUT-1.XR.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.GAIN -12.00
```

5.81 OUT-{OID}.XR.HIGHPASS_FREQUENCY

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

High-pass crossover frequency

Example:

```
$> "GET OUT-1.XR.HIGHPASS_FREQUENCY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.HIGHPASS_FREQUENCY 1000.0
*GET OUT-1.XR.HIGHPASS_FREQUENCY

$> "SET OUT-1.XR.HIGHPASS_FREQUENCY 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.HIGHPASS_FREQUENCY 2000.0
```

5.82 OUT-{OID}.XR.HIGHPASS_TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum {OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}

High-pass crossover filter type

Example:

```
$> "GET OUT-1.XR.HIGHPASS_TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.HIGHPASS_TYPE OFF
*GET OUT-1.XR.HIGHPASS_TYPE

$> "SET OUT-1.XR.HIGHPASS_TYPE BUT6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.HIGHPASS_TYPE BUT6
```

5.83 OUT-{OID}.XR.LOWPASS_FREQUENCY

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Float] (Range: 20.0 to 20000.0 Hz)

Low-pass crossover frequency

Example:

```
$> "GET OUT-1.XR.LOWPASS_FREQUENCY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.LOWPASS_FREQUENCY 1000.0
*GET OUT-1.XR.LOWPASS_FREQUENCY

$> "SET OUT-1.XR.LOWPASS_FREQUENCY 2000.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.LOWPASS_FREQUENCY 2000.0
```

5.84 OUT-{OID}.XR.LOWPASS_TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: Enum {OFF, BUT6, BUT12, BUT18, BUT24, BUT36, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}

Low-pass crossover filter type

Example:

```
$> "GET OUT-1.XR.LOWPASS_TYPE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.LOWPASS_TYPE OFF
*GET OUT-1.XR.LOWPASS_TYPE

$> "SET OUT-1.XR.LOWPASS_TYPE BUT6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET OUT-1.XR.LOWPASS_TYPE BUT6
```

5.85 OUT-{OID}.XR.PROTECTED

TYPE: Register

METHODS: Get

PARAMS:

- **{OID}**: See paragraph Output Channels

VALUES: [Boolean]

Crossover settings are protected by preset

Example:

```
$> "GET OUT-1.XR.PROTECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT-1.XR.PROTECTED 1
*GET OUT-1.XR.PROTECTED
```

5.86 OUT.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of output channels

Example:

```
$> "GET OUT.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT.COUNT 4
*GET OUT.COUNT
```

5.87 OUT.EQ.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of output EQ bands

Example:

```
$> "GET OUT.EQ.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT.EQ.COUNT 4
*GET OUT.EQ.COUNT
```

5.88 OUT.SPEAKER_EQ.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of speaker EQ bands

Example:

```
$> "GET OUT.SPEAKER_EQ.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+OUT.SPEAKER_EQ.COUNT 4
*GET OUT.SPEAKER_EQ.COUNT
```

5.89 ROUT-{RID}.DYN.CLIP

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Boolean]

NOTES: Subscription only — streamed on SUBSCRIBE DYN

Route clip indicator (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+ROUT-1.DYN.CLIP 1
```

5.90 ROUT-{RID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Float] (dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN

Route signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+ROUT-1.DYN.SIGNAL -12.00
```

5.91 ROUT-{RID}.GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Float] (Range: -40.0 to 20.0 dB)

Routing gain level

Example:

```
$> "GET ROUT-1.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ROUT-1.GAIN -6.00
*GET ROUT-1.GAIN

$> "SET ROUT-1.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ROUT-1.GAIN -12.00
```

5.92 ROUT-{RID}.SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: [Integer]

NOTES: See {RSID} Route Source definitions

Routing source (RSID value)

Example:

```
$> "GET ROUT-1.SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ROUT-1.SRC 1
*GET ROUT-1.SRC

$> "SET ROUT-1.SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ROUT-1.SRC 100
```

5.93 ROUT-{RID}.SRC_CHANNEL

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{RID}**: See paragraph Routing Channels

VALUES: Enum {L, R, S}

Routing source channel (Left, Right, Summed)

Example:

```
$> "GET ROUT-1.SRC_CHANNEL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ROUT-1.SRC_CHANNEL L
*GET ROUT-1.SRC_CHANNEL

$> "SET ROUT-1.SRC_CHANNEL R" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ROUT-1.SRC_CHANNEL R
```

5.94 SETUP.GPIO.PIN2

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, STANDBY_NO, STANDBY_NC, MUTE_NO, MUTE_NC}

GPIO Pin 2 function

Example:

```
$> "GET SETUP.GPIO.PIN2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN2 OFF
*GET SETUP.GPIO.PIN2

$> "SET SETUP.GPIO.PIN2 STANDBY_NO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN2 STANDBY_NO
```

5.95 SETUP.GPIO.PIN4

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL}

GPIO Pin 4 function

Example:

```
$> "GET SETUP.GPIO.PIN4" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN4 OFF
*GET SETUP.GPIO.PIN4

$> "SET SETUP.GPIO.PIN4 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN4 VOLUME_CONTROL
```

5.96 SETUP.GPIO.PIN5

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL}

GPIO Pin 5 function

Example:

```
$> "GET SETUP.GPIO.PIN5" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN5 OFF
*GET SETUP.GPIO.PIN5

$> "SET SETUP.GPIO.PIN5 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN5 VOLUME_CONTROL
```

5.97 SETUP.GPIO.PIN6

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL, TRIGGER_12V_IN}

GPIO Pin 6 function

Example:

```
$> "GET SETUP.GPIO.PIN6" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN6 OFF
*GET SETUP.GPIO.PIN6

$> "SET SETUP.GPIO.PIN6 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN6 VOLUME_CONTROL
```

5.98 SETUP.GPIO.PIN7

TYPE: Register

METHODS: Get, Set

VALUES: Enum {OFF, VOLUME_CONTROL, TRIGGER_12V_OUT}

GPIO Pin 7 function

Example:

```
$> "GET SETUP.GPIO.PIN7" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN7 OFF
*GET SETUP.GPIO.PIN7

$> "SET SETUP.GPIO.PIN7 VOLUME_CONTROL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN7 VOLUME_CONTROL
```

5.99 SETUP.GPIO.PIN8

TYPE: Register

METHODS: Get, Set

VALUES: Enum {VCC_3V3}

GPIO Pin 8 function

Example:

```
$> "GET SETUP.GPIO.PIN8" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.GPIO.PIN8 VCC_3V3
*GET SETUP.GPIO.PIN8

$> "SET SETUP.GPIO.PIN8 VCC_3V3" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.GPIO.PIN8 VCC_3V3
```

5.100 SETUP.LAN.DNS1

TYPE: Register

METHODS: Get

VALUES: [String]

Primary DNS server

Example:

```
$> "GET SETUP.LAN.DNS1" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.DNS1 ""
*GET SETUP.LAN.DNS1
```

5.101 SETUP.LAN.DNS2

TYPE: Register

METHODS: Get

VALUES: [String]

Secondary DNS server

Example:

```
$> "GET SETUP.LAN.DNS2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.DNS2 ""
*GET SETUP.LAN.DNS2
```

5.102 SETUP.LAN.GATEWAY

TYPE: Register

METHODS: Get

VALUES: [String]

LAN gateway address

Example:

```
$> "GET SETUP.LAN.GATEWAY" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.GATEWAY ""
*GET SETUP.LAN.GATEWAY
```

5.103 SETUP.LAN.IP

TYPE: Register

METHODS: Get

VALUES: [String]

LAN IP address

Example:

```
$> "GET SETUP.LAN.IP" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.IP "192.168.1.100"
*GET SETUP.LAN.IP
```

5.104 SETUP.LAN.MASK

TYPE: Register

METHODS: Get

VALUES: [String]

LAN subnet mask

Example:

```
$> "GET SETUP.LAN.MASK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.MASK ""
*GET SETUP.LAN.MASK
```

5.105 SETUP.LAN.NETWORK_MODE

TYPE: Register

METHODS: Get

VALUES: Enum:

- **STATIC** - Static IP configuration
- **DHCP** - Dynamic IP via DHCP

Network configuration mode

Example:

```
$> "GET SETUP.LAN.NETWORK_MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.LAN.NETWORK_MODE STATIC
*GET SETUP.LAN.NETWORK_MODE
```

5.106 SETUP.POWER.MUTE_TIME

TYPE: Register

METHODS: Get, Set

VALUES: [Integer] (Range: 0 to 3600 Sec)

Mute time after power on

Example:

```
$> "GET SETUP.POWER.MUTE_TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.POWER.MUTE_TIME 1800
*GET SETUP.POWER.MUTE_TIME

$> "SET SETUP.POWER.MUTE_TIME 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.POWER.MUTE_TIME 100
```

5.107 SETUP.POWER.POWER_ON

TYPE: Register

METHODS: Get, Set

VALUES: Enum {AUDIO, AUDIO_ECO, AUDIO_DSP, TRIGGER, TRIGGER_ECO, NETWORK}

Power-on / standby behaviour mode

Example:

```
$> "GET SETUP.POWER.POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.POWER.POWER_ON AUDIO
*GET SETUP.POWER.POWER_ON

$> "SET SETUP.POWER.POWER_ON AUDIO_ECO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.POWER.POWER_ON AUDIO_ECO
```

5.108 SETUP.POWER.STANDBY_TIME

TYPE: Register

METHODS: Get, Set

VALUES: [Integer] (Range: 0 to 3600 Sec)

Auto standby after no signal timeout (0 = disabled)

Example:

```
$> "GET SETUP.POWER.STANDBY_TIME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.POWER.STANDBY_TIME 1800
*GET SETUP.POWER.STANDBY_TIME

$> "SET SETUP.POWER.STANDBY_TIME 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.POWER.STANDBY_TIME 100
```

5.109 SETUP.SYSTEM.ASSET_TAG

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Asset tag identifier

Example:

```
$> "GET SETUP.SYSTEM.ASSET_TAG" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.ASSET_TAG ""
*GET SETUP.SYSTEM.ASSET_TAG

$> "SET SETUP.SYSTEM.ASSET_TAG "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.ASSET_TAG "New Name"
```

5.110 SETUP.SYSTEM.CONTACT_INFO

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Contact information

Example:

```
$> "GET SETUP.SYSTEM.CONTACT_INFO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CONTACT_INFO ""
*GET SETUP.SYSTEM.CONTACT_INFO

$> "SET SETUP.SYSTEM.CONTACT_INFO "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.CONTACT_INFO "New Name"
```

5.111 SETUP.SYSTEM.CUSTOM1

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 8192 chars)

Custom field 1 for integrator use

Example:

```
$> "GET SETUP.SYSTEM.CUSTOM1" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOM1 ""
*GET SETUP.SYSTEM.CUSTOM1

$> "SET SETUP.SYSTEM.CUSTOM1 "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.CUSTOM1 "New Name"
```

5.112 SETUP.SYSTEM.CUSTOM2

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 8192 chars)

Custom field 2 for integrator use

Example:

```
$> "GET SETUP.SYSTEM.CUSTOM2" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOM2 ""
*GET SETUP.SYSTEM.CUSTOM2

$> "SET SETUP.SYSTEM.CUSTOM2 "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.CUSTOM2 "New Name"
```

5.113 SETUP.SYSTEM.CUSTOM3

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 8192 chars)

Custom field 3 for integrator use

Example:

```
$> "GET SETUP.SYSTEM.CUSTOM3" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOM3 ""
*GET SETUP.SYSTEM.CUSTOM3

$> "SET SETUP.SYSTEM.CUSTOM3 "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.CUSTOM3 "New Name"
```

5.114 SETUP.SYSTEM.CUSTOMER_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Customer name

Example:

```
$> "GET SETUP.SYSTEM.CUSTOMER_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.CUSTOMER_NAME "Acme Corp"
*GET SETUP.SYSTEM.CUSTOMER_NAME

$> "SET SETUP.SYSTEM.CUSTOMER_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.CUSTOMER_NAME "New Name"
```

5.115 SETUP.SYSTEM.DEVICE_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

User-defined device name

Example:

```
$> "GET SETUP.SYSTEM.DEVICE_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.DEVICE_NAME "Amp-01"
*GET SETUP.SYSTEM.DEVICE_NAME

$> "SET SETUP.SYSTEM.DEVICE_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.DEVICE_NAME "New Name"
```

5.116 SETUP.SYSTEM.INSTALLER_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Installer name

Example:

```
$> "GET SETUP.SYSTEM.INSTALLER_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.INSTALLER_NAME "Channel 1"
*GET SETUP.SYSTEM.INSTALLER_NAME

$> "SET SETUP.SYSTEM.INSTALLER_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET SETUP.SYSTEM.INSTALLER_NAME "New Name"
```

5.117 SETUP.SYSTEM.INSTALL_DATE

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Installation date

Example:

```
$> "GET SETUP.SYSTEM.INSTALL_DATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.INSTALL_DATE ""
*GET SETUP.SYSTEM.INSTALL_DATE

$> "SET SETUP.SYSTEM.INSTALL_DATE "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.INSTALL_DATE "New Name"
```

5.118 SETUP.SYSTEM.INSTALL_NOTES

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 256 chars)

Installation notes

Example:

```
$> "GET SETUP.SYSTEM.INSTALL_NOTES" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.INSTALL_NOTES ""
*GET SETUP.SYSTEM.INSTALL_NOTES

$> "SET SETUP.SYSTEM.INSTALL_NOTES "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i
 ↪  1
*SET SETUP.SYSTEM.INSTALL_NOTES "New Name"
```

5.119 SETUP.SYSTEM.LOCATING

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable locating mode (flashing LEDs)

Example:

```
$> "GET SETUP.SYSTEM.LOCATING" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.LOCATING 1
*GET SETUP.SYSTEM.LOCATING

$> "SET SETUP.SYSTEM.LOCATING 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.LOCATING 0
```

5.120 SETUP.SYSTEM.VENUE_NAME

TYPE: Register

METHODS: Get, Set

VALUES: [String] (Max Length 32 chars)

Installation venue name

Example:

```
$> "GET SETUP.SYSTEM.VENUE_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.SYSTEM.VENUE_NAME "Main Hall"
*GET SETUP.SYSTEM.VENUE_NAME

$> "SET SETUP.SYSTEM.VENUE_NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SETUP.SYSTEM.VENUE_NAME "New Name"
```

5.121 SETUP.WIFI.AP_PASS

TYPE: Register

METHODS: Get

VALUES: [String]

Access Point password

Example:

```
$> "GET SETUP.WIFI.AP_PASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.AP_PASS ""
*GET SETUP.WIFI.AP_PASS
```

5.122 SETUP.WIFI.AP_SSID

TYPE: Register

METHODS: Get

VALUES: [String]

Access Point SSID

Example:

```
$> "GET SETUP.WIFI.AP_SSID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.AP_SSID ""
*GET SETUP.WIFI.AP_SSID
```

5.123 SETUP.WIFI.DISABLE_AFTER

TYPE: Register

METHODS: Get

VALUES: [Float] (Range: 0.0 to 3600.0 Sec)

Disable WiFi after timeout (0 = never)

Example:

```
$> "GET SETUP.WIFI.DISABLE_AFTER" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.DISABLE_AFTER 1800.00
*GET SETUP.WIFI.DISABLE_AFTER
```

5.124 SETUP.WIFI.DISABLE_LAN_CONNECTED

TYPE: Register

METHODS: Get

VALUES: [Boolean]

Disable WiFi when LAN connected

Example:

```
$> "GET SETUP.WIFI.DISABLE_LAN_CONNECTED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.DISABLE_LAN_CONNECTED 1
*GET SETUP.WIFI.DISABLE_LAN_CONNECTED
```

5.125 SETUP.WIFI.ENABLE

TYPE: Register

METHODS: Get

VALUES: [Boolean]

WiFi enabled

Example:

```
$> "GET SETUP.WIFI.ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.ENABLE 1
*GET SETUP.WIFI.ENABLE
```

5.126 SETUP.WIFI.MODE

TYPE: Register

METHODS: Get

VALUES: Enum:

- **AP** - Access Point mode
- **STA** - Station (client) mode

WiFi mode (Access Point or Station)

Example:

```
$> "GET SETUP.WIFI.MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.MODE AP
*GET SETUP.WIFI.MODE
```

5.127 SETUP.WIFI.STA_PASS

TYPE: Register

METHODS: Get

VALUES: [String]

Station mode password

Example:

```
$> "GET SETUP.WIFI.STA_PASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.STA_PASS ""
*GET SETUP.WIFI.STA_PASS
```

5.128 SETUP.WIFI.STA_SSID

TYPE: Register

METHODS: Get

VALUES: [String]

Station mode SSID

Example:

```
$> "GET SETUP.WIFI.STA_SSID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SETUP.WIFI.STA_SSID ""
*GET SETUP.WIFI.STA_SSID
```

5.129 SYSTEM.DANTE.AES67_ENABLED

TYPE: Register

METHODS: Get

VALUES: [Boolean]

AES67 mode enabled

Example:

```
$> "GET SYSTEM.DANTE.AES67_ENABLED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.AES67_ENABLED 1
*GET SYSTEM.DANTE.AES67_ENABLED
```

5.130 SYSTEM.DANTE.CLOCK_STATE

TYPE: Register

METHODS: Get

VALUES: Enum {NO_CLOCK, LOCKING, LOCKED}

Dante clock synchronization state

Example:

```
$> "GET SYSTEM.DANTE.CLOCK_STATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.CLOCK_STATE NO_CLOCK
*GET SYSTEM.DANTE.CLOCK_STATE
```

5.131 SYSTEM.DANTE.DEVICE_NAME

TYPE: Register

METHODS: Get

VALUES: [String] (Max Length 32 chars)

Dante device name

Example:

```
$> "GET SYSTEM.DANTE.DEVICE_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.DEVICE_NAME "Pascal-01"
*GET SYSTEM.DANTE.DEVICE_NAME
```

5.132 SYSTEM.DANTE.ENCODING

TYPE: Register

METHODS: Get

VALUES: Enum {PCM16, PCM24, PCM32}

Dante audio encoding format

Example:

```
$> "GET SYSTEM.DANTE.ENCODING" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.ENCODING PCM16
*GET SYSTEM.DANTE.ENCODING
```

5.133 SYSTEM.DANTE.FIRMWARE_VERSION

TYPE: Register

METHODS: Get

VALUES: [String]

Dante firmware version

Example:

```
$> "GET SYSTEM.DANTE.FIRMWARE_VERSION" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.FIRMWARE_VERSION "5.3"
*GET SYSTEM.DANTE.FIRMWARE_VERSION
```

5.134 SYSTEM.DANTE.IP

TYPE: Register

METHODS: Get

VALUES: [String]

Dante network IP address

Example:

```
$> "GET SYSTEM.DANTE.IP" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.IP "192.168.1.100"
*GET SYSTEM.DANTE.IP
```

5.135 SYSTEM.DANTE.LINK_SPEED

TYPE: Register

METHODS: Get

VALUES: [Integer]

Dante network link speed

Example:

```
$> "GET SYSTEM.DANTE.LINK_SPEED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.LINK_SPEED 1
*GET SYSTEM.DANTE.LINK_SPEED
```

5.136 SYSTEM.DANTE.MAC

TYPE: Register

METHODS: Get

VALUES: [String]

Dante MAC address

Example:

```
$> "GET SYSTEM.DANTE.MAC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.MAC "00:1A:2B:3C:4D:5E"
*GET SYSTEM.DANTE.MAC
```

5.137 SYSTEM.DANTE.MUTE_STATE

TYPE: Register

METHODS: Get

VALUES: [String]

Dante mute state

Example:

```
$> "GET SYSTEM.DANTE.MUTE_STATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.MUTE_STATE ""
*GET SYSTEM.DANTE.MUTE_STATE
```

5.138 SYSTEM.DANTE.SAMPLE_RATE

TYPE: Register

METHODS: Get

VALUES: Enum {44100, 48000, 96000}

Dante sample rate

Example:

```
$> "GET SYSTEM.DANTE.SAMPLE_RATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.SAMPLE_RATE 44100
*GET SYSTEM.DANTE.SAMPLE_RATE
```

5.139 SYSTEM.DANTE.SOFTWARE_VERSION

TYPE: Register

METHODS: Get

VALUES: [String]

Dante software version

Example:

```
$> "GET SYSTEM.DANTE.SOFTWARE_VERSION" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DANTE.SOFTWARE_VERSION "5.3"
*GET SYSTEM.DANTE.SOFTWARE_VERSION
```

5.140 SYSTEM.DEVICE.FIRMWARE

TYPE: Register

METHODS: Get

VALUES: [String]

Firmware version

Example:

```
$> "GET SYSTEM.DEVICE.FIRMWARE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.FIRMWARE "1.8.0"
*GET SYSTEM.DEVICE.FIRMWARE
```

5.141 SYSTEM.DEVICE.FIRMWARE_DATE

TYPE: Register

METHODS: Get

VALUES: [String]

Firmware build date

Example:

```
$> "GET SYSTEM.DEVICE.FIRMWARE_DATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.FIRMWARE_DATE "1.8.0"
*GET SYSTEM.DEVICE.FIRMWARE_DATE
```

5.142 SYSTEM.DEVICE.HWID

TYPE: Register

METHODS: Get

VALUES: [Integer]

Hardware ID

Example:

```
$> "GET SYSTEM.DEVICE.HWID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.HWID 1
*GET SYSTEM.DEVICE.HWID
```

5.143 SYSTEM.DEVICE.MAC

TYPE: Register

METHODS: Get

VALUES: [String]

LAN MAC address

Example:

```
$> "GET SYSTEM.DEVICE.MAC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.MAC "00:1A:2B:3C:4D:5E"
*GET SYSTEM.DEVICE.MAC
```

5.144 SYSTEM.DEVICE.MODEL_NAME

TYPE: Register

METHODS: Get

VALUES: [String] (Max Length 32 chars)

Product model name

Example:

```
$> "GET SYSTEM.DEVICE.MODEL_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.MODEL_NAME "BLAZE-4"
*GET SYSTEM.DEVICE.MODEL_NAME
```

5.145 SYSTEM.DEVICE.SERIAL

TYPE: Register

METHODS: Get

VALUES: [String]

Serial number

Example:

```
$> "GET SYSTEM.DEVICE.SERIAL" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.SERIAL "SN12345678"
*GET SYSTEM.DEVICE.SERIAL
```

5.146 SYSTEM.DEVICE.SWID

TYPE: Register

METHODS: Get

VALUES: [Integer]

Software ID

Example:

```
$> "GET SYSTEM.DEVICE.SWID" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.SWID 1
*GET SYSTEM.DEVICE.SWID
```

5.147 SYSTEM.DEVICE.VENDOR_NAME

TYPE: Register

METHODS: Get

VALUES: [String] (Max Length 32 chars)

Manufacturer name

Example:

```
$> "GET SYSTEM.DEVICE.VENDOR_NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.VENDOR_NAME "Pascal"
*GET SYSTEM.DEVICE.VENDOR_NAME
```

5.148 SYSTEM.DEVICE.WIFI_MAC

TYPE: Register

METHODS: Get

VALUES: [String]

WiFi MAC address

Example:

```
$> "GET SYSTEM.DEVICE.WIFI_MAC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.DEVICE.WIFI_MAC "00:1A:2B:3C:4D:5E"
*GET SYSTEM.DEVICE.WIFI_MAC
```

5.149 SYSTEM.SECURITY.PASSWORD_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: [Boolean]

Enable password protection

Example:

```
$> "GET SYSTEM.SECURITY.PASSWORD_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.SECURITY.PASSWORD_ENABLE 1
*GET SYSTEM.SECURITY.PASSWORD_ENABLE

$> "SET SYSTEM.SECURITY.PASSWORD_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET SYSTEM.SECURITY.PASSWORD_ENABLE 0
```

5.150 SYSTEM.SECURITY.PASSWORD_HASH

TYPE: Register

METHODS: Get, Set

VALUES: [String]

Password hash (write-only for security)

Example:

```
$> "GET SYSTEM.SECURITY.PASSWORD_HASH" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.SECURITY.PASSWORD_HASH ""
*GET SYSTEM.SECURITY.PASSWORD_HASH

$> "SET SYSTEM.SECURITY.PASSWORD_HASH "New Name"" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET SYSTEM.SECURITY.PASSWORD_HASH "New Name"
```

5.151 SYSTEM.STATUS.LAN

TYPE: Register

METHODS: Get

VALUES: [String]

LAN connection status - IP address or empty if not connected

Example:

```
$> "GET SYSTEM.STATUS.LAN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.LAN ""
*GET SYSTEM.STATUS.LAN
```

5.152 SYSTEM.STATUS.SIGNAL_IN

TYPE: Register

METHODS: Get

VALUES: Enum:

- **OFF** - Input(s) is off
- **NO_SIGNAL** - Input(s) has no signal (below threshold)
- **SIGNAL** - Input(s) has signal (above threshold)
- **CLIP** - Input(s) is clipping ADC - please decrease sensitivity

Input signal status

Example:

```
$> "GET SYSTEM.STATUS.SIGNAL_IN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.SIGNAL_IN OFF
*GET SYSTEM.STATUS.SIGNAL_IN
```

5.153 SYSTEM.STATUS.SIGNAL_OUT

TYPE: Register

METHODS: Get

VALUES: Enum:

- **OFF** - Output(s) is off
- **NO_SIGNAL** - Output(s) has no signal (below threshold)
- **SIGNAL** - Output(s) has signal (above threshold)
- **CLIP** - Output(s) is clipping in amplifier - please decrease volume
- **FAULT** - Output(s) has unspecified fault

Output signal status

Example:

```
$> "GET SYSTEM.STATUS.SIGNAL_OUT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.SIGNAL_OUT OFF
*GET SYSTEM.STATUS.SIGNAL_OUT
```

5.154 SYSTEM.STATUS.STATE

TYPE: Register

METHODS: Get

VALUES: Enum:

- **INIT** - Amplifier is initializing
- **STANDBY** - Amplifier is in standby
- **STANDBY_FORCED** - Amplifier forced into standby (protection/over-temperature)
- **ON** - Amplifier is on

Current system state

Example:

```
$> "GET SYSTEM.STATUS.STATE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.STATE INIT
*GET SYSTEM.STATUS.STATE
```

5.155 SYSTEM.STATUS.WIFI

TYPE: Register

METHODS: Get

VALUES: [String]

WiFi connection status

Example:

```
$> "GET SYSTEM.STATUS.WIFI" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+SYSTEM.STATUS.WIFI ""
*GET SYSTEM.STATUS.WIFI
```

5.156 VC-{VID}.NAME

TYPE: Register

METHODS: Get

PARAMS:

- **{VID}**: See paragraph Volume Control Channels

VALUES: [String] (Max Length 32 chars)

Volume control channel name

Example:

```
$> "GET VC-1.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+VC-1.NAME "Wall VC 1"
*GET VC-1.NAME
```

5.157 VC-{VID}.VALUE

TYPE: Register

METHODS: Get

PARAMS:

- **{VID}**: See paragraph Volume Control Channels

VALUES: [Float] (Range: 0.0 to 100.0 Percent)

Volume control value (0-100%)

Example:

```
$> "GET VC-1.VALUE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+VC-1.VALUE 50.00
*GET VC-1.VALUE
```

5.158 VC.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of volume control channels

Example:

```
$> "GET VC.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+VC.COUNT 4
*GET VC.COUNT
```

5.159 ZONE-{ZID}.COMPRESSOR.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0003 to 0.05 Sec)

Compressor attack time

Example:

```
$> "GET ZONE-A.COMPRESSOR.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.ATTACK 0.03
*GET ZONE-A.COMPRESSOR.ATTACK

$> "SET ZONE-A.COMPRESSOR.ATTACK 0.03" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.ATTACK 0.03
```

5.160 ZONE-{ZID}.COMPRESSOR.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable automatic compressor makeup gain

Example:

```
$> "GET ZONE-A.COMPRESSOR.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.AUTO 1
*GET ZONE-A.COMPRESSOR.AUTO

$> "SET ZONE-A.COMPRESSOR.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.AUTO 0
```

5.161 ZONE-{ZID}.COMPRESSOR.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Bypass zone compressor

Example:

```
$> "GET ZONE-A.COMPRESSOR.BYPASS" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.BYPASS 1
*GET ZONE-A.COMPRESSOR.BYPASS

$> "SET ZONE-A.COMPRESSOR.BYPASS 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.BYPASS 0
```

5.162 ZONE-{ZID}.COMPRESSOR.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0 to 1.0 Sec)

Compressor hold time

Example:

```
$> "GET ZONE-A.COMPRESSOR.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.HOLD 0.50
*GET ZONE-A.COMPRESSOR.HOLD

$> "SET ZONE-A.COMPRESSOR.HOLD 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.HOLD 0.50
```

5.163 ZONE-{ZID}.COMPRESSOR.KNEE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0 to 12.0 dB)

Compressor knee width

Example:

```
$> "GET ZONE-A.COMPRESSOR.KNEE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.KNEE 6.00
*GET ZONE-A.COMPRESSOR.KNEE

$> "SET ZONE-A.COMPRESSOR.KNEE 6.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.KNEE 6.00
```

5.164 ZONE-{ZID}.COMPRESSOR.RATIO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 1.0 to 50.0)

Compressor ratio

Example:

```
$> "GET ZONE-A.COMPRESSOR.RATIO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.RATIO 25.50
*GET ZONE-A.COMPRESSOR.RATIO

$> "SET ZONE-A.COMPRESSOR.RATIO 4.0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.RATIO 4.0
```

5.165 ZONE-{ZID}.COMPRESSOR.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.001 to 1.0 Sec)

Compressor release time

Example:

```
$> "GET ZONE-A.COMPRESSOR.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.RELEASE 0.50
*GET ZONE-A.COMPRESSOR.RELEASE

$> "SET ZONE-A.COMPRESSOR.RELEASE 0.50" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.RELEASE 0.50
```

5.166 ZONE-{ZID}.COMPRESSOR.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -40.0 to 20.0 dB)

Compressor threshold

Example:

```
$> "GET ZONE-A.COMPRESSOR.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.COMPRESSOR.THRESHOLD -20.00
*GET ZONE-A.COMPRESSOR.THRESHOLD

$> "SET ZONE-A.COMPRESSOR.THRESHOLD -30.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.COMPRESSOR.THRESHOLD -30.00
```

5.167 ZONE-{ZID}.DUCK.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.001 to 0.2 Sec)

Ducker attack time

Example:

```
$> "GET ZONE-A.DUCK.ATTACK" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.ATTACK 0.10
*GET ZONE-A.DUCK.ATTACK

$> "SET ZONE-A.DUCK.ATTACK 0.10" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.ATTACK 0.10
```

5.168 ZONE-{ZID}.DUCK.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable automatic ducking

Example:

```
$> "GET ZONE-A.DUCK.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.AUTO 1
*GET ZONE-A.DUCK.AUTO

$> "SET ZONE-A.DUCK.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.AUTO 0
```

5.169 ZONE-{ZID}.DUCK.DEPTH

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -144.0 to 0.0 dB)

Ducking depth (attenuation amount)

Example:

```
$> "GET ZONE-A.DUCK.DEPTH" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.DEPTH -72.00
*GET ZONE-A.DUCK.DEPTH

$> "SET ZONE-A.DUCK.DEPTH -72.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.DEPTH -72.00
```

5.170 ZONE-{ZID}.DUCK.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.0 to 10.0 Sec)

Ducker hold time

Example:

```
$> "GET ZONE-A.DUCK.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.HOLD 5.00
*GET ZONE-A.DUCK.HOLD

$> "SET ZONE-A.DUCK.HOLD 5.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.HOLD 5.00
```

5.171 ZONE-{ZID}.DUCK.MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: Enum:

- **OFF** - Ducking disabled
- **DUCKER** - Standard ducking mode
- **OVERRIDE** - Override mode - priority source replaces primary

Ducker mode

Example:

```
$> "GET ZONE-A.DUCK.MODE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.MODE OFF
*GET ZONE-A.DUCK.MODE

$> "SET ZONE-A.DUCK.MODE DUCKER" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.MODE DUCKER
```

5.172 ZONE-{ZID}.DUCK.OVERRIDE_GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -144.0 to 15.0 dB)

Override gain for primary source during ducking

Example:

```
$> "GET ZONE-A.DUCK.OVERRIDE_GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.OVERRIDE_GAIN -6.00
*GET ZONE-A.DUCK.OVERRIDE_GAIN

$> "SET ZONE-A.DUCK.OVERRIDE_GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.OVERRIDE_GAIN -12.00
```

5.173 ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable override gain during ducking

Example:

```
$> "GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 1
*GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE

$> "SET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 0
```

5.174 ZONE-{ZID}.DUCK.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: 0.01 to 10.0 Sec)

Ducker release time

Example:

```
$> "GET ZONE-A.DUCK.RELEASE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.RELEASE 5.00
*GET ZONE-A.DUCK.RELEASE

$> "SET ZONE-A.DUCK.RELEASE 5.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.RELEASE 5.00
```

5.175 ZONE-{ZID}.DUCK.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

Ducker trigger threshold

Example:

```
$> "GET ZONE-A.DUCK.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.DUCK.THRESHOLD -20.00
*GET ZONE-A.DUCK.THRESHOLD

$> "SET ZONE-A.DUCK.THRESHOLD -30.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.DUCK.THRESHOLD -30.00
```

5.176 ZONE-{ZID}.DYN.SIGNAL

TYPE: Subscription Only

METHODS: Subscribe

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -144.0 to 20.0 dB)

NOTES: Subscription only — streamed on SUBSCRIBE DYN; not GET-able

Zone signal level (dynamic)

Example:

```
$> "SUBSCRIBE DYN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SUBSCRIBE DYN
+ZONE-A.DYN.SIGNAL -62.00
```

5.177 ZONE-{ZID}.GAIN

TYPE: Register

METHODS: Get, Set, Inc

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

NOTES: Read-Only if ZONE-{ZID}.GPIO_VC is set on zone. Actual range is [ZONE-{ZID}.GAIN_MIN, ZONE-{ZID}.GAIN_MAX]

Zone gain level

Example:

```
$> "GET ZONE-A.GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN -6.00
*GET ZONE-A.GAIN

$> "SET ZONE-A.GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GAIN -12.00
```

5.178 ZONE-{ZID}.GAIN_MAX

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

NOTES: Lower bound is the current ZONE-{ZID}.GAIN_MIN; ceiling is 0.0

Maximum gain limit

Example:

```
$> "GET ZONE-A.GAIN_MAX" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN_MAX -6.00
*GET ZONE-A.GAIN_MAX

$> "SET ZONE-A.GAIN_MAX -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GAIN_MAX -12.00
```

5.179 ZONE-{ZID}.GAIN_MIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Float] (Range: -80.0 to 0.0 dB)

NOTES: Upper bound is the current ZONE-{ZID}.GAIN_MAX

Minimum gain limit

Example:

```
$> "GET ZONE-A.GAIN_MIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GAIN_MIN -6.00
*GET ZONE-A.GAIN_MIN

$> "SET ZONE-A.GAIN_MIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GAIN_MIN -12.00
```

5.180 ZONE-{ZID}.GPIO_VC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Integer]

GPIO volume control assignment (VID, 0 for OFF)

Example:

```
$> "GET ZONE-A.GPIO_VC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.GPIO_VC 1
*GET ZONE-A.GPIO_VC

$> "SET ZONE-A.GPIO_VC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.GPIO_VC 100
```

5.181 ZONE-{ZID}.MUTE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Zone mute state

Example:

```
$> "GET ZONE-A.MUTE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.MUTE 1
*GET ZONE-A.MUTE

$> "SET ZONE-A.MUTE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.MUTE 0
```

5.182 ZONE-{ZID}.MUTE_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Enable mute functionality for zone

Example:

```
$> "GET ZONE-A.MUTE_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.MUTE_ENABLE 1
*GET ZONE-A.MUTE_ENABLE

$> "SET ZONE-A.MUTE_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.MUTE_ENABLE 0
```

5.183 ZONE-{ZID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [String] (Max Length 32 chars)

Zone name

Example:

```
$> "GET ZONE-A.NAME" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.NAME "Lobby"
*GET ZONE-A.NAME

$> "SET ZONE-A.NAME "New Name"" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.NAME "New Name"
```

5.184 ZONE-{ZID}.PRIMARY_SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Integer]

NOTES: See {SID} Input Source definitions

Primary source input (SID value)

Example:

```
$> "GET ZONE-A.PRIMARY_SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIMARY_SRC 1
*GET ZONE-A.PRIMARY_SRC

$> "SET ZONE-A.PRIMARY_SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIMARY_SRC 100
```

5.185 ZONE-{ZID}.PRIORITY-{ZP}.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Boolean]

Enable automatic priority override

Example:

```
$> "GET ZONE-A.PRIORITY-2.AUTO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.AUTO 1
*GET ZONE-A.PRIORITY-2.AUTO

$> "SET ZONE-A.PRIORITY-2.AUTO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.AUTO 0
```

5.186 ZONE-{ZID}.PRIORITY-{ZP}.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Float] (Range: 0.001 to 10.0 Sec)

Priority hold time

Example:

```
$> "GET ZONE-A.PRIORITY-2.HOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.HOLD 5.00
*GET ZONE-A.PRIORITY-2.HOLD

$> "SET ZONE-A.PRIORITY-2.HOLD 5.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.HOLD 5.00
```

5.187 ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Float] (Range: -144.0 to 15.0 dB)

Override gain level for primary source

Example:

```
$> "GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.OVERRIDE_GAIN -6.00
*GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN
```

```
$> "SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN -12.00" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN -12.00
```

5.188 ZONE-{ZID}.PRIORITY-{ZP}.OVERRIDE_GAIN_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Boolean]

Enable override gain adjustment

Example:

```
$> "GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
+ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE 1
*GET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE

$> "SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE 0" | ncat 192.168.64.100 7621 --no-shutdown
 ↪  -i 1
*SET ZONE-A.PRIORITY-2.OVERRIDE_GAIN_ENABLE 0
```

5.189 ZONE-{ZID}.PRIORITY-{ZP}.SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Integer]

NOTES: See {SID} Input Source definitions

Priority source input (SID value)

Example:

```
$> "GET ZONE-A.PRIORITY-2.SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.SRC 1
*GET ZONE-A.PRIORITY-2.SRC

$> "SET ZONE-A.PRIORITY-2.SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.SRC 100
```

5.190 ZONE-{ZID}.PRIORITY-{ZP}.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{ZP}**: See paragraph Zone Priorities

VALUES: [Float] (Range: -80.0 to 0.0 dB)

Priority trigger threshold

Example:

```
$> "GET ZONE-A.PRIORITY-2.THRESHOLD" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY-2.THRESHOLD -20.00
*GET ZONE-A.PRIORITY-2.THRESHOLD

$> "SET ZONE-A.PRIORITY-2.THRESHOLD -30.00" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY-2.THRESHOLD -30.00
```

5.191 ZONE-{ZID}.PRIORITY_SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Integer]

NOTES: Valid SID value

Priority source input (legacy, use PRIORITY-{ZP}.SRC for FW 1.8+)

Example:

```
$> "GET ZONE-A.PRIORITY_SRC" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.PRIORITY_SRC 1
*GET ZONE-A.PRIORITY_SRC

$> "SET ZONE-A.PRIORITY_SRC 100" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.PRIORITY_SRC 100
```

5.192 ZONE-{ZID}.SRC-{IID}.ENABLED

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones
- **{IID}**: See paragraph Input Channels

VALUES: [Boolean]

Enable input source for zone mixing

Example:

```
$> "GET ZONE-A.SRC-100.ENABLED" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.SRC-100.ENABLED 1
*GET ZONE-A.SRC-100.ENABLED

$> "SET ZONE-A.SRC-100.ENABLED 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.SRC-100.ENABLED 0
```

5.193 ZONE-{ZID}.STEREO

TYPE: Register

METHODS: Get, Set

PARAMS:

- **{ZID}**: See paragraph Zones

VALUES: [Boolean]

Stereo link zone with next zone

Example:

```
$> "GET ZONE-A.STEREO" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE-A.STEREO 1
*GET ZONE-A.STEREO

$> "SET ZONE-A.STEREO 0" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*SET ZONE-A.STEREO 0
```

5.194 ZONE.COUNT

TYPE: Register

METHODS: Get

VALUES: [Integer]

Number of zones

Example:

```
$> "GET ZONE.COUNT" | ncat 192.168.64.100 7621 --no-shutdown -i 1
+ZONE.COUNT 4
*GET ZONE.COUNT
```