TELNET PROTOCOL DOCUMENTATION FOR

The BarnMini-05

Revision 1.0

Barnfind Technologies AS
https://www.barnfind.no

Contents

1. Introduction

This document provides technical details on the Telnet protocol implementation for **BarnMini-05**. It describes the command structure, network configurations and a brief summery for all other sections that can be present.

1.1 Purpose

The purpose of this document is to assist developers, system integrators, and IT administrators in configuring and using the Telnet interface for device management and control.

2. Command structure

Every command is terminated with a new line (0x0a). From the client there are a total of 2 types of commands: select section and to write to values/properties.

To select a target section (object). The name of the ***section*** is put inside square brakes.

Example:

```
[port.1]
```

- When a valid section has been selected, values can be sent.
- Numeric values are given as is, string values are put inside quotations.
- In newer firmware versions, a value can be deleted if it set to an empty string without any quotations.

Example:

```
someint=1
somestring="foo"
someoldvar=
```

Strings can be escaped using the follow codes:

\n - for a newline (0x0a)

\r - for a return carrier (0x0d)

\x00 - for a hexadecimal value where 00 is the designated value

\\ - for a backslash

\t - for a tab (0x09 is also valid, server will always return \x09 notation)

We recommend that all values outside the ASCII range 32-126, \ ' and " to be escaped.

From the server, there are two additional commands. ACK and NAK. These are sent as reply to all commands from the client to verify that the command is valid or not.

The server and client streams are not synchronized, with exception of ACK/NAK being appended to the stream from the server to the client as it receives commands. When the client receives a matching ACK/NAK, the command has been processed, and parameters that are directly influenced by the command should already have been transmitted at this point.

3. Sections

3.1 Section “[barnmini05]”

This section contains identification, model and version information:

[barnmini05]	
API=1	Version number of the API. 1 = the API described in this document.
firmwareversion="1.2.21"	Running firmware version.
model="BarnMini-05"	Identifies the device model.
serialnumber="BM551JBP123456"	Device serial number.
sysName="Test Frame"	Device name / identifier (writable).
sysContact="Operator Foo"	Contact person responsible for this device (writable).
sysLocation="The office"	Physical location of the system (writable).
GPIOLED="GPO"	This is writable property, determining how the LEDs at the GPI/O connector are controlled. Operation modes: • "GPI" = LEDs are controlled by a GPI (on when detected low/shorted). • "!GPI" = LEDs are controlled by GPI, but with inverted logic (on when detected high/floating). • "GPO" = LEDs are controlled by a GPO (on while closed). • "!GPO" = LEDs are controlled by GPO (on while open).

3.2 The Network sections

3.2.1 Section "[network]"

The first section within the network namespace is name "[network]" and currently only contains two read-only properties: **version** and **id**.

- The software version of the network handling software (named “emnema” internally).
- The unique identifier of the frame (the Ethernet MAC address is use for this) .

Examples:

[network]	
version="100.1.0"	Specifies the software version of the network handling software being used.
id="e8:eb:1b:37:f7:94"	The MAC address of the network interface/Unique ID of the device. (in addition to Serial Number)

3.2.2 Section "[network.config]"

The “[network.config]” section contains the current configuration file used by the network configuration software and all properties in this section are writeable.

If the client wants to modify this configuration, it should first update all the desired properties (or all properties if needed) before finalizing the changes by setting “commit=1”. This is to avoid the frame reconfiguring its networks settings while they are being uploaded.

Example:

[network.config]	
ipv4.static.addr[0]="192.168.0.95"	Up to 4 static IPv4 addresses can be configured.
ipv4.static.prefix[0]=24	This contains the prefix (also known as the netmask in IPv4: 24=255.255.255.0) for the given static IPv4 address.
...	
ipv4.static.addr[3]="0.0.0.0"	
ipv4.static.prefix[3]=0	

ipv4.static.gw="192.168.0.1"	The gateway if IPv4 is configured to the static.
ipv4.mode=1	Operation mode for IPv4. 0=Disabled, 1=DHCP, 2=Static, 8=LinkLocal.
ipv6.static.addr[0]="2001::2"	Up to 4 static IPv6 addresses can be configured.
ipv6.static.prefix[0]=64	This contains the prefix for the given static IPv6 address.
…	
ipv6.static.addr[3]="::"	
ipv6.static.prefix[3]=0	
ipv6.static.gw="::"	The gateway if IPv6 is configured to the static.
ipv6.mode=12	Operation mode for IPv6. 8=LinkLocal only, 9=DHCP/StateFull, 10=Static,12=StateLess/Router Advertisement.
commit=0	Set this value to 1, in order to commit any changes.

3.2.3 Section "[network.status.ip.ID]"

This section provides the current status regarding IP address for the network interfaces. The current active IP addresses assigned to the frame are readable by the "[network.status.ip.ID]" sections. Each section will contain 3 properties: **dev**, **addr**, and **prefix** (The subnet mask length). Non-used entries will have prefix=0.

Example:

[network.status.ip.0]	
dev="st"	Assigned to the network interface st.
addr="192.168.0.116"	An IPv4 address assigned to this interface. This is the IPv4 address assigned to st (192.168.0.x).
prefix=16	Subnet mask of 255.255.0.0 (/16 means that the upper 16 bits in the subnet mask is set to 1).
[network.status.ip.1]	
dev="st"	Assigned to the network interface st.

addr="FE80::821F:12FF:FE40:8741"	A Link-Local IPv6 address. fe80::/10 addresses are link-local and used only within the local network. These addresses cannot be routed outside the local network.
prefix=64	Current prefix for the IPv6 address.
…	
[network.status.ip.3]	
dev=""	No network interface is assigned.
addr=""	No IP address is assigned.
prefix=0	No subnet mask is assigned.

3.2.4 Section "[network.status.route.ID]"

This section defines the routing table, specifying how network traffic should be directed.

The current routing table assigned to the frame are readable by the "[network.status.route.ID]" sections. Each section will contain four properties: **dev** (Network interface (eth0, etc.).), **dst** (for the destination network or IP address), **dstprefix** (**for** the subnet prefix length for the destination.) and **gw** (the IP address of the gateway / next-hop router for forwarding packets). Non-used entries will have dst="".

Example:

[network.status.route.0]	
dev="st"	The network interface assigned is st.
dst="192.168.0.0"	This route covers the 192.168.0.0/16 network.
dstprefix=16	The subnet mask is /16 (255.255.0.0), meaning it applies to all IPs in 0.0.0.0/16.
gw=""	No gateway assigned.
[network.status.route.1]	
dev="st"	The network interface st is used.
dst="0.0.0.0"	This is the default route (matches all destinations). Any traffic not matching other routes will be sent to gw, 192.168.0.1.
dstprefix=0	No specific subnet mask (applies to all traffic).

gw="192.168.0.1"	The gateway (router) IP is 192.168.0.1.
[network.status.route.2]	
dev="st"	
dst="FE80::"	This is the IPv6 Link-Local network.
dstprefix=64	Covers only fe80::/64 (Link-Local addresses). fe80::/64 is reserved for IPv6 link-local communication. It allows devices on the same physical network to communicate without an external router.
gw=""	No gateway (link-local traffic only).
[network.status.route.3]	
dev=""	No network interface assigned.
dst=""	No destination address assigned.
dstprefix=0	No subnet mask.
gw=""	No gateway assigned.
…	

3. 3 The serial UART section

3.3.1. Section "[uart.ID]"

This “[uart.ID]” section contains the configuration settings for the UART (Universal Asynchronous Receiver-Transmitter) port. This “Serial UART” port can be configured to operate in RS-422, RS-485 and DMX modes and can be used for remote accessing serial devices over TCP/IP. The BarnMini-05 can operate both as a server and a client on TCP/IP.

Example:

[uart.0]	
baudrate=115200	This is writable property, configures the transmission speed in bits per second (bps).
stopbits=1	This is writable property, configures the number of stop bits per byte. Options: 1 or 2.
parity="none"	This is writable property, configures the parity mode.

	Possible options: • "none" (No parity check) • "even" (Even parity check) • "odd" (Odd parity check)
rs=422	This is writable property, selects the operation mode to use. Possible options: • 422 = RS-422, duplex operation mode. • 485 = RS-485, half-duplex operation mode. • 486 = RS-485 with no echo. The data transmitted will be ignored as they appear on the RX, hence “no-echo”. • 512 = DMX-512 transmitter, a protocol used in stage lighting and entertainment applications. Can be controlled using Art-Not on UDP port 0x1936 (6454 decimal).
clienthost="192.168.0.95"	This is writable property. If using the UART TCP/IP client, this is the target host IP address to connect to. Both IPv4 and IPv6 is supported. Set to “0.0.0.0” to disable the client.
clientport=2167	This is writable property, specifies the port number for the UART client to connect to.
client="connected"	Indicates the current connection status of the UART client.
serverport=2167	Defines the port number on which the UART server is listening to. The system accepts incoming UART connections on this port.
remotehost.0="fe80:0000:0000:0000:5537:1bcf:4b3c:c735"	The UART TCP/IP server can have up to 2 incoming connections at once. The current connected client will be visible here. If slot is unused, “” will be returned.
remoteport.0=61738	Specifies the port number of the current connected client.
remotehost.1=""	
remoteport.1=0	
…	

3.4 The Tally/GPIO sections

3.4.1 General Information for TSL Tally/GPIO Section

The BarnMini-05 have 4 GPI and 4 GPO ports and supports multiple protocols for transferring tallies in and out of the device.

- **GPI (Inputs):**
  The GPI (inputs) can be forwarded to multiple destinations. They receive signals and can forward those signals to multiple destinations, allowing signals to be used by different systems simultaneously, including:
  - **UMD v4.0 Client**: Connect to a remote UMD v4.0 server and set the UMD tallies/colors.
  - **UMD v5.0 Client**: Connect to a remote UMD v5.0 server and set the UMD colors.
  - **TSL Tally Server**: When receiving connections from a remote TSL Tally Client, GPI can be sent in return.
  - **TSL Tally Client**: When connected to a remote TSL Tally Server, GPI can be sent.
  - **To Telnet**: A remote command interface (CLI) that makes it possible to receive the GPI signals over a network. This is what BarnStudio utilizes.

- **GPO (Outputs) Control:**
  The GPO (outputs) are typically used for controlling remote devices and tally lights. The GPOs can be controlled by multiple sources at the same time, including:
  - **UMD v4.0 Server**: Receive tallies from a remote UMD 4.0 client.
  - **TSL Tally Server**: Connect to a remote TSL Tally Server and receive tallies.
  - **TSL Tally Client**: When receiving connections from a remote TSL Tally Client, incoming tallies can be forwarded to the GPOs.
  - **Ping Client**: A built-in system that checks network availability using ping and can trigger GPOs based on connectivity status (e.g., if a device is offline, it can activate an alert).
  - **From Telnet**: A remote command interface (CLI) that makes it possible to receive GPO signals over a network. This is what BarnStudio utilizes.

While it is possible to configure multiple sources to control a GPO at the same time, it is not recommended, as conflicts may arise where multiple systems try to set different states for the same output.

There are LEDs available in the connector. These can be configured to follow either the GPI or the GPO. See section “[barnmini-05]”.

3.4.2 Section "[GPI.ID]"

This section covers the General-Purpose Inputs (GPI) and contain its state. Each section is named "[GPI.ID]", where ID starts from 0 and will contain only one property: **state**.

BarnStudio adds 1 to the ID number when presented to the users (e.g., GPI.0 is presented as “GPI.1”).

Example:

[GPI.0]	
state="open"	The current detected state of the input. “open” = Circuit is detected as open (high voltage). “close” = Circuit is detected as close (low voltage).
…	

3.4.3 Section "[GPO.ID]"

This section covers the General-Purpose Outputs (GPO) and contains its state and configuration for:

- UMD 4.0 server (mapping of Inbound signals to GPO)
- TSL Tally Server (mapping of Inbound signals to GPO).

Each section is named "[GPO.ID]", where ID starts from 0. Each section will contain four properties: **UMDv40.display**, **UMDv40.tally**, **TSL.tally** and **state**.
BarnStudio adds 1 to the ID number when presented to the users (e.g., GPO.0 is presented as “GPO.1”).

Example:

[GPO.0]

UMDv40.display=1	This is writable property, configures the display ID from UMD v4.0 Server that is associated with this GPO.ID. Example: If the UMD server is used, this configures which display should be linked to this GPO.ID. Refer to chapter 3.6.1 for details.
UMDv40.tally=2	This is writable property, configures the tally ID from the UMD v4.0 Server that is linked to this GPO.ID. Refer to chapter 3.6.1 for details.
TSL.tally=1	This is writable property, configures the tally ID from the TSL Tally Server that is mapped to this GPO.ID. Refer to chapter 3.9.1 for details.
state="open"	The current state of the output and is writable. “open” = The pins on the output are floating/open. “closed” = The pins on the output are shorted together (using a relay).
…	

3.5 The UMD v4.0 server section

3.5.1 GPO Visibility/Patch Configuration

This section defines the configuration of four GPO ports (GPO.0 to GPO.3). Each GPO port can be assigned a display ID (ranging from 0 to 126 or set to "Disabled" by setting it to -1) and a Tally ID (ranging from Tally 1 to Tally 4).

This configuration allows mapping of specific displays and tally signals to GPO ports, ensuring proper integration with the UMD v4.0 Server.

Examples:

[GPO.0]	
UMDv40.display=1	This is writable property, configures the display ID from UMD v4.0 Server that is associated with this GPO.ID.

	Example: If a UMD system is used, this configures which display is linked to this specific GPO.ID. Possible values are: • 0 to 126: This GPO will be linked to UMD v4.0 server Display **ID**, Tally ID. • -1: This GPO will not be linked to UMD v4.0 Server.
UMDv40.tally=2	This is writable property, configures which tally signal is linked to this specific GPO.ID. Configures the tally ID from the UMD v4.0 Server that is linked to this GPO.ID. Possible values are: • 1 to 4: This GPO will be linked to UMD v4.0 server Display ID, Tally **ID**
…	
[GPO.3]	
UMDv40.display=3	
UMDv40.tally=4	

3.5.2 Section "[UMD_server.tcp]"

This “[UMD_server.tcp]” section contains the current incoming TCP connections to the UMD server. It also provides the server port number.

This section currently contains three properties: **port**, and for each client slot a **remotehost** and **remoteport**.

Example:

[UMD_server.tcp]	
port=8910	This defines the TCP port number that the UMD server listens on for incoming connections.
remotehost.0=""	This provides the IP address of the first remote client, counting from slot 0. If slot is available/unused it will be an empty string (“”).
remoteport.0=0	This provides the port number of the first remote client.

…	
remotehost.3=""	Fourth remote client slot.
remoteport.3=0	Fourth remote client slot.

3.5.3 Section "[UMD_server.udp]"

This “[UMD_server.udp]” section contains the current incoming UDP connections to the UMD server. It also provides the server port number.

This section currently contains three properties: **port**, and for each client **remotehost** and **remoteport**.

Example:

[UMD_server.udp]	
port=8900	This defines the UDP port number that the UMD server listens on for incoming connections.
remotehost.0="192.168.1.50"	This provides the IP address of the first remote client, counting from slot 0. If slot is available/unused it will be an empty string (“”).
remoteport.0=50	This provides the port number of the first remote client.
…	
remotehost.3=""	Fourth remote client slot.
remoteport.3=0	Fourth remote client slot.

3.6 The UMD v4.0 client section

3.6.1 Section "[UMD_client.ID]"

This “[UMD_client.ID]” section contains the configuration of a UMD v4.0 client. This client connects to a UMD v4.0 server to transmit tally and text.

Each UMD client is assigned a unique ID (starting from 0) and has fourteen properties that define how it to connect and interacts with the server: **host**, **port**, **transport**, **v40_xdata**, **display**, **brightness**, **tally_out**, **xdata_L_L**, **xdata_L_T**, **xdata_L_R**, **xdata_R_L**, **xdata_R_T**, **xdata_R_R** and **status**.

xdata properties:

- The **xdata** properties in UMD v4.0 client is used to configure the optional color data for six display positions, sent to the on a UMD v4.0 server. These six positions are:
  - Left display: left tally, text and right tally
  - Right display: left tally, text and right tally
- Each of these positions can display one of four colors:
  - 0=off
  - 1= Red
  - 2=Green
  - 3=Amber
- There are 16 numbers separated by a comma.

How Colors are mapped using GPI inputs

The **xdata** values are determined using the state of four GPI inputs (GPI.3 high/low, GPI.2 high/low, GPI.1 high/low and GPI.0 high/low). These four GPI signals form a binary number, which is then used as an index to select one of 16 possible color values in the **xdata** list.

GPI.3	GPI.2	GPI.1	GPI.0	Decimal Index
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
…	…	…	…	…
1	1	1	1	15

Example: Consider the xdata_L_L property for "Left Display - Left Tally" with the following values:

xdata_L_L="0,1,2,3,0,1,2,3,1,2,3,0,1,2,3,0"

- If all GPI inputs are LOW (GPI.3 = 0, GPI.2 = 0, GPI.1 = 0, GPI.0 = 0), the first value is used, 0 - Off.

- If only GPI.0 is HIGH (GPI.3 = 0, GPI.2 = 0, GPI.1 = 0, GPI.0 = 1), the second value is used, 1 - Red.
- If GPI.0 and GPI.3 are HIGH (GPI.3 = 1, GPI.2 = 0, GPI.1 = 0, GPI.0 = 1), the tenth value is used, 2 - Green.

This system allows dynamic color assignment based on GPI states, making tally indications adaptable based on input conditions.

Example:

[UMD_client.0]	
host="192.168.1.50"	A writable property, that configures the target address that the UMD v4.0 client will connect to.
port=8900	A writable property, that configures the target port number that the UMD v4.0 client will connect to. (TCP normally should use port 8910 and UDP use 8900)
transport="tcp"	A writable property, that configures the connection type that the UMD v4.0 client will use. Possible values are: "udp" or "tcp".
v40_xdata=1	A writable property, that configures if xdata (e.g., extended text formatting) should be transmitted alongside the tallies or not. Possible values are: 0 = Disabled, 1 = Enabled.
display=0	A writable property, configuring the target display that should be controlled by this client. Display 127 is broadcast (all displays). Some implementations of UMD servers start counting from 1 instead of 0.
brightness=2	This controls the brightness level of the tallies/text. Possible values are: • 0 = Off

	• 1 = Low brightness (one seventh) • 2 = Medium brightness (half) • 3 = High brightness (full)
tally_out.0=1	This defines tally output mappings from GPI. Possible values are: • 0 through 3 for GPI.0 to GPI.3 • 4 through 7 for inverse value of GPI.0 to GPI.3 • 8 for always clear • 9 for always set Example: tally_out.0=1 links "Tally Output 1 to GPI.1".
tally_out.1=5	
tally_out.2=9	
tally_out.3=8	
xdata_L_L= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	This property is for configuring position: Left display, left tally
xdata_L_T= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	Left display, text
xdata_L_R= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	Left display, right tally
xdata_R_L= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	Right display, left tally
xdata_R_T= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	Right display, text
xdata_R_R= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	Right display, right tally
status="disconnected"	Displays the current connection status of the UMD client.

3.7 The UMD v5.0 client section

3.7.1 Section "[UMD_v50_client.ID]"

Each "[UMD_v50_client.ID]" section contains the configuration of a UMD v5.0 client. This client connects to a UMD v5.0 server to transmit tally and text.

Each UMD v5.0 client has a unique ID (starting from 0) and has ten properties **host**, **port**, **transport**, **screen**, **index**, **brightness**, **RH**, **TXT**, **LH** and **status**.

Example:

[UMD_v50_client.0]	
host="192.168.0.95"	A writable property, that configures the target address that the UMD v5.0 client will connect to.
port=8900	A writable property, that configures the target port number that the UMD v5.0 client will connect to. (TCP normally should use port 8910 and UDP use 8900)
transport="udp"	A writable property, that configures the connection type that the UMD v5.0 will connect to. Possible values are: "udp" or "tcp".
screen=1	A writable property, that defines which screen to control on the remote UMD system. Screen 65535 is broadcast (all screens). Some implementations of UMD servers start counting from 1 instead of 0.
index=0	A writable property, that defines the index number within the target screen on the UMD v5.0 system. Index 65535 is broadcast (all indexes). Some implementations of UMD servers start counting from 1 instead of 0.
brightness=0	This controls the brightness level of the tallies/text. Possible values are: • 0 = Off • 1 = Low brightness (one seventh) • 2 = Medium brightness (half) • 3 = High brightness (full)

RH= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	This property is for configuring position: For the Right-hand(RH) tally indicators. Refer chapter 3.7.1 (**xdata** property)
TXT= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	For the Text displayed on the screen. Refer chapter 3.7.1 (**xdata** property)
LH= "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"	For the Left-Hand (LH) Tally indicators. Refer chapter 3.7.1 (**xdata** property)
status="disconnected"	Displays the current connection status of the UMD client.

3.8 The TSL Tally server section

This section contains the configuration for the TSL Tally Server, which exchange tallies with connected clients.

3.8.1 Section “[GPO.ID]”

(GPO Visibility/Patch)

This “[GPO.ID]” section defines the configuration of four GPO ports (GPO.0 to GPO.3). Each GPO port can be assigned to an incoming tally (ranging from 1 to 8 or set to "Disabled" by setting it to -1).

This configuration allows mapping of specific tally signals to GPO ports, ensuring proper integration with the TSL Tally server.

Example:

[GPO.0]	
TSL.tally=1	Writable property for mapping the first tally output from the TSL Tally Server to a specific GPO. Example: First tally output is mapped to GPI.0
…	

[GPO.3]	
TSL.tally=-1	-1: This GPO will not be linked to the TSL Tally Server

3.8.2 Section "[tally_server]"

(GPI Visibility/Patch)

This “[tally_server]” section is used to map the input sources to the target inputs for the Tally Server. It specifies which inputs will be linked to specific tally signals and how the tally server will manage and interpret these inputs.

Each **input.ID** defines a connection to a particular input source for the tally server. The inputs can be connected to GPI.ID (with number 0 through 3) or be fixed to cleared (with number -1). This section currently contains eight tally inputs that can be transmitted.

Example:

[tally_server]	
input.0=0	Writable property for mapping the first tally input that can be transmitted with TSL Tally Server. Possible values are: • 0 to 3, represents GPI.0 to GPI.3. • -1 for not connecting this to a GPI and always send clear. First tally input is mapped to GPI.0
input.1=2	Second tally input is mapped is to GPI.2
input.2=1	
input.3=3	
input.4=-1	The value -1 means this input is always cleared.
input.5=-1	
input.6=-1	
input.7=-1	

3.8.3 Section "[tally_server.tcp]"

This “[tally_server.tcp]” section contains the current incoming TCP connections to the tally server. It also provides the server port number.

This section currently contains three properties: **port**, and for each client slot contain **remotehost** and **remoteport**.

Example:

[tally_server.tcp]	
port=8001	This defines the TCP port number that the tally server listens on for incoming connections.
remotehost.0="192.168.1.50"	This provides the IP address of the first remote client, counting from slot 0. If slot is available/unused it will be an empty string (“”).
remoteport.0=500	This provides the port number of the first remote client.
…	
remotehost.3=""	Fourth remote client slot.
remoteport.3=0	Fourth remote client slot.

3.8.4 Section "[tally_server.udp]"

This “[tally_server.udp]” section contains the current incoming UDP connections to the tally server. It also provides the server port number.

This section currently contains three properties: **port**, and for each client **remotehost** and **remoteport**.

Example:

[tally_server.udp]	
port=9001	This defines the UDP port number that the UMD server listens on for incoming connections.

remotehost.0="192.168.1.10"	This provides the IP address of the first remote client, counting from slot 0. If slot is available/unused it will be an empty string (“”).
remoteport.0=400	This provides the port number of the first remote client.
...	
remotehost.3=""	Fourth remote client slot.
remoteport.3=0	Fourth remote client slot.

3.9The TSL Tally client section

This section contains the configuration for the TSL Tally Clients, which connects to a TSL Tally Server to exchange tallies.

3.9.1 Section "[tally_client.ID]"

Each section here contains one available TSL Tally client. Each section is named "[tallyClient.ID]", where ID starts from 0. These sections contain properties: **host**, **port**, **transport**, **tally_out**, **tally_in** and **status**.

In **tally_out.ID:** These properties define the outbound tally signal mapping to the Tally Client from specific GPI ports or tally functions. Each tally_out.ID property assigns a tally output signal to a General-Purpose Input (GPI) or a tally control function.

- GPI: The tally output is linked to a GPI port (GPI.0 to GPI.3).
- Inverse GPI: The tally output is linked to the inverse value of a GPI port.
- Other Functions:
    - "clear": send always 0 / low / clear.
    - "set": send always 1 / high / set.

In **tally_in.ID** : This section defines the inbound tally signal mapping from the remote TSL Tally Server and to local GPOs. Each tally_in.ID represents a GPO.ID, and its value tells which GPO it should be connected to (or leave as disconnected)

Example:

```
[tally_client.0]
```

host="0.0.0.0"	A writable property, that configures the target address that the TSL Tally client will connect to.
port=9001	A writable property, that configures the target port number that the TSL Tally client will connect to. TCP normally should use port 8001 and UDP port 9001.
transport="udp"	A writable property, that configures the connection type that the TSL Tally client will connect to. Possible values are: "udp" or "tcp".
tally_out.0=0	This defines tally output mappings from GPI. The values can be: • 0 through 3 for GPI.0 to GPI.3 • 4 through 7 for inverse value of GPI.0 to GPI.3 • 8 for always clear • 9 for always set Maps the first tally to GPO.0
tally_out.1=3	Maps the second tally to GPI.3.
tally_out.2=4	
tally_out.3=7	
tally_out.4=8	
tally_out.5=8	
tally_out.6=9	
tally_out.7=8	
tally_in.0=0	• GPO.0 should be mapped to which received Tally. • Values 0 through 7 represents the 8 received tallies. • Value -1 leave the GPO not connected. GPO.0 should follow the first received tally.
tally_in.1=2	GPO.1 should follow the third received tally.
tally_in.2=7	
tally_in.3=-1	GPO.3 should not be connected to a received tally.

status="disconnected"	Displays the current connection status of the Tally Client. Possible values are: “active”, "connected", "disconnected".

3.10. The Ping client section

3.10.1 Section "[ping_client.ID]"

This “[ping_client.ID]” section contains a ping-based monitoring system that checks the availability of a specified host (IP address). If the host becomes unreachable, it can trigger actions like controlling GPO.

Example:

[ping_client.0]	
host="0.0.0.0"	Specifies the target host/IP address to be monitored.
status="disconnected"	Indicates the current connection status of the ping client. Possible values are: “initializing”, "online", “offline” or "disconnected".
gpo=0	This configures which GPO should be controlled, and how it should react when the target host is online/offline. Possible values are: • 0 – 3: GPO.ID is closed if online, otherwise open. • 4 - 7: GPO.ID is closed if offline, otherwise open. • 8 - 11: GPO.ID toggles while the host is detected as offline.
interval=500	Configure the time (in milliseconds) between each ping attempt. Range is 50 to 2000
timeout=400	Configure time (in milliseconds) to wait for a reply before considering the ping as failed. Range is 5 to 4*interval.
toggle_cooldown_interval=1000	Configures the cooldown period (in milliseconds) / toggle speed, if the GPO is set to toggle.

	Range is 50 to 60000.
errorcount_trigger=3	Sets the error threshold for detecting a failure in order to detect a host being offline. Range is 1 to 3. e.g., In this case If 3 consecutive pings fail, the system marks the host as offline.
failures=0	Displays the current number of ping failures detected. This resets when the host is reachable again.
average_response=65535	Provides the current average ping response time in milliseconds. If no reply was available, the value of 65535 will be returned.

3.11 The Firmware upload section

Firmware can be upgraded using a file upload using the "[firmware.upload]" section described here. Start by selecting this section.

- Then set the size parameter to the expected file upload size.
  - If successful, you will see that both the size and busy parameters will be set.
- If only the busy parameter is set to a non-zero value, it means another client is uploading. You can cancel any upload (including those from other clients) by setting the size parameter to 0.
- When the client has the handle for uploading content (with both size and busy set to non-zero values), data can be uploaded using a property called "chunk." The recommended size for each data packet is up to 64 bytes.

Examples:

[firmware.upload]	
size=0	Writeable property. This tells the server the size of the file the client wants to upload. Will only be non-zero for the client that currently has the handle. Set to zero to cancel any clients currently uploading.

busy=0	Set to non-zero value if a client has the upload-handle.
chunk="...."	Write-only property for performing the actual file upload.