TELNET PROTOCOL DOCUMENTATION FOR

The BTF1-series

Revision 2.0

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

Contents

1. Introduction

This document provides technical details on the Telnet protocol implementation for **BTF1-series**. It describes the command structure, authentication, 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. Authentication

When a client connects to a server, it will initially only dump the [btf1x] and [auth] sections, allowing the client to provide credentials.

Example:

```
[btf1x]
matrixLockLevel=2
matrixLockMessage="We are live, ask Dave before unlocking."
model="BTF1-41"
pcb_serialnumber="86620103"
pcb_version=2
right.model="n/a"
serialnumber="ser12345"
swUptime=14549
sysContact="Operator Foo"
sysLocation="The office"
sysName="Test Frame"
sysUptime=14558
version="1.3.0"
[auth]
user=""
```

The client should at this point request the [auth] section and provide a username.

Example:

[auth]

user="test"

If the username was excepted, the server will return that the user has been set and provide the two salts needed for the hashing algorithm (salt1 will be static until password has been reset, while salt2 is randomly chosen for each login attempt). If the username was rejected, the server re-issue the user="".

Example:

[auth]

user="admin"

salt1="\xf90\xd9\xfd\x8a\xa8\x9c\x1b\xf1\xe1|j\x830\x853F\n\x91\xcb\n\x1f?\x9c6\xccc
\xb2hi\xa0"

salt2="\xca*\x22\xd7\xeei\xdf\xf4\x12i\xd9K\xdco\xa3D\xd2\x9e\xe1\x9b\x1f>\x91\xb0`
\xdcI\x d5N\x07\x83\xdb"

At this step, the client should provide a correct hash2="XXXX" value. For blank passwords, the hash should be empty.

Here is pseudo code in C# for generating the hash:

```
byte[] Hash2(string Password, byte[] salt1, byte[] salt2)
{
    if (Password == "")
    {
        return ASCIIEncoding.ASCII.GetBytes(Password);
    }
    Rfc2898DeriveBytes hash1 = new Rfc2898DeriveBytes(Password, salt1, 5413);
    byte[] hash1_data = hash1.GetBytes(32);
    Rfc2898DeriveBytes hash2 = new Rfc2898DeriveBytes(hash1_data, salt2, 5235);
    byte[] hash2_data = hash2.GetBytes(32);
    return hash2_data;
}
```

And in C:

```c
#include <openssl/evp.h>
#include <openssl/aes.h>
#include <string.h>

int main (int argc, char *argv[])
{
   const char *password = "secret password";
   const char *salt1 = "salt1salt1salt1salt1salt1salt1sa";
   unsigned char hash1[32];
   const char *salt2 = "salt2salt2salt2salt2salt2salt2sa";
   unsigned char hash2[32];
   int i;
   printf ("salt1: %s\n", salt1);
   PKCS5_PBKDF2_HMAC_SHA1 (password,
                           strlen (password),
                           (const unsigned char *)salt1,
                           strlen (salt1),
                           5413,
                           sizeof (hash1), hash1);
   printf ("hash1: ");
   for (i=0; i < sizeof (hash1); i++)
   {
     printf ("\\x%02x", hash1[i]);
   }
   printf ("\n");
   printf ("salt2: %s\n", salt2);
   PKCS5_PBKDF2_HMAC_SHA1 ((const char *)hash1, sizeof (hash1),
                           (const unsigned char *)salt2,
                           strlen (salt2),
                           5235,
                           sizeof (hash2), hash2);
   printf ("hash2: ");
   for (i=0; i < sizeof (hash2); i++)
   {
     printf ("\\x%02x ", hash2[i]);
   }
   printf ("\n");
   return 0;
}
```

If the hash2 value is accepted, the server will return access="granted". If the hash is not accepted, the server will revert the username by setting user=""

[auth]

access="granted"

4. Sections

4.1 Section “[btf1x]”

This section contains identification, model and version information, along with access control settings for the matrix:

[btf1x]	
matrixLockLevel=2	This is a writable property, lock level for the matrix. 0= Unlocked, 1= Can move matrix crosspoints to predefined values as described in auto source, 2= fully locked.
matrixLockMessage="We are live, ask Dave before unlocking."	Message related to matrix lock status. Displays **"message"** when unlocking is attempted.
model="BTF1-41"	Identifies the device model.
pcb_serialnumber="86620103"	Serial number of the main-board (only applicable for BTF1-41).
pcb_version=2	Version number of the main-board (only applicable for BTF1-41).
right.model="n/a"	Model number of the right-side plugin-module on mainboard. (only applicable for BTF1-41)
serialnumber="ser12345"	Device serial number.
swUptime=14549	Software uptime in seconds.
sysContact="Operator Foo"	Contact person responsible for this device (writable)
sysLocation="The office"	Physical location of the system (writable).
sysName="Test Frame"	Device name / identifier (writable).
sysUptime=14558	System uptime in seconds.
version="1.3.0"	Running firmware/software version. (Barnmini-22 uses firmwareversion= instead)
outofsync=1	Initially this value is “1” when connecting.

	Set to “0” when all sections and properties have been transferred.

4.2 Section "[hardware.ID]"

These sections contain information about how the hardware blocks used by input and output ports are constructed.

Each section is named "[hardware.ID]", where ID is a numeric identifier. Currently, these sections contain only one property: ***name***. “Hardware.1” is always None, describing that no hardware is present at this location.

Examples:

[hardware.1]

name="None"

If this hardware block has parameters, these will be given by sections named "[hardware.ID.parameter.SUBID]" where SUBID is a number counting from 0.

These sections contain two properties: **name** and **syntax**. The syntax property is documented in the appendix.

Examples:

[hardware.2]	
name="M2135x"	Defines the name of this hardware block
[hardware.2.parameter.0]	Parameter of hardware.2
name="Rate"	Defines the name of this parameter
syntax="wE;0=Auto;1=Powered Down;2=Bypassed;4=SD;8=HD;12=3G"	Syntax/formatting is described in the appendix.
[hardware.8]	
name="LMH0387"	
[hardware.8.parameter.0]	
name="Direction"	
syntax="wE;0=Input;1=Output"	This example is a writable enumeration
[hardware.8.parameter.1]	

name="3G ext. reach"	
syntax="wB"	This example is a writable Boolean (“1” is true, “2” is false).
[hardware.8.parameter.2]	
name="Coarse amplitude"	
syntax="wE;0=800mV p-p;1=400mV p-p"	
…	

4.3 Section "[port.ID]"

These sections contain all parameters associated with each physical port in the BTF1 frame.

Each section is named "[port.ID]", where ID starts from 1.

- If the port is not unidirectional, the corresponding input.* and output.* variables will be skipped.
- The word external is if the functionality is built directly into the matrix circuit or not.
- If the port is an SFP port, the SFP.* variables will be included.

Examples:

[port.4]	
name="SFP #4"	The actual name of the port. Read-only.
SFP.present=1	Is an SFP detected in the port? 0=No, 1=Yes
SFP.connector="LC"	What kind of connector does this SFP have. Most common is LC.
SFP.lengths="10000;0;0;0;0;0"	Which cable lengths is this SFP designed for? The 5 numbers represent single mode fiber, multi-mode fiber OM1, OM2, OM3 and copper.
SFP.vendorname="Barnfind Tech"	The vendor of the SFP module.
SFP.partnumber="BT-LX-SM-12G10"	The part number of the SFP module.
SFP.serialnumber="A89T140016"	The serial number of the SFP module.

SFP.productiondate=20181107	The production date of the SFP module (given in yymmdd format).
SFP.productionlot=""	The production lot of the SFP module.
SFP.revision="0"	The revision of the SFP module.
SFP.wavelengthnm=1310	The wavelength of the transmitter given in nm (for DWDM, the digits behind the decimal point are missing).
SFP.bitrate="12000000;600000;12600000"	This will contain 3 numbers separated by (;) if a SFP module is present. The first number will be the nominated designed bitrate given in KHz. The second will be the minimum and the third will be the maximum designed bitrates.
input.label="Input port 4"	User-writeable label for the input part of the port.
input.exteq.type=1	Specifies the type of external equalizer used for signal conditioning. (refer to [hardware.ID] for specific details).
input.exteq.parameters=""	User-writeable parameters, lookup in [hardware.ID] for syntax.
input.exteq.signaldetected=0	Signal detection for external equalizer. 0=Not able to detect (SFP module not inserted, or similar), 1=Detected, 2=Not detected.
input.extrc.type=1	Specifies the type of external reclocker for stabilizing SDI signals (refer to [hardware.ID] for details).
input.extrc.locked=0	Indicates if the external reclocker is locked to a signal. 1=Locked, 0=Not locked.
input.extrc.lockedat=0	Bitrate at which the reclocker is locked, in KHz.
input.extrc.parameters=""	User-writeable parameters for reclocker. Refer to [hardware.ID] for syntax.
input.extrc.signaldetected=0	Signal detection for the external equalizer.

input.inteq.type=15	Specifies the type of internal equalizer used for this input. (refer to [hardware.ID] for details).
input.inteq.parameters="1;2"	User-writeable parameters for internal equalizer.
input.inteq.signaldetected=2	Signal detection for the internal equalizer.
input.analyzer.type=25	Type of SDI analyzer used for this input. A value other than 1 (“None”) indicates the presence of an analyzer. Analyzer differs from the other hardware blocks, that the parameter is stored in “.enabled” instead of “.parameter” and always being a number.
input.analyzer.enabled=2	User-writeable parameter. Is the SDI analyzer enabled for this input port? See relevant [Hardware.ID] section for possible values.
input.analyzer.rawformat=""	First part of the SDI detection routine.
input.analyzer.standardformat=""	Second part of the SDI detection routing, trying to match payload and rawformat against SMPTE standards.
input.analyzer.errors=""	If no errors are detected, "none" will be given. If errors are detected, they will be concatenated with a space between them and at the end. Possible errors as of today are NOSIGNAL EAV SAV LineNo CRC-Luma CRC-Chroma CRC-ANC-Luma CRC-ANC-Chroma CRC-EDH-ActivePicture CRC-EDH-FullPicture VideoStandard.
input.analyzer.signaldetected=2	Signal detection for the signal analyzer.
output.label="Output port 4"	User-writeable label for the output part of the port.

output.source=4	User-writeable signal source for this port. -1 means no input selected. Uses the same ID numbers as in port.ID.
output.syncsource=65535	User-writeable reference sync used when source is changed. 65534 – analog reference, 65535 – no sync selected, or a port number. (Only available on BTF1-x 3G-SDI generation)
output.extrc.type=14	Specifies the type of external reclocker used for output signals, ensuring a stable signal for downstream devices (refer to [hardware.ID] for details).
output.extrc.locked=0	Indicates if the reclocker is locked. 1=Locked, 0=Not locked.
output.extrc.lockedat=0	What bitrate is this reclocker locked at (given in KHz)
output.extrc.parameters="0"	User-writeable parameters (refer to [hardware.ID] for details).
output.extrc.signaldetected=2	Signal detection for the external reclocker.
output.extcd.type=1	Specifies the type of external cable driver used for output (refer to [hardware.ID] for details).
output.extcd.parameters=""	User-writeable parameters (refer to [hardware.ID] for details).
output.extcd.signaldetected=2	Signal detection for the external cable driver.
output.autosource.mask=" 00000000000000000000000000000000- 00000000000000000000000000000000- 00000000000000000000000000000000- 00000000000000000000000000000000"	User-writeable, auto-changeover mask, 4 groups of parameters. One digit per .input port object in each group. The first group selects if input is 0=not used, 1=backup or 2=main. The second group selects if input is sensitive to LOS (loss of signal) 0=not sensitive and 1=sensitive. Third group selects if input is sensitive to SDI signal analyzer being able to lock onto the

	signal 0=not sensitive and 1=sensitive. Forth group selects if input is sensitive to SDI signal analyzer errors 0=not sensitive and 1=sensitive.
output.autosource.enabled=0	User-writeable parameter, for enabling automatic changeover active for this output. 0=disables, 1=enabled and 2=temporary disabled.
output.autosource.status=0	Status of the automatic changeover 0=normal operation, 1=degraded (one or more input failed) and 2=failure (no valid inputs available).
output.autosource.timeout=1000	User-writeable parameter. How long does an input signal needs to be continuously detected as failed/normal before input status changes. This is given in “ms” and range is 1- 60000.
output.autosource.switchbackenabled=0	User-writeable parameter. Use the main/backup part of the input channel selection. When enabled, it will switch from backup to main, even while the active selected backup has a valid signal.
output.autosource.switchbacktimeout=30000	User-writeable parameter. This is the timer running when main signal becomes available, and the frame is on a backup channel. The value is given in “ms” and range is 1- 120000.

4.4 Section for “SFP diagnostics”

SFP modules can have multiple diagnostics entries dynamically allocated. These entries will be located into sections with the following name scheme: “[port.ID.SFP.diag.SUBID]”. Here, the "ID" refers to the port number to which the module belongs, while "SUBID" represents the specific diagnostic entry for the SFP module.

Each section will contain 3 properties: the ***name***, ***value*** and the ***syntax***. (For description of the syntax, please see the appendix.)

If a diagnostics-entry is removed (e.g., when the SFP module is unplugged), the property named ***remove*** will be set to =1 in the affected sections.

Examples:

[port.8.SFP.diag.1529]	This list of diagnostics available for a given SFP is dynamic. Each entry will have a different number, in this example we have diagnostic entry “1529”
name="User selectable link speed"	The name of the diagnostic entry.
syntax="wE;1=Auto(default);2=1Gbps Full Duplex;3=1Gbps Half Duplex;4=100Mbps Full Duplex;5=100Mbps Half Duplex;6=10Mbps Full Duplex;7=10Mbps Half Duplex"	Syntax is described in the appendix. In this example, a writable enumeration (wE) allowing selection of different link speeds.
value="1"	According to syntax above, the current link speed is set to Auto. Number will be stored as text, since syntax could allow text too.
remove=0	The diagnostic entry is active and not marked for removal.
[port.8.SFP.diag.1530]	In this example we have diagnostic entry “1530”
name="Operating mode"	The name of the diagnostic entry.
syntax="wE;1=SGMII(default);2=1000BASE-X with copper auto-neg (GBIC);3=1000BASE-X without copper auto-neg"	Syntax is described in the appendix. In this example, a writable enumeration (wE) for selecting the SFP operation mode.
value="1"	According to syntax above, the current operating mode is set to “SGMII(default)”.
remove=0	
[port.8.SFP.diag.12]	In this example we have diagnostic entry “12”
name="Present"	The name of the diagnostic entry.

syntax="rE;0=Empty;1=MSA;2=nonMSA"	Syntax is described in the appendix. In this example, this read-only enumeration (rE) specifies if an inserted SFP module is Empty (0), MSA-compliant (1), or non-MSA (2)
value="1"	According to syntax above, “MSA” is the current state of the SFP port.
remove=0	

4.5 Section for “self-upgrade via Internet”

The original method for performing a firmware upgrade on the BTF1-x frame was to trigger the unit to self-upgrade via the Internet. For this to function, the frame must be connected to an IP network with Internet access and a valid DNS server. All the necessary information to use this feature is contained within the section named "[firmware.aptengine]".

Example:

[firmware.aptengine]	
state=1	The current state of the upgrade engine. 1=idle, 2=idleUpdatesAvailable, 3=init, 4=checking and 5=upgrading
trigger_update=0	Set this to one to trigger the engine to search for updates
trigger_upgrade=0	Set this to one to trigger the engine to download and install the update found by update
clearlog=0	This property will be set to 1 and back to 0 when the frame firmware clears the log This is the output from the init/update/upgrade process.
log.1="Health check #1"	The returned log file. Each line will have a separate log.ID entry.
log.2="Health check #2"	Second log line, and so on.
…	

4.6 Sections for “upgrade via file upload (and USB)”

The other two methods of firmware upgrade are either via file-upload or by USB memory sticks. Both upgrade methods will cause the same logging mechanism to be used, found in the “[firmware.lastlog]” section.

- File upload can be done 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 1024 bytes.

Example:

[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.
[firmware.lastlog]	
active=0	This goes non-zero every time an upgrade software is executing. If an upload is ongoing, this will not go non-zero until upload is complete. When this property goes from non-zero to 0, the clients should clear its local copy of the log file.
line.0="foo"	The returned log file. Each line will have a separate log.ID entry.
line.1="bar"	Second log line, and so on.

4.7 Section "[authdb.ID]"

This section contains the username / password combinations used by the telnet/web API. Each user entry will have a separate "[authdb.ID]" section, and all properties are writeable.

- To create a new user, simply open an authdb section with an unused ID number and fill in username, realname, salt1 and hash1.
- To change password for a user, set salt1 and hash1. Algorithm for hash1, please see the chapter 3 authentication.
- To remove a user, set the remove property to 1.
- Regarding access level, these are the definitions so far:

  0 = Disable

  100 = Guest (read-only)

  200 = Panels (future use)

  300 = Administrator (write access)

Example:

[authdb.1]	
username="admin"	Refer the chapter 3 authentication. This is a writable property, the username for this user entry is "admin".
realname="The Boss"	This is a writable property, the real name of the user.
salt1="f930d9fd8aa89c1bf1e1 7c6a83308533460a9120cb0a 1f3f9c36cc63b26869a0"	Refer the chapter 3 authentication. This is a writable property, the salt1 used for generating hash1.
accesslevel=300	This is a writable property, the user's access level in the system.
remove=0	This is a writable property. To remove a user, set the remove property to 1.
hash1="(secret)"	This is write-only property. The hash will never be returned from the device.
...	

4.8 Section "[diag.ID]":

Diagnostics related to the frame's health are provided in sections named "[diag.ID]", with each ID representing a separate diagnostic entry. Each diagnostic entry contains four properties: **name**, **value** (a numeric value), **sendToLed** (indicating whether the diagnostics can trigger the front warning LED to blink), and a **syntax** description (see appendix).

Examples:

[diag.33]	
name="Power-2 input voltage"	The name of the entry
value=120	The numeric value
sendToLed=1	User writeable property. If set to 1, the front LED on the frame will blink if the value is outside the warning/error thresholds. To disable this diagnostics entry, set this value to 2.
syntax="rI;wmin=112;wmax=130; emin=110;emax=132;suffix=V; scale=0.100000"	For description of the syntax, please see the appendix
…	

4.9 Section "[logo]"

If the frame features an LED-illuminated logo on the front left, this section will appear. It contains two properties: **brightness** and **findme**.

- brightness: A numeric value from 0 to 100, specifying the LED intensity.
- findme: Determines how long the front LED blinks (in seconds) to help identify the frame.
    - To disable blinking, set this value to 0.
    - The maximum allowed value is 86,400 (24 hours).

Example:

[logo]	
brightness=100	Sets the logo brightness to 100% (full brightness).
findme=0	Findme function is currently disabled (0); Set to number of seconds the logo should blink, used to highlight or locate the device visually in your rack.

4.10 The Network sections

On the server-side, the network is handled by a separate process, but communication is mixed into the same telnet stream. This should be fully transparent for the client.

4.10.1 Section "[network]"

The first section within the network namespace is named "[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 used for this).

Example:

[network]	
version="1.1.0"	Specifies the software version of the network handling software being used (standalone software, not part of the main program in BTF1-xx).
id="b8:27:eb:17:3e:e2"	The MAC address of the network interface/Unique ID of the device. (in addition to Serial Number)

4.10.2 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]="192.168.0.87"	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, otherwise “0.0.0.0”.
ipv4.dns.server[0]="8.8.8.8”	Up to 3 DNS servers can be given when configured using static IPv4 addresses.
ipv4.dns.server[1]="0.0.0.0"	
ipv4.dns.server[2]="0.0.0.0"	
ipv4.dns.search[0]=""	Up to 4 DNS searches can be given when configured using static IPv4 addresses.
…	
ipv4.dns.search[3]=""	
ipv4.mode=2	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.dns.server[0]="::"	Up to 3 DNS servers can be given when using static IPv6 addresses.
ipv6.dns.server[1]="::"	
ipv6.dns.server[2]="::"	
ipv6.dns.search[0]=""	Up to 4 DNS searches can be given when using static IPv6 addresses.
…	
ipv6.dns.search[3]=""	
ipv6.mode=10	Operation mode for IPv6. 8=LinkLocal only, 9=DHCP/StateFull, 10=Static,12=StateLess/Router Advertisement.
ntpserver[0]=""	Up to 4 NTP servers can be provided, starting from 0. The internal time/data is not critical for any functionality per current firmware. The first NTP server slot is empty.
ntpserver[1]=""	
ntpserver[2]=""	
ntpserver[3]=""	
hostname=""	Primarily used as identification in DHCP requests.
commit=0	Set this value to 1, in order to commit any changes.

4.10.3 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=""	No network interface is assigned.

addr=""	No IP address is assigned.
prefix=0	No subnet mask is assigned.
...	
[network.status.ip.26]	
dev="eth0"	Assigned to the Ethernet interface eth0.
addr="2002::2123"	An IPv6 address assigned to this interface.
prefix=64	Current prefix for the IPv6 address.
[network.status.ip.27]	
dev="eth0"	Assigned to the Ethernet interface eth0.
addr="fe80::ba27:ebff:fe17:3ee2"	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	
[network.status.ip.28]	
dev="eth0"	
addr="2001::2"	This is another IPv6 address configured for eth0. Multiple addresses can be assigned to a single interface.
prefix=64	
[network.status.ip.29]	
dev="eth0"	
addr="192.168.0.87"	This is an IPv4 address assigned to interface eth0 (192.168.0.x).
prefix=24	Subnet mask of 255.255.255.0 (/24 means that the upper 24 bits in the subnet mask is set to 1).
...	
[network.status.ip.31]	
dev="lo"	Assigned to the loopback interface.
addr="127.0.0.1"	The loopback IP address (localhost). 127.0.0.1 is a special address used for local communication within the device. It is not reachable from outside the device.

prefix=8	Subnet mask of 255.0.0.0 (/8 means that the upper 8 bits in the subnet mask is set to 1).

4.10.4 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=""	No network interface assigned.
dst=""	No destination address assigned.
dstprefix=0	No subnet mask.
gw=""	No gateway assigned.
...	
[network.status.route.61]	
dev="eth0"	The Ethernet interface eth0 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.62]	
dev="eth0"	
dst="192.168.0.0"	This route covers the 192.168.0.0/8 network.
dstprefix=8	Subnet mask 255.255.255.0, covering 192.168.0.0 - 192.168.0.255.

gw=""	No gateway (local network traffic). This means traffic within the 192.168.0.x network is routed locally without needing a gateway.
[network.status.route.63]	
dev="eth0"	
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).

4.10.5 Section "[network.status]"

This section currently contains only one property: **resolvconf**. The active DNS configuration file, known as /etc/resolv.conf in Unix systems.

Example:

[network.status]	
resolvconf="nameserver 8.8.8.8\n"	The current selected nameservers and search prefixes will be visible here. Can be multiline, hence \n is present.

4.10.6 Section "[network.8021x] "

This section handles 802.1x authentication settings, which are used for secure network access in enterprise environments. For enterprise networks that are configured with 802.1x security that needs authentication to access the network.

supports property having the lists the supported authentication methods:

- NONE - No authentication.
- MD5 - Simple challenge-response authentication.
- PEAP-MSCHAPV2 - Secure authentication method often used in enterprise networks.

Example:

[network.8021x]	
supports="NONE MD5 PEAP-MSCHAPV2"	Lists the supported authentication modes. The modes currently supported (read-only)
mode="NONE"	802.1x authentication is disabled “NONE”. This property is writable.
eapol_version=2	Uses EAPOL version 2 (Extensible Authentication Protocol over LAN). This property is writable.
username=""	This is a writable property, the username used for 802.1x authentication.
password="(secret)"	Write-only, password will not be returned from the device, but replaced with “(secret)”
commit=0	Set this value to 1, in order to commit any changes.

4.10.7 Section "[network.log]"

This section contains logs from the networking software. The network configuration server maintains a finite number of log entries. As the log files grows, older entries are removed, meaning the log may no longer start from entry 1.

If the networking software restarts while connected to the Telnet server, it will generate a new log, starting from 1 again.

Each log entry consists of three parts, separated by a semicolon (;):

1. Source of the entry
2. Channel (either stdout or stderr)
3. Log message / data output

Example:

[network.log]	
entry.1="main;stdout;starting configuration static for ipv4\n"	
entry.2="main;stdout;starting configuration link-local and static for ipv6\n"	
…	

4.11 The Tally/GPIO sections

There are multiple instances of tallies and GPIO available. They all share some common structure and syntax described in 4.11.1 and 4.11.2. Each output has available the concept of "scripts" that make them configurable for automation.

4.11.1 General Information for TSL Tally/GPIO Section

This chapter describes the possibility of *scripting* in the GPIO and TSL Tally sections. The master section for these includes the properties: **script_maxlength**, **script_maxtokens** and **script_supports**.

- **script_maxlength** = Specifies the maximum length (in characters) of a script that can be used for each script.
- **script_maxtokens** = Defines the maximum number of tokens allowed in a script (tokens are logical/conditional operators, functions, and variables used in the scripts).
- **script_supports="& | ! False True G M L C S"** = This parameter defines the supported functions and tokens available for scripting logic in the system.
    - **Boolean Constants & Operators**

        "False" = Always returns false (logic condition is never met).

        "True" = Always returns true (logic condition is always met).

        "&"(AND) = &(expression1, expression2)
        - Returns true if both expressions are true.

        "|"(OR) = |(expression1, expression2)
        - Returns true if either or both expressions are true.

        "!"(NOT) = !(expression)
        - Inverses the expression. It returns true if the expression is false, and vice versa.

    - **Functions**

        "G"(GPI Input Detection) = G(Input Number)
        - Checks the state of a GPI.

- The input number should be the GPI.n, counted from 0.
- Examples:

    G(0) checks GPI.0

    G(1) checks GPI.1

“M” (Matrix Input/Output Mapping) = M(Matrix Output, Matrix Input)

- Monitors matrix connections (mapping inputs to outputs).
- Returns true if the given output, has input selected as its source.
- Matrix output number count from 1 (Matches [port.ID])
- Matrix input number count from 1 (Matches [port.ID])
- Example:

    M(1,3) = Evaluates to true if “Output port 1” is connected to “Input port 3”.

“L” (Loss Detection at Input) = L(input number)

- Detects signal loss at a specific input port.
- The number should be the Input number, counting from 1 (Matches [port.ID])
- Example:

    L(1) = checks “Input port 1”

"C" (TSL Tally Client Detection) = C(TSL Tally Client, Inbound Tally)

- Checks the state of an Inbound Tally signal for a specific Tally Client.
- Tally Client, should be the ID of the Tally Client that should be evaluated (Matching TSLTallyClient.ID)
- Inbound Tally, is which tally from the client to evaluate, counting from 0.
- Example:

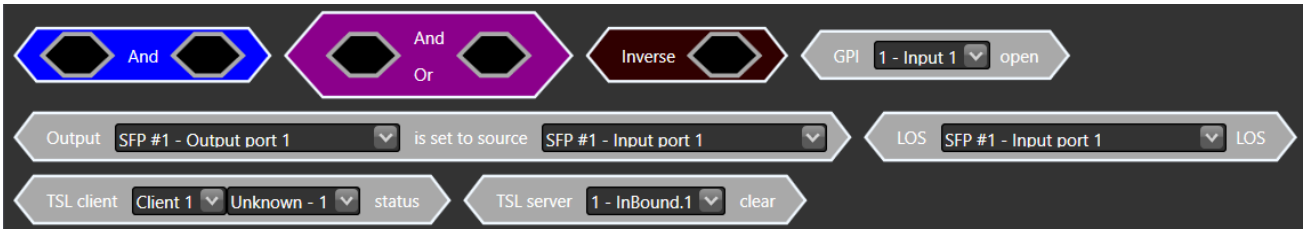
    C(0,4)= Evaluate the first TSL Tally client, Inbound Tally number 5 when counting from zero.

"S" (TSL Tally Server Detection) = S(Inbound Tally Number)

- Checks an Inbound Tally signal at the TSL Tally Server.
- Inbound Tally, is which tally from the server to evaluate, counting from 0
- Example:

    S(0) = Evaluate Inbound Tally 1 on the TSL Tally Server.

Examples of how BarnStudio presents the functions as given in **"script_supports"** (BarnStudio does not show False and True in the toolbar):

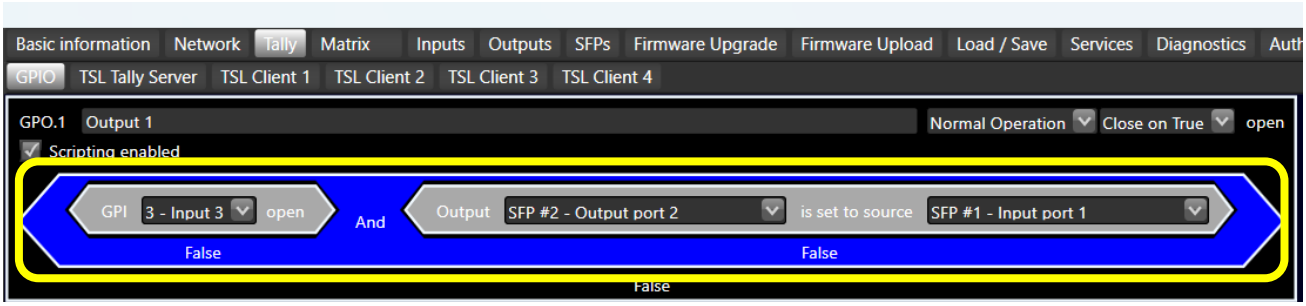


4.11.2 Tally OutBound/GPO sections

This chapter describes the possibility of *scripting* in the GPIO and TSL Tally sections, and other common properties. Each tally output / GPO section will have the properties: **state**, **label**, **control**, **scriptpolarity** and **script**.

- **state** = Defines the current detected state of the input. For GPO, the valid values as "open" and "closed", while outbound tallies are "set" and "clear". This property is writable if **control**="manual".
- **label** = A user configurable name for the tally output or GPO, helping users easily identify its purpose.
- **control** = This writable property controls how the output is managed.
    - "manual" = The output is manually controlled by writing to **state**.
    - "script" = The output is controlled with a script.
- **scriptpolarity**= This property specifies how the output state changes using the script result. The possible values differ for GPO, and TSL Tally. For GPO the possible values are "close_on_true" and "close_on_false", while for TSL Tally the possible values are "set_on_true" and "set_on_false".
    - "close_on_true" means that when the script evaluates to true, the output will be in a "closed" state, shorting the pins in the GPO connector. Otherwise, the pins will be remain open/disconnected in the GPO connector.
    - "close_on_false" is the inverse of close_on_true.
    - "set_on_true" means that when the script evaluates to true, the transmitted tally will be 1 / logically set.
    - "set_on_false", is the inverse of set_on_true.

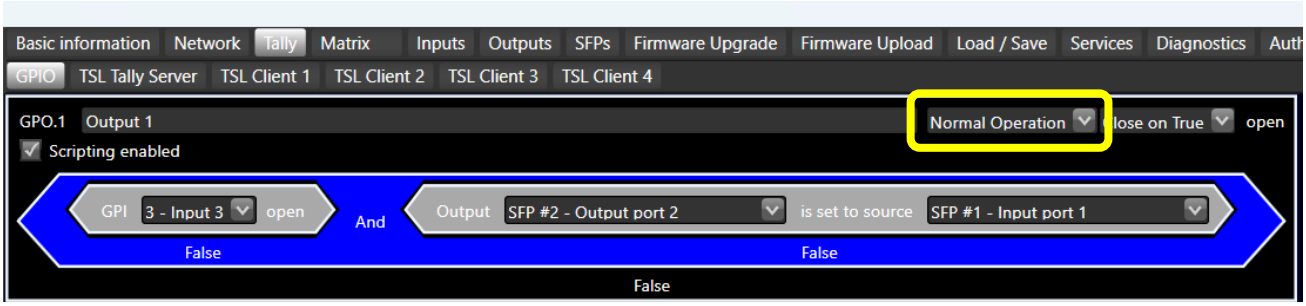
- **script="&(G(2),M(2,1))"** = The script controlling this output. In this example we used the following functions:
    - **G(2)** = Checks the state of **GPIO input 3 (index counting from zero)**.
    - **M(2,1)** = In the **Matrix**, look at output **"2"**, is input **"1"** selected?
    - **& (**G(2),M(2,1)**)** = The result will be true **only if both conditions are true**.



4.11.3 Section "[GPIO]" (*ONLY FOR BTF1-41)

This section contains the General-Purpose Input/Output (GPIO) settings, including script configuration, available functions, and LED control. This section is named "[GPIO.ID]".

- **function** property having the list of available output logic that will be applied to the **script**:
    - **toggle** = Keeps toggling as long as the evaluation of the **script** is true. The toggle speed is adjustable.
    - **normal** = Follows the **script** as given.
    - **oneshot** = Momentarily changes state when the **script** evaluates as true and remain there until an explicit reset is provided.



Example:

[GPIO]	
script_maxlength=220	Refer to chapter 4.11.1.

script_maxtokens=40	Refer to chapter 4.11.1.
script_supports="& | ! False True G M L C S"	Refer to chapter 4.11.1.
functions="toggle;normal;oneshot"	Refer to text-block above.
led_control="GPI"	This is a writable property, determining how the LEDs at the GPIO connector are controlled. • "GPI" LEDs are controlled by a GPI (on when detected high/floating). • "!GPI" LEDs are controlled by GPI, but with inverted logic (on when detected low/shorted). • "GPO" LEDs are controlled by a GPO (on while closed) • "!GPO" LEDs are controlled by GPO (on while open)

4.11.3.1 Section "[GPI.ID]" (*ONLY FOR BTF1-41)

This section contains General-Purpose Input (GPI), defining its state and label. Each section is named "[GPI.ID]", where ID starts from 0. Each section will contain only two properties: **state** and **label**.

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" or "close"	The current detected state of the input. "open" = Circuit is detected as open (high voltage). "close" = Circuit is detected as closed (low voltage).
label="Input 1"	A user-configurable label for [GPI.ID].
…	

4.11.3.2 Section "[GPO.ID]": (*ONLY FOR BTF1-41)

This section contains General-Purpose Output (GPO), defining its state, control method, and script logic. Each section is named "[GPO.ID]", where ID starts from 0. Each section will contain nine properties.

BarnStudio adds 1 to the ID number when presented to the users (e.g., GPO.0 is presented as "GPO.1").

Example:

[GPO.0]	
state="open"	Refer to chapter 4.11.2. Defines the current state of the output. "open" = The pins on the output are floating/open. "closed" = The pins on the output are shorted together (using a relay). This is a writable if **control**="manual"
label="Output 1"	Refer to chapter 4.11.2.
control="script"	Refer to chapter 4.11.2.
scriptpolarity="close_on_true"	Refer to chapter 4.11.2.
script="&(G(2),M(2,1))"	Refer to chapter 4.11.2.
toggle_timer=500	When in toggle mode, this sets the toggle interval in milliseconds.
oneshot_reset=1	In oneshot mode, writing back a 1 will reset the oneshot.
oneshot_status=0	Indicates that the oneshot function has been fired. If it has been fired, a reset will be needed to recharge the functionality.
function="normal"	Refer to chapter 4.11.3. Defines the current function that should be applied to the script (can be toggle, normal, or oneshot).
...	

4.11.4 The TSLTally sections

4.11.4.1 Section "[TSLTallyClient]"

This section contains the configuration for the TSL Tally Clients, which connects to the TSL Tally Server to exchange tally (on-air) status updates.

This section will contain only three properties: **script_maxlength**, **script_maxtokens** and **script_supports**. Please refer to chapter 4.11.1.

Example:

[TSLTallyClient]	
script_maxlength=220	Refer to chapter 4.11.1.
script_maxtokens=40	Refer to chapter 4.11.1.
script_supports="& | ! False True G M L C S"	Refer to chapter 4.11.1.

4.11.4.2 Section "[TSLTallyClient.ID]"

Each section here contains one available TSL Tally client (for connecting to a remote TSL Tally server). Each section is named "[TSLTallyClient.ID]", where ID starts from 0. These sections contain only two properties: **host** and **status**.

(**The InBound and OutBound tallies are not visible, unless the client is connected to a server*)

Example:

[TSLTallyClient.1]	
host="192.168.0.1;8001;tcp"	Writable parameter that defines the target host connection details using three fields “A;B;C” separated by semicolons(;) Field 1(A): The target IPv4/IPv6 Address (left empty for non-used clients). Field 2(B): The target port number. Field 3(C): Specifies the protocol to use, either “tcp” or “udp”

	Example: host="192.168.1.10;8001;tcp" ○ Connect to remote host 192.168.1.10 on port 8001 using TCP.
status="Idle"	The current status of the tally client.
…	

4.11.4.3 Section "[TSLTallyClient.ID.InBound.SUBID]"

These sections contain the inbound tally signals received by the TSL Tally Client.

Each section is named "[TSLTallyClient.ID.InBound.SUBID]", where SUBID starts from 0. Each section will contain only two properties: **state** and **label**.

(**The InBound and OutBound tallies are not visible, unless the client is connected to a server*)

Example:

[TSLTallyClient.0.InBound.0]	
state="clear"	The current state of the inbound tally signal. Possible values: • "clear" – The inbound tally signal is inactive. • "set" – The inbound tally signal is active.
label="InBound.1"	A custom name for this inbound tally input, used for identification.
…	

4.11.4.4 Section "[TSLTallyClient.ID.OutBound.SUBID]"

These sections contain the outbound tally signals, which is transmitted via the TSL Tally Client.

Each section is named "[TSLTallyClient.ID.OutBound.SUBID]", where SUBID starts from 0. Each section will contain five properties: **state**, **label**, **control**, **scriptpolarity** and **script**.

(**The InBound and OutBound tallies are not visible, unless the client is connected to a server*)

Example:

[TSLTallyClient.0.OutBound.0]	
state="set"	Refer to chapter 4.11.2.

	Indicates the current status of this outbound tally. Possible values: • "clear" – The tally signal is inactive. • "set" – The tally signal is active.
label="OutBound.1"	Refer to chapter 4.11.2.
control="manual"	Refer to chapter 4.11.2.
scriptpolarity="set_on_true"	Refer to chapter 4.11.2.
script=""	Refer to chapter 4.11.2.
…	

4.11.4.5 Section "[TSLTallyServer]"

This “[TSLTallyServer]” section contains the configuration for the TSL Tally Server, which exchange tally data with connected clients.

Example:

[TSLTallyServer]	
script_maxlength=220	Refer to chapter 4.11.1.
script_maxtokens=40	Refer to chapter 4.11.1.
script_supports="& | ! False True G M L C S"	Refer to chapter 4.11.1.
tcpport=8001	The TCP port the server listens on for incoming connections.
udpport=9001	The UDP port the server listens on for incoming connections.
tcpstatus="listening"	The server is actively listening for incoming TCP connections.
udpstatus="listening"	The server is actively listening for incoming UDP packets.
tcpclient.0=“[192.168.0.121]:46794”	These entries show the clients connected to the Tally Server via TCP.

	A client is currently connected from IP 192.168.0.121, using port 46794.
tcpclient.1=""	These entries are empty, meaning no additional clients are connected.
tcpclient.2=""	
tcpclient.3=""	
tcpclient.4=""	
tcpclient.5=""	
tcpclient.6=""	
tcpclient.7=""	
udpclient.0=""	These entries show the clients connected to the Tally Server via UDP. These entries are empty, meaning no UDP clients are currently connected.
udpclient.1=""	
udpclient.2=""	
udpclient.3=""	
udpclient.4=""	
udpclient.5=""	
udpclient.6=""	
udpclient.7=""	

4.11.4.6 Section "[TSLTallyServer.InBound.ID]"

These sections contain the inbound tally signals received by the TSL Tally Server. Inbound tallies indicate signals coming from an external source, such as switcher.

Each section is named "[TSLTallyServer.InBound.ID]", where ID starts from 0. Each section will contain only two properties: **state** and **label**.

Example:

[TSLTallyServer.InBound.0]	
state="clear"	Indicates the current status of this inbound tally signal. “clear” or ”set” • "clear" = The inbound tally signal is inactive. • "set" = The inbound tally signal is active.

label="InBound.1"	A custom name for this tally input, used for identification.
…	

4.11.4.7 Section "[TSLTallyServer.OutBound.ID]"

These sections contain the outbound tally signals, which is sent from the TSL Tally Server to other devices, such as camera tally lights or monitor overlays.

Each section is named "[TSLTallyServer.OutBound.ID]", where ID starts from 0. Each section will contain five properties: **state**, **label**, **control**, **scriptpolarity** and **script**.

Example:

[TSLTallyServer.OutBound.0]	
state="clear"	Refer to chapter 4.11.2. Indicates the current status of this outbound tally. Possible values: • "clear" – The tally signal is inactive. • "set" – The tally signal is active.
label="OutBound.1"	Refer to chapter 4.11.2.
control="manual"	Refer to chapter 4.11.2.
scriptpolarity="set_on_true"	Refer to chapter 4.11.2.
script=""	Refer to chapter 4.11.2.
…	

4.12 The Service sections-

4.12.1 Section "[service.ID]"

Each possible service will be represented by its own section.

- Services that have a configurable TCP listening port will have a property named "**DynamicTCPPort**".
- Services that have a fixed TCP listening port will have a property named "**FixedTCPPort**".

- Services that have a fixed UDP listening port will have a property named "**FixedUDPPort**".
- This is a writable property, services that can be disabled will have a property named "**enabled**" that can be set to either 1 or 0.

If a service has connections, there will be [client.ID] properties, one per connection.

Example:

[service.http]	
status="ready for clients"	The status for this service.
FixedTCPPort=80	The port number used by this service. This service is fixed to TCP port 80.
description="HTTP / web server for control"	The description for the given service.
enabled=1	This property makes it possible to disable this service.
...	
[service.telnet-alt]	
status="disabled"	The alternative Telnet service is not active.
DynamicTCPPort=10050	The port number used by this service. This service is configurable and is currently configured to port 10050.
description="Telnet, alternative port - Control Protocol"	
enabled=0	This service is disabled.
[service.telnet]	
status="ready for clients"	
FixedTCPPort=23	
description="Telnet - Control Protocol"	
client.20="[192.168.0.117]:52330; connected;authenticated;1"	A client is connected. This string is separated into 3 or 4 parts, separated with ; characters. Part 1: The remote IP address and port. Part 2: The status of the connection.
client.21="[fe80::12d5:f7a9:50e3:3122]:54963;connected;authenticated;1"	

client.22="[192.168.0.107]:51598;connected;authenticated;1"	Part 3: The status of authentication. Part 4: The user number [auth.ID] (only available for remote connection)
client.23="[192.168.0.128]:52175;connected;authenticated;1"	
client.26="[::1]:56272;local service"	The rest of these clients are connected from ::1 (localhost), meaning they are local services running on the same device.
client.27="[::1]:56284;local service"	
client.28="[::1]:56300;local service"	
...	
[service.snmp]	
status="ready for clients"	
FixedTCPPort=161	
FixedUDPPort=161	The port number used by this service. This service is fixed to UDP port 161.
description="SNMP control protocol"	
enabled=1	

5. APPENDIX A: Syntax specifier

Various parts of the server software use a syntax string to define how a parameter should be interpreted. A syntax specifier can consist of multiple components or properties, each separated by a semicolon (;).

The first property will always define the data type and protection. The first character indicates whether it is read-only or writable: r for read-only and w for writable. The remaining characters specify the data type:

- **B** for boolean:

  (0 = unset, 1 = true, 2 = false). A writable boolean will normally fail if you try to set it to 0.
- **I** for integer:

  This may be followed by options to apply scaling, warnings, error limits, and suffixes.

  Example:
  - I;scale=0.1;wmin=10;wmax=100;emin=0;emax=110;min=-100;max=200;offset=10;suffix=mW

- **E** for enumeration:

  The following properties will define the available options.

  Example:1=option1; 2=option2

- **S** for string:

  Possible properties include len=x to limit the string length. The redundancy switch has a special rule when setting autosource-mask-v1. This string is expected to consist of 4 series of digits separated by hyphens (-) sign. Each digit matches a valid input port from the port that has input properties.

  The groups represent:

  - The first group: "Enabled" column (0 = not used, 1 = main, 2 = backup)
  - The second group: "Sensitive to LOS" column (0 = not set, 1 = set)
  - The third group: "Sensitive to analyzer lock"
  - The fourth group: "Sensitive to analyzer errors" column.

Future Data Types:

- **IPv4** for an IPv4 address.
- **IPv4Net** for an IPv4 address with a prefix (e.g., 192.168.0.255/24).
- **IPv6** for an IPv6 address.
- **IPv6Net** for an IPv6 address with a prefix.
- **IP** for an IPv4 or IPv6 address.
- **IPNet** for an IPv4 or IPv6 address with a prefix.

6. Change log

Additions in Revision 2.0 (Since Revision 1.0):

This revision introduces new sections focusing on network configuration, the Tally/GPIO, the TSLTally and service Section to enhance the documentation.

The following sections have been updated in this revision:

2. Command structure
 - Added that a value can be deleted by newer firmware versions

4.1 Section [btf1x]
 - Added “logo”, “MatrixLock” and “outofsync”

The following sections have been added in this revision:

4.1 Section [btf1x]

4.10.6 Section [network.8021x]

4.11 The Tally/GPIO Sections

- 4.11.1 General Information for TSL Tally/GPIO
- 4.11.2 Tally OutBound/GPO Sections
- 4.11.3 Section [GPIO]
    - 4.11.3.1 Section [GPI.ID]
    - 4.11.3.2 Section [GPO.ID]
- 4.11.4 The TSLTally Sections
    - 4.11.4.1 Section [TSLTallyClient]
    - 4.11.4.2 Section [TSLTallyClient.ID]
    - 4.11.4.3 Section [TSLTallyClient.ID.InBound.SUBID]
    - 4.11.4.4 Section [TSLTallyClient.ID.OutBound.SUBID]
    - 4.11.4.5 Section [TSLTallyServer]
    - 4.11.4.6 Section [TSLTallyServer.InBound.ID]
    - 4.11.4.7 Section [TSLTallyServer.OutBound.ID]

4.12 The Service Sections

- 4.12.1 Section [service.ID]