Telnet protocol for the BTF1-series

revision 1.0

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, while string values are put inside quotations.

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

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, server will not return this, but use \x00 notation)

We recommand 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 syncronized, 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 proceesed, and parameters that are directly influenced by the command should already have been transmitted at this point.

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]
version="0.2.12"
model="BTF1-01"
serialnumber="ser99999"
sysName="Test Frame"
sysContact="Operator Foo"
sysLocation="The office"
[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 choosen for each login attempt). If the username was rejected, the server re-issue the user="".

```
[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`\xdcl\x
d5N\x07\x83\xdb"
```

At this step, the client should provide a correct hash2="" value. For blank passwords, the hash should be empty. Here is psuedo code in C# for generating the hash:

```csharp
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"
```

Sections

Section "hardware"

These sections contains information about how the hardware blocks used by input and output ports are constructed. The sections are named as following "hardware.ID" where ID is a numeric ID. Within these sections there currently only one value stored, the name. 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. Within these sections there will be two values. A name and a syntax specifier. This syntax specifier is documented in the appendix. Examples of the hardware sections:

```
[hardware.1]
name="None"
[hardware.2]
name="M2135x"
[hardware.2.parameter.0]
name="Rate"
syntax="wE;0=Auto;1=Powered Down;2=Bypassed;4=SD;8=HD;12=3G"
[hardware.8]
name="LMH0387"
[hardware.8.parameter.0]
name="Direction"
syntax="wE;0=Input;1=Output"
[hardware.8.parameter.1]
name="3G ext. reach"
syntax="wB"
[hardware.8.parameter.2]
name="Coarse amplitude"
syntax="wE;0=800mV p-p;1=400mV p-p"
```

Section "port"

These sections contains all the parameters bound to each hardware port in the BTF1 frame. Each section will be named "port.ID", where ID counts from 1. If a port is unidirectional, the affected input.* and output.* variables will be skipped. If the port is an SFP port, the SFP.* variables will be present

[port.4]	
name="SFP #4"	The actual name of the port. Read-only.
output.label="foo"	User-writeable label for the output part of the port.
output.source=-1	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.
output.extrc.type=2	See [hardware.2] to locate name and parameter specifiers for the output reclocker.
output.extrc.locked=0	Is this reclocker locked, if so this will be set to 1.

output.extrc.lockedat=0	What bitrate is this reclocker locked at (given in KHz)
output.extrc.parameters="0"	User-writeable parameters, lookup in [hardware.2] for syntax.
output.extrc.signaldetected=0	Is signal detected? 0=not able to detect, 1=detected, 2=not detected.
output.extcd.type=1	See [hardware.1] to locate name and parameter specifiers for the output cable driver.
output.extcd.parameters=""	User-writeable parameters, lookup in [hardware.1] for syntax.
output.extcd.signaldetected=0	Is signal detected? (0=not able to detect, 1=detected, 2=not detected).
output.autosource.mask="000000000000000000000000-000000000000000000000000-000000000000000000000000-000000000000000000000000"	User-writeable, auto-changeover mask, 4 groups of parameters. One digit per port.input 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 changover 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.
input.label="bar"	User-writeable label for the input part of the port.
input.exteq.type=1	See [hardware.1] to locate name and parameter specifiers for the input equalizer.
input.exteq.parameters=""	User-writeable parameters, lookup in [hardware.1] for syntax.
input.exteq.signaldetected=0	Is signal detected? 0=not able to detect, 1=detected, 2=not detected.

input.analyzer.enabled=2	User-writeable parameter. Is the SDI analyzer enabled for this input port? 1=Enabled, 2=Disabled.
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="none"	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=0	Is signal detected? 0=not able to detect, 1=detected, 2=not detected.
SFP.present=0	Is an SFP detected in the port? 0=No, 1=Yes
SFP.connector=""	What kind of connector does this SFP have. Most common is LC.
SFP.lengths="0;0;0;0;0"	Which cable lenghts is this SFP designed for? The 5 numbers represent single mode fiber, multi mode fiber OM1, OM2, OM3 and copper.
SFP.vendorname=""	The vendor of the SFP module.
SFP.partnumber=""	The part number of the SFP module.
SFP.serialnumber=""	The serial number of the SFP module.
SFP.productiondate=0	The production date of the SFP module (given in yyyymmdd format).
SFP.productionlot=""	The production lot of the SFP module.
SFP.revision=""	The revision of the SFP module.
SFP.wavelengthnm=0	The wavelength of the transmitter given in nm (for DWDM, the digits behind the decimal point are missing).
SFP.bitrate=""	This will contain 3 numbers separated with a ; 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.

Section SFP diagnostics

SFP modules can have multiple diagnostics entries dynamically allocated. These will be located in sections with the following name scheme: port.ID.SFP.diag.SUBID where ID port will reflect which port they belong to while SUBID is the induvidial diagnostics entries for the SFP. Each section will containt 3 properties. The name, value and the syntax. For description of the syntax, please see the appendix.
If a diagnostics-entry is removed (SFP module unplugged), a property named remove=1 will appear in the affected sections.

```
[port.8.SFP.diag.1529]
name="User selectable link speed"
value="1"
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"
```

```
[port.8.SFP.diag.1530]
name="Operating mode"
value="1"
syntax="wE;1=SGMII(default);2=1000BASE-X with copper auto-neg (GBIC);3=1000BASE-X without copper
auto-neg"
```

Section self-upgrade via Internet

The original method for doing firmware upgrade of the BTF1-x frame, was to trigger the unit to self-upgrade via Internet. For this to function, the frame needs to be connected to an IP network with Internet access and a valid DNS-server. All information needed to use this feature is inside the section named "firmware.aptengine".

```
[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
log.1="Health check #1"            This is the output from the init/update/upgrade process.
log.2="Health check #2"
log.3="Reading package lists..."
log.4="Building dependency tree..."
log.5="Reading state information..."
log.6="0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded."
log.7="Health check #3"
log.8="Software versions currently installed:"
log.9="barnone-apt       Software Version: BarnOne SubFunction APT                  0.0.1-1"
log.10="barnone-config     Software Version: BarnOne SubFunction Configuration        0.0.2-1"
log.11="barnone-firmware   Software Version: BarnOne SubFunction Firmware             0.2.96"
log.12="barnone-snmpd      Software Version: BarnOne SubFunction Communication        0.3.5"
log.13="barnone-watchdog   Software Version: BarnOne SubFunction Watchdog             0.0.2"
log.14="emnema             Software Version: BarnOne SubFunction Network Management   0.1.6"
log.15="Sync"
```

Sections for upgrade via file upload (and USB)

The other two methods of firmware upgrade is 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 or via other control protocols like openGear. To upload via Telnet, first set the size parameter to the expected file upload size. If successfull, you will see that size and busy both will be set. If only busy is set to non-zero value, then another client is uploading. You can cancel any upload (also other clients) by setting size=0.

When the client has the handle for uploading content (size and busy both have non-zero values), data can be uploaded by using a property named chunk. Recommended size for each packed of data is up to 1024 bytes.

[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 everytime 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 0 to 1, the clients should clear its local copy of the log file, since the server will start to count log lines from ID=0 again.
line.0="foo"	The actual log file
line.1="bar"	

Section "authdb"

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. In order to create a new user, simply open a authdb section with an unused ID number and fill in username, realname, salt1 and hash1. In order to change password for a user, set salt1 and hash1. Algorithm for hash1, please see the authentication section. In order to remove a user, set the remove property to 1. Regarding accesslevel, these are the definitions so far
0 = Disable
100 = Guest (read-only)
200 = panels (future use)
300 = administrator (write access)

```
 [authdb.5]
username="admin"
realname=""
salt1="f930d9fd8aa89c1bf1e17c6a83308533460a9120cb0a1f3f9c36cc63b26869a0"
accesslevel=300
remove=0
```

Section "diag"

Diagnostics about the frames health are given in sections named "diag.ID", one ID per diagnostic entry available. Each diagnostic entry have 4 properties. Name, a numeric value, indication if the diagnostics can cause the front warning LED to blink and a syntax description.

[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 of the value is outside the warning/error thresholds. To disable this diagnostics entry, set this value to 2.
syntax="rl;wmin=112;wmax=130;emin=110;emax=132;suffix=V;scale=0.100000"	For description of the syntax, please see the appendix

The network sections

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

Section "network"

The first section within the network namespace is just named "network" and currently only contains two read-only properties. The software version of the network handling software (named emnema internally) and the unique identifier of the frame which also happens to be its Ethernet MAC address.

```
[network]
version="0.1.6"
id="b8:27:eb:17:3e:e2"
```

Section "network.config"

The "network.config" section contains the current configuration file used be the network configuration software. If the client wants to change this configuration, it should first write to all of the properties it wants to change (or all of them), before finishing by setting commit=1

[network.config]	
ipv4.static.addr[0]="192.168.0.87"	Up to 4 static IPv4 adresses can be configured
ipv4.static.prefix[0]=24	This contains the prefix (also known as the netmask in IPv4. 24=255.255.255.0)
ipv4.static.addr[1]="0.0.0.0"	
ipv4.static.prefix[1]=0	
ipv4.static.addr[2]="0.0.0.0"	
ipv4.static.prefix[2]=0	
ipv4.static.addr[3]="0.0.0.0"	
ipv4.static.prefix[3]=0	
ipv4.static.gw="192.168.0.1"	The gateway if using static IPv4 adresses
ipv4.dns.server[0]="8.8.8.8	Up to 3 DNS servers can be given when using static IPv4 adresses
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 using static IPv4 adresses
ipv4.dns.search[1]=""	
ipv4.dns.search[2]=""	
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 adresses can be configured

ipv6.static.prefix[0]=64	And this contains the prefix for the static IPv6 adresses
ipv6.static.addr[1]="2002::2123"	
ipv6.static.prefix[1]=64	
ipv6.static.addr[2]="::"	
ipv6.static.prefix[2]=0	
ipv6.static.addr[3]="::"	
ipv6.static.prefix[3]=0	
ipv6.static.gw="::"	The gateway if using static IPv6 adresses
ipv6.dns.server[0]="::"	Up to 3 DNS servers can be given when susing tatic IPv6 adresses
ipv6.dns.server[1]="::"	
ipv6.dns.server[2]="::"	
ipv6.dns.search[0]=""	Up to 4 DNS searches can be given when using static IPv6 adresses
ipv6.dns.search[1]=""	
ipv6.dns.search[2]=""	
ipv6.dns.search[3]=""	
ipv6.mode=10	Operation mode for IPv6. 8=LinkLocal only, 9=DHCP/Statefull, 10=Static,12=StateLess/Router Advertisment
commit=0	Set to 1 in order to commit the configuration

Section "network.status.ip"

The current active IP adresses assigned to the frame are readable by the "network.status.ip.ID" sections. Non-used entries will have prefix=0

```
[network.status.ip.25]
dev=""
addr=""
prefix=0
[network.status.ip.26]
dev="eth0"
addr="2002::2123"
prefix=64
[network.status.ip.27]
dev="eth0"
addr="fe80::ba27:ebff:fe17:3ee2"
prefix=64
[network.status.ip.28]
dev="eth0"
addr="2001::2"
prefix=64
[network.status.ip.29]
dev="eth0"
addr="192.168.0.87"
prefix=24
```

Section "network.status.route"

The current routing table assigned to the frame are readable by the "network.status.route.ID" sections. Non-used entries will have dstprefix=0

```
[network.status.route.60]
dev="eth0"
dst="0.0.0.0"
dstprefix=0
gw="192.168.0.1"
[network.status.route.61]
dev="eth0"
dst="192.168.0.0"
dstprefix=24
gw=""
[network.status.route.62]
dev="eth0"
dst="fe80::"
dstprefix=64
gw=""
```

Section "network.status"

This current currently only contains only one property, the current active DNS configuration file (known as resolv.conf in unix systems)

```
[network.status]
resolvconf="nameserver 8.8.8.8\n"
```

Section "network.log"

This section contains the log from the networking softwares involved. The network configuration server only has a finate number of entries, so as the log files grows, it will start to forget the oldest entries, causing the log no longer counting from 1. If the networking software is restarted while connecting to the telnet server, it will start to make new log, counting from 1 again.

Each entry in the log contains of three parts, splitted by a ; character. The first field is the source of this entry, the second is the channel (stdout or stderr) and the last field is the actual log/data output.

```
[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"
```

Section "logo"

If the frame features the LED illuminated logo on the front left, this section will appear. It will contains two properties: brightness and findme. Brightness is an numeric value ranging from 0 to 100, specifying the intensity of the LED. Findme property will make the front LED blink the n amount of seconds, making it easier to indentify the frame. To disable the blinking, set this value to zero. Maximum allowed value is 86400 (24 hours).

```
[logo]
brightness=100
findme=0
```

APPENDIX A: Syntax specifier

Various parts of the server software specifies a syntax string, specifying how a parameter is to be interpreted. A syntax specifier can consist of several components/properties, were each one will be separated by a ; character.

The first property will always be the data type and protection. The first character is either a r or w meaning read-only or writeable. The rest of the characters give the data-type:

B for boolean (0 for unset, 1 for true, 2 for false). A writable boolean will normally fail if you try to set it 0

I for integer. This might be followed by options to give it scaling, warnings and error limits and suffix:
I;scale=0.1;wmin=10;wmax=100;emin=0;emax=110;min=-100;max=200;offset=10;suffix=mW

E for enumeration. The properties following this will be the options an can look like this:
1=option1;2=option2

S for string. Possible properties are len=x to limit the length of the string. The redundancy switch has one special rule, setting autosource-mask-v1. This string is expected to be 4 series of digits split by a - sign. Each digit matches a valid input port from the *port* that has input properties. The first group is the “Enabled” column (0 = not used, 1 = main and 2= backup), the second group “Sensitive to LOS” column (0 = not set and 1 = set), the third group column “Sensitive to analyzer lock” and the fourth group “Sensitive to analyzer errors” column.

Future data-types:

IPv4 for an IPv4 address

IPv4Net for an IPv4 address with prefix (192.168.0.255/24)

IPv6 for an IPv6 address.

IPv6Net for an IPv6 address with prefix.

IP for an IPv4 or IPv6 address.

IPNet for an IPv4 or IPv6 address.