UM2004
User manual

STreamPlug ST2100 SDK and quick start guide

Introduction

This manual explains the steps necessary to install the STreamPlug SDK, and how to build and load a firmware image for a STreamPlug based device. It will also cover creating a new configuration to support different hardware configurations.

It is assumed the reader has a thorough knowledge of Linux.

For updates and support please consult the IoTecha Web site: www.iotecha.com.

Contents

1 SDK setup

1.1 Release package

The full release package contains the SDK, STreamPlug firmware, Linux source, as well as prebuilt images to get you started quickly. Partial releases may contain updates on any released component such as firmware, or an update to the Linux kernel. However, this document relates to a full release package.

After extracting the release archive, the following files/folders can be found:

```
stp_sdk-{build_date}.tar.bz2
firmware/stp-bridge-release-{build_id}
firmware/stp-lnx-release-{build_id}

linux/prebuilt/backport-prebuilt.tgz
linux/prebuilt/tools-prebuilt.tgz
linux/prebuilt/vmlinux
linux/prebuilt/rootfs/full_nand
linux/prebuilt/rootfs/full_smi
linux/prebuilt/rootfs/minimal_nand
linux/prebuilt/rootfs/minimal_smi

linux/src/backports.tar.bz2
linux/src/buildroot.tar.bz2
linux/src/examples.tar.bz2
linux/src/oklinux-2.6.35-streamplug.tar.bz2

prebuilt/rel-stp-{ver}-bridge-release-{build_id}/
(.stpupg/.smiprog/.fw.bin)
prebuilt/rel-stp-{ver}-stplnx-release-{build_id}/
(.stpupg/.smiprog/.fw.bin)
prebuilt/rel-stp-{ver}-bridge-release-{build_id}-FACTORY/{board}/
(.smiprog/.fw.bin)
```

The files are grouped and described as follows:

- **SDK**
  - stp_sdk-{build_date}.tar.bz2
    - This SDK file contains the tools and scripts to generate and assemble firmware images. The SDK also contains some example 'board' configurations. Board configurations are discussed in more details in *Section 4: Hardware target (board) configuration on page 14*. The build_date uniquely identifies a SDK bundle. Always use the latest SDK.

- **STreamPlug firmware**
  - firmware/stp-bridge-release-{build_id}
    - One of the two firmware packages supported by the STreamPlug. This bridge firmware is for PLC Ethernet bridging (including all management support such as firmware update, etc.) and does not enable Linux. Due to its small size, the bridge firmware can be useful as a 'backup' firmware image for the Linux system if the Flash space is limited.
  - firmware/stp-lnx-release-{build_id}
    - This is the second firmware package for the STreamPlug. This firmware includes the hypervisor support to run a Linux kernel that can be built from the provided sources.
- **Linux**
  - Prebuilt binaries
    - linux/prebuilt/backport-prebuilt.tgz
      - Prebuilt backport for Wi-Fi drivers. For more information on Linux see the UM2003 - “Getting started guide - STreamPlug ST2100 Linux support package”/ UM1942 - “Linux software user manual for STreamPlug ST2100” on www.st.com.
    - linux/prebuilt/tools-prebuilt.tgz
      - Prebuilt STreamPlug tool binaries. This contains Linux tools that can be used to manage the STreamPlug modem from Linux environment, including:
        - stpnetutil - firmware update, discovery, device stat. access, etc.
        - stpconsole - allows access to the modem management console from within Linux
        - stpimagevalidate - used to indicate to the system that the boot was successful. This tool is used to support a fault-tolerant remote update.
    - linux/prebuilt/vmlinux
      - Linux kernel prebuilt from included sources
    - Root file systems
      - Prebuilt root file systems with different options. These correspond to the config. files found in the buildroot sources. See Linux manuals UM1942 and UM2003 for more information.
      - Included configurations
        - linux/prebuilt/rootfs/full_nand
        - linux/prebuilt/rootfs/full_smi
        - linux/prebuilt/rootfs/minimal_nand
        - linux/prebuilt/rootfs/minimal_smi
  - Linux sources
    - Source archives for the above-mentioned prebuilt binaries
    - Archives
      - linux/src/backports.tar.bz2
      - linux/src/buildroot.tar.bz2
      - linux/src/examples.tar.bz2 (contains tools)
      - linux/src/oklinux-2.6.35-streamplug.tar.bz2

- **Prebuilt binary images suitable to load onto hardware**
  - prebuilt/rel-stp-{ver}-bridge-release-{build_id}/ (.stpupg/.smiprog/.fw.bin)
  - prebuilt/rel-stp-{ver}-stplnx-release-{build_id}/ (.stpupg/.smiprog/.fw.bin)
  - prebuilt/rel-stp-{ver}-bridge-release-{build_id}-FACTORY/{board}/ (.smiprog/.fw.bin).

1.2 Installing SDK

The first step is to install the SDK in your environment. To do this, extract the SDK to its own folder. For this example, make a directory named stp_sdk at the same level as the archive and extract to that location:

```
$ mkdir stp_sdk
$ cd stp_sdk
$ tar xfj ../stp_sdk-{build_date}.tar.bz2
```

Note: The SDK must be installed on a Linux PC in order to build and assemble images.

To use the SDK, some environment variables and paths must be setup. The below example assumes using a bash shell.

```
$ export STP_SDK_DIR={path to stp_sdk dir made above}
$ export
PATH=${STP_SDK_DIR}/system/scripts:${STP_SDK_DIR}/system/tools/bin/linux-gnu-x86_64:$PATH
```

After the SDK is setup you can test by running the utility 'stpmake'. It should run and report errors as required command line arguments are not present. If the system cannot find the stpmake utility, verify the above steps.

1.3 Windows® tools

Although the SDK must be run on Linux to build images, several of the tools are available to run on a Windows PC. Windows tools are located in the SDK under: ${STP_SDK_DIR}/system/tools/bin/cygwin-i686.

Note: Cygwin is NOT required. In the future this folder may be renamed “win32”.

2 Firmware releases

STreamPlug firmware images can be built from either of the supplied firmware releases. As enumerated above (*Section 1.1*) in the files contained in the SDK release package, two base firmware images are released with the SDK:

- stp-bridge-release
  - Bridge firmware is a complete firmware package that enables PLC Ethernet bridging.

    Tip: stp-bridge-release can be used as a good initial step in your application development or debug. With the bridge mode, you can use a desktop Linux PC to develop and debug your application and test with bridge mode firmware connecting your PC to STP via Ethernet. Later, you can make transition to the stp-lnx-release image to run your application in the embedded Linux environment.
- stp-lnx-release
  - Linux firmware contains the PLC MAC as well as a microvisor that will run a Linux kernel.
  - This is NOT a complete image and requires a Linux kernel to be 'woven' into it in order to build a bootable image. A pre-compiled Linux kernel is provided to get started quickly without the need to compile a kernel.

When using SDK tools with the stp-lnx-release package, the tool will also require you to specify the path to a Linux kernel (vmlinux) to use.

Note: Each firmware build contains a unique build ID that is a part of its filename.

3 Firmware regions and boot sequence

STreamPlug firmware images can be placed into two 'regions' defined by the memory map, these are the:

- Factory region
- Upgrade region

Images to be loaded into the factory region are referred to as factory images, and when generated by the tools have the string FACTORY in the name, while images to be loaded into the upgrade region have the string UPGRADE in the resultant image filename.

Factory and upgrade images are different. While they can each contain the same firmware (within size restrictions based on the region sizes), the factory image contains additional information that is unique to the hardware platform.

Factory images contain:

- Bootloader
- DDR memory parameters
- Memory map [specifies where to look for an upgrade image (if it is present)]
- Manufacturer parameters [hardware specific configuration parameters (more on this later)]
- Firmware

Upgrade images contain only the bootloader and firmware.

3.1 Boot sequence

In order to help understand the different images, a few example cases of boot sequence are examined in the following sections.

Example boot scenarios

Case 1: Flash contains a factory image whose memmap does not specify an upgrade image (see the memmap definition detail in *Section 4.3.2: Memory map on page 16*).

- The STreamPlug starts to load the factory image from a location based on the BMODE pin value: (see *Appendix D: STreamPlug boot source selection on page 53* for more details)
  - If SMI: address 0x0 of SMI CS1
  - If NAND: starts looking at the address 0x0 for Image #0 in NAND
- Loads DDR, initializes memory
- Loads and stores the memory map for use by later boot stages
- Loads and stores manufacturer parameters (some of these parameters can specify optimized timing for SMI/NAND interfaces; if so they are applied)
- Loads and executes the factory firmware image.

Case 2: Flash contains a factory image whose memmap does specify an upgrade image, but this region is empty or corrupt (see the memmap definition detail in *Section 4.3.2: Memory map on page 16*).

- The STreamPlug starts to load the factory image from a location based on the BMODE pin value (see *Appendix D: STreamPlug boot source selection on page 53* for more details):
  - If SMI: address 0x0 of SMI CS1
  - If NAND: starts looking at address 0x0 for Image #0 in NAND
- Loads DDR, initializes memory
- Loads and stores the memory map for use by later boot stages
- Loads and stores manufacturer parameters
  - Some of these parameters can specify optimized timing for SMI/NAND interfaces; if so they are applied.
- Since the upgrade image location was specified, it starts to boot the firmware at the upgrade region.
- Since there is no valid upgrade region, the load fails, and it continues to boot the factory firmware.

Case 3: Flash contains a factory image whose memmap does specify an upgrade image, and the upgrade image is present (see the memmap definition detail in *Section 4.3.2: Memory map on page 16*).

- The STreamPlug starts to load the factory image from a location based on the BMODE pin value (see *Appendix D* for more details):
  - If SMI: address 0x0 of SMI CS1
  - If NAND: starts looking at address 0x0 for Image #0 in NAND
- Loads DDR, initializes memory
- Loads and stores the memory map for use by later boot stages
- Loads and stores manufacturer parameters
  - Some of these parameters can specify optimized timing for SMI/NAND interfaces; if so they are applied.
- Since the upgrade image location was specified, it starts to boot the firmware at the upgrade region.
- Upgrade image is loaded and running.

Case 4: There is no valid factory image found.

- If the STreamPlug cannot find a valid factory image to start booting based on the BMODE (see *Appendix D* for more details) configuration pins, it will enter a boot recovery mode where it will try to boot via the serial port. See *Section 3.4: Boot recovery mode* for more details on the boot recovery mode.

3.2 Firmware integrity

The firmware images are well protected with incremental 64-bit CRCs that are checked as the image is loaded by the bootloader making it highly unlikely for a 'bad' image to be executed.

Network update

The network update is robust against failure due to:

- Firmware integrity checks (CRCs) - in the image will prevent a bad image from being executed.
- A factory firmware image that has the capability to perform a network based update is present and will never be changed.
- In field updates will only write to the upgrade region.

Of course, in development, writing the factory firmware is allowed and necessary. However, writing the factory image from the network update tool is dangerous. If the process fails or is interrupted, the device will need to be recovered using the serial loader mechanism described in *Section 3.4*.

3.3 Upgrade image rationale

The driving force behind storing device unique data in the factory region (DDR parameters, memmap, manufacturer configuration parameters) is to decouple the upgrade firmware image from a specific hardware build and/or revision of hardware.

This method allows a common upgrade image to be used in many products. With this method, it is also possible to have one upgrade binary that could be deployed to several different products, thereby reducing testing time and user confusion when selecting the correct firmware to load.

3.4 Boot recovery mode

If the device fails to boot a valid firmware image (this could be due to some update failure, or it could be the case of a new device that has an empty Flash) it will enter a boot recovery mode.

In the boot recovery mode, the device will alternate listening on both UARTs (checking each set of IOs since each UART can be selected on two different pin locations) for a firmware loader beacon.

There is a utility as a part of the SDK (sfwloader) that is available for both Linux and Windows PCs that implements the serial firmware loader protocol for use with this recovery mode.

It is possible via the SDK to generate an image for use with the sfwloader that will program SMI (or NAND) and this can be used to recover a 'bricked' (unresponsive due to bad programming) board.

Note: The SDK will generate files named .smiprog or .nandprog to indicate which device they will program. Since the STreamPlug does not contain any code that can program the Flash in its small internal boot ROM, these images will load a small programmer first, then do the programming (all as a part of a single image). Since the DDR memory must be accessed to execute the programming algorithm, the smiprog/nandprog images have a dependency on the DDR memory type/configuration. The SDK ships with generic configurations for the different DDR configurations known to be used by STreamPlug modules. When generating these images from the SDK it will require a board configuration to be specified due to this DDR memory dependency.

Manually entering boot recovery mode

Sometimes it may be necessary or convenient to enter the boot recovery mode manually even if there is a bootable image on the device. Two examples of this would be:

1. Programming a device via serial when there is no network connection easily available.
2. A firmware image was loaded that boot successfully but when executed enters some deadlock before watchdogs could be enabled.

In this section the various methods of manually entering the boot recovery mode are discussed.

1. Hardware pushbutton

a) Some module designs include a small pushbutton that will force the device into the boot recovery mode. If your hardware platform has this feature; this is the easiest method. To use this feature, follow these steps:

i. Power off the device (or hold in reset button)

ii. Press and hold the recovery button

iii. Apply power (or release reset button)

iv. Release the recovery button

2. **From bootloader**

a) If your image is configured such that the bootloaders (both in the factory and upgrade images) output boot messages, you can use this method. (To determine if the bootloaders are outputting debug messages; connect a console and you should observe output immediate out of reset).

i. During the image loading process (where the '.' characters are displayed) quickly press the '!' key 5 times. This will cause the boot to abort.

1) If the upgrade image is booting, aborting the load of the upgrade image by pressing '!' will cause the factory image to load immediately following the upgrade image load failure, so you must continue to press '!' quickly to abort the load of the factory image. When all console output stops with an error message, the device has entered the boot recovery mode.

3. **From the RTOS console**

a) If the RTOS console is enabled on a serial port, (or accessible via the stpconsole application from within Linux) you can reboot to the recovery mode.

i. When you access the RTOS console, make sure you are at the top menu by pressing 'q' several times.

ii. Press 'R', then 'S'

```
 [target:okl-linux] ##>R
!!!!! Are you sure you want to RESET the board???
(CAPITAL 'Y' if you are,
 CAPITAL 'S' to reboot to the standard region.)
The system is going down NOW!
```

iii. This will cause the device to boot to the previous image in the boot chain that would be:

1) If running the upgrade, it would boot the factory image. So if running the upgrade, you will need to do this twice to get to the boot recovery mode.

Note: If the FACTORY image is running bridge firmware, the stpconsole will not be available. In this case use the RTOS serial console or another method to exit factory firmware into boot recovery mode.

2) If running the factory, since there is no previous image, it would enter the boot recovery mode.

4. Via hardware intervention

Warning: RISK OF DAMAGING HARDWARE.
Only use this method as a last resort and at your own risk.

---

a) As can be inferred from the above methods, the recovery mode is entered when a boot attempt fails. (In fact, the recovery push-button works by pulling one of the BMODE inputs high, so that an invalid boot method is selected, with the result that the boot attempt will fail).

b) As a last resort, the boot recovery mode can be entered by forcing the boot to fail, perhaps by:

i. Changing the BMODE pins (if possible with the dip switch or jumpers on a development board)

ii. Inserting some error from the SMI read if booting from SMI by disconnecting CS, or temporally shorting the DOUT of the SMI part.

4 Hardware target (board) configuration

4.1 Introduction

The SDK allows for custom configurations to support different hardware builds. In the SDK these configuration sets are referred to as 'boards'. Board configurations include:

- DDR memory configuration (size, organization, and timing values)
- Multi-purpose IO (MFIO) mappings
- Peripheral configurations (via Linux command line)
  Note: The 'Linux' command line is also processed by modem firmware, and therefore is required for the non-linux 'bridge' images as well.
- Flash memory map
- HPAV modem configuration

Some generic board configurations are shipped with the SDK and can be found in the folder: `$STP_SDK_DIR/board`. In addition to board configurations provided in the SDK, hardware partners may also provide example configurations for development boards and STreamPlug modules.

4.2 User board configurations

To maintain a clean SDK installation, it is recommend that 3rd party and user-created board configurations are managed outside the SDK directories. This will enable easier local version control for these files. To support this, set the environment variable STP_SDK_USER_BOARDS to point to a folder containing board configurations.

The stpmake utility uses the board configurations to assemble firmware images. You can also use stpmake to see what boards are available.

```
$ export STP_SDK_CUSTOM_BOARDS=/work/streamplug/custom_boards
$ stpmake --list-boards

stpmake v1.11
From: /work/streamplug/custom_boards
      test_board : My Custom board
From: /work/streamplug/stp_sdk/board
   generic-128MB-smi : Std Config, 128MB Ddr, SMI memmap, eth=off
   generic-256MB-smi : Std Config, 256MB Ddr, SMI memmap, eth=off
   generic-512MB-smi : Std Config, 512MB Ddr, SMI memmap, eth=off
```

In the above example you can see a custom board named 'test_board' is located in: /work/streamplug/custom_boards that is the custom board folder.

4.3 Board configuration detail

Each board configuration is a folder that contains a set of configuration files. An overview of these configuration files is presented below.

4.3.1 Board configuration - board.cfg

The board.cfg file is the top-level configuration file for the board. It is a parameter=value pair file with some mandatory and some optional fields as described in *Table 1*.

Table 1. Configuration parameters (board.cfg)

Parameter	Required	Description
description	N	A textual description of the board configuration to be displayed when using stpmake --list-boards.
ddr_config	Y	Specifies the DDR memory configuration. Several 'standard' configurations are available under: $STP_SDK_DIR/system/ddrconfig – DDR2_1x_x16_128MB - 128 MB (one x16 device) – DDR2_1x_x16_256MB - 256 MB (one x16 device) – DDR2_2x_x8_256MB - 256 MB (two x8 devices) – DDR2_2x_x8_512MB - 512 MB (two x8 devices) Contact ST for support if you require a different configuration not provided by default.
memmap	Y	Specifies the file to be used as the system memory map. See *Section 4.3.2* for more details on the format of the memmap file.
bootdbg	N	Defines what UART is used to display low level boot status messages. This is useful for early debug, and is advisable to set if a serial console is available. Allowed values are: – uart1g6 - UART1, with IOs mapped to 'primary' config. on MFIOS 24-27 – uart1g20 - UART1, with IOs mapped to 'secondary' config. on MFIOS 80-83 – uart2g5 - UART2, with IOs mapped to 'primary' config. on MFIOs 20-23 – uart2g21 - UART2, with IOs mapped to 'secondary' config. on MFIOs 84-87 – release - boot debug messages disabled (default value).
mfct_file	N	Specifies the file to be used for the manufacturer parameters. See *Section 4.3.3* for more details on the format of the mfct file.
hpav_config	N	Specifies the file to be used for the hpav modem configuration file. See *Section 4.3.4* for details on the format of the hapvconfig file.

Note: Some parameters specify a file location. The path of the file is determined by the leading characters of the filename as defined in Table 2.

Table 2. File location spec. (board.cfg)

Example	File location
memmap=my.map	Within SDK (see comments in “board.cfg” files included in the SDK for details).
memmap=/abspath/my.map	Absolute location as specified
memmap=./my.map	Within the current board directory

4.3.2 Memory map

The memory map specifies the layout of the Flash for the system. In addition to specifying where the firmware images are, it specifies the location of the STreamPlug firmware (RTOS) NVRAM area, as well as Linux partitions. The Linux partition information is automatically generated from the memmap.

File format

Comments in the memory map are in the 'shell' style, where a leading '#' for a line denotes a comment.

The file has one entry per line, and the entry is described as:

```
# Comment
Entry
Entry
```

```
Entry            := name:type:type:fwupdate_allowed[:device_variant]
name             := alphanumeric
type             := sys_factory_fw | sys_upgrade_fw | sys_nvram | lnx_part
| lnx_part_ro
device           := smi1 | smi2 | smi3 | nand
fwupdate_allowed := y | n

Where device == smi1 | smi2 | smi3
  device_variant := start_address:size
Where device == nand
  device_variant := start_address:size:fw_image_number
start_address    := number
size             := number
number           := 0xnnnn | nnn[K|M]
```

Example 1 SMI memory map

```
factory :sys_factory_fw :smi1:n:0x00000000:0x00100000
upgrade :sys_upgrade_fw :smi1:y:0x00100000:0x004c0000
nvram   :sys_nvram      :smi1:n:0x005c0000:0x00040000
rootfs  :lnx_part       :smi1:n:0x00600000:0x00a00000
```

In this memory map it is assumed that there is a 16 MB SMI part on the first chip select, and the STreamPlug is configured to boot via SMI (setting BMODE to 0b000 - see *STreamPlug boot source selection on page 53* for more details). It is important that the factory image section matches the boot type. Only the upgrade firmware is allowed to be updated in the field by the network update utility.

Example 2 NAND memory map

```
factory :sys_factory_fw :nand:n:0x00000000:0x00600000:0
upgrade :sys_upgrade_fw :nand:y:0x00600000:0x00600000:1
nvram   :sys_nvram      :nand:n:0x00C00000:0x00040000
rootfs  :lnx_part       :nand:n:0x00C40000:0x1F3C0000
```

In this memory map, it is assumed that there is no SMI, and all firmware is to be in the NAND. This is similar to the above SMI example, except the fw_image_number defined for the NAND device type. The fw_image_number is required for NAND firmware and fixed as 0==factory firmware, 1==upgrade_firmware.

Special considerations

- For Linux partitions, the name (first field) of the memory map entry has significance.
  - If the type is rootfs, it will be used as the root partition from the command line.
  - File systems can be written using the network update tool, stpnetutil. For this to function, the file partition name must be one of:
    - rootfs
    - aux1fs
    - aux2fs
- 'nvram ' type must be included in the memory map, and should be at least 2 x the device erase block size.

The flag 'fwupdate_allowed' is the permission after the device parameters are locked down. In the development phase without locking device parameters down, all regions can be updated using the stpnetutil regardless of the value of this field. See “**mfct.writeprot**” in *Section : MFCT keys* for more details on lockdown.

4.3.3 Manufacturer parameters and the mfct.xml configuration file

Manufacturer parameters can be thought of as a set of key/value pairs that define 'build time' configuration for a device.

The mfct.xml configuration file is where the user specified manufacturer parameters are stored. The mfct.xml should be modeled after the template in the SDK as the SDK tools will post process this file and the format must lend itself to such processing.

Warning: **WARNING: RISK OF INCORRECT PROGRAMMING.**
This file directly specifies binary data used by the system. There are very few format checks on this data, and errors in parameter sizes and field locations could cause to a device a fail to boot. When making changes to this file double check your work and test on one device before deploying the update to many devices.

The valid keys and format of their values are defined within the SDK in a parameter definition file located at: ${STP_SDK_DIR}/system/mfctparam/params.def. This file specifies the key name as well as the data structures for the data.

An example of this is the definition for one of the required parameters hpav.core.

```
# Hpav core, required parameters
hpav.core : Hpav
{
    # Version of this param. Must be 0.
    Version:u8
```

```
    # HPAV Mac address to be used by the modem
    MacAddr:byte[6]
   # Default NMK to be used by HPAV. This is the 'reset' value.  The device
will store an
    # updated version in NVRAM based on user input.  If the device is reset
to factory
    # defaults, this value is used.
    Nmk:byte[16]
    # Device encryption key.  Unique per-device key - not changeable by
user.
    Dak:byte[16]
    # Default Device friendly name
    DeviceName:string[*32]
}
```

Note that MAC addresses are a part of this config set. Since the manufacturer parameters are stored in the factory region, they are not updated when a network update is done. This allows device specific configuration values to be set.

For MAC addressees specifically, if a special value of ff:ff:ff:ff:ff:ff is used, the STreamPlug firmware will generate a unique 'locally managed' MAC address based on that device's unique silicon ID. This can help to avoid the burden of assigning MAC addresses during the development cycle.

Updating MFCT parameters

There is a tool that can be used at production time to update these parameters via some special messages; this gives an easy way to program per device information after all production tests have passed.

Some of the parameters that would be updated at manufacture time would include per device unique information such as:

- MAC address
- Security keys
- Certificates

In addition to programming per device unique information, the write protection key would be added (see “**mfct.writeprot**” in *Section : MFCT keys*) before the changes were committed. At this point the device would no longer be able to accept parameter updates, and the 'fwupdate_allowd' permissions set in the memmap will now be honoured.

MFCT keys

hpav.core

This parameter is used by the HomePlugAV modem; contains necessary configuration data.

Table 3. hpav.core

Key	Field	Type	Len.	Description
hpav.core	HPAV core parameters. The hpav.core section is mandatory.			
	Version	ubyte		Version of this parameter. Must be set to 0.
	MacAddr	ubyte	6	HPAV Mac address to be used by the modem. If set to “ffffffffffff”, a valid MAC address will be automatically generated and set during the firmware load.
	Nmk	ubyte	16	Default NMK to be used by HPAV. This is the 'reset' value. The device will store an updated version in the NVRAM based on user input. If the device is reset to factory defaults, this value will be used.
	Dak	ubyte	16	Device encryption key. Unique per device key. This value cannot be changed by the user.
	DeviceName	string	Up to 32	Default device user friendly name.

Note: The NMK and DAK specified in the manufacture parameter region are keys, not passwords. Passwords are exposed to the user, while keys are used internally. See Appendix E: Hashing passwords to keys on page 54 for more details on creating keys from passwords for use in the manufacturer parameter region. To avoid confusion, keep in mind that the stpnetutil also uses a command line option “--nmk” that refers to the password (see Appendix B: stpnetutil on page 34 for details).

virtif

This parameter is used to configure the virtual network interface between the Linux and HPAV modem.

Table 4. virtif

Key	Field	Type	Len.	Description
virtif	Virtual network interface between Linux and STP firmware.			
	Version	ubyte	1	Version of this parameter. It must be set to 0.
	MacAddr	ubyte	6	MAC address to be used by default. This value will be the one displayed on the Linux console when using the ifconfig command. If modified by user, the driver will override it.

ethernet

This parameter is used to configure the Ethernet MAC address.

Table 5. ethernet

Key	Field	Type	Len.	Description
ethernet	Ethernet configuration			
	Version	ubyte	1	Version of this parameter. It must be set to 0.
	MacAddr	ubyte	6	MAC address of the Ethernet interface. This is not needed in the bridge mode since the interface will run in the promiscuous mode.

target.smiconfig

This parameter is used to configure the timing parameters for the SMI memory. See the “param.def” file in the SDK: `${STP_SDK_DIR}/system/mfctparam/params.def` for details.

Table 6. target.smiconfig

Key	Field	Type	Len.	Description
target.smiconfig	Timing parameters for the SMI memory. This parameter will be parsed by the boot opcode that reads mfct params so that boot speed can be maximized.			
	Version	ubyte	1	Version of this parameter. It must be set to 0.
	BankMask	ubyte	1	Bank enable mask. Setting bits in this byte indicates what banks are enabled (LSB=CS0). *Note: Because the STreamPlug has dedicated IO for each bank there is no harm to enable all banks.*
	TwoByteAddrMask	ubyte	1	TwoByteAddress mode mask. LSB=CS0. Setting bits in this byte instructs the controller to generate two byte address cycles (for EEPROM).
	ClkHoldPeriod	ubyte	1	Setting ClkHold to non-zero stops the clk between each byte, while CS remains active.
	FastMode	ubyte	1	Setting to 1 enables the fast read by selecting the read opcode 0x03/0x0b -- 0: normal read: 0x03, address, reception -- 1: fast read: 0x0b, address, dummy Byte, reception -->
	Prescaler	ubyte	1	Prescaler: 0-127. Sets SMI_CK frequency to SMI_CK = 166 MHz / prescaler. *Note: For prescaler = 0, SMI_CK = 166 Mhz.*
	DeselectTime	ubyte	1	When CS is deselected it remains deselected for at least (DeselectTime + 1) SMI_CK clock periods.

target.nandconfig

This parameter is used to configure the timing parameters for the NAND memory. See the “param.def” file in the SDK for details.

Table 7. target.nandconfig

Key	Field	Type	Len.	Description
target.nandconfig	Timing parameters for the NAND memory. This parameter will be parsed by the boot opcode that reads mfct params so that boot speed can be maximized.			
	Version	ubyte	1	Version of this parameter. It must be set to 0.
	Tset	ubyte	1	Time from address valid to /OE /EW activation. Total time is Tclk × (Tset + 1).
	Twait	ubyte	1	Time from enable on to enable off for all signals: /OE /EW. Total time is Tclk × (Twait + 1) *Note: Min. value is 1.*
	Thold	ubyte	1	Time from enable off (/OE, /EW) and the end of the cycle: address/data going to invalid. Total time is Tclk × Thold *Note: Min. value is 1.* *Note: Tperiod = Tset + Twait + Thold.*
	Thiz	ubyte	1	Time from address valid to data bus being driven. (Write cycle only). *Note: Total time is Tclk × Thiz.*

target.cmdline

This parameter is used to set the Linux command line.

Table 8. target.cmdline

Key	Field	Type	Len.	Description
target.cmdline	Command line			
	Version	ubyte	1	
	Cmdline	string	Up to 1024	System command line (used in both Linux and non-Linux targets).

An example of a command line configuration is:

```
<xmldata:string>console=none rtosconsole=uart1,115200n81
ubi.mtd=@@CMDLINE_BOOTPART_rootfs@@ root=ubi0_0 rootfstype=ubifs clcd=off
sata=off pcie=off usb=on:host eth=on:secondary i2c=on ssp=on:39
uart1=rtos:primary uart2=on:primary can=on:secondary firda=off fsmc=off
sport=off ts=off ark_gpio=on:112211 arm_gpio1=on arm_gpio2=on
@@CMDLINE_PARTINFO@@</xmldata:string>
```

Note the @@XXX@@ items above. These must remain “as is”, as they are updated by the SDK tools with information from the memory map.

target.info

This parameter specifies hardware information such as the product name/revision. These values are exposed to the user via the device discovery tool.

Table 9. target.info

Key	Field	Type	Len.	Description
target.info	Target specific information: information that can be set at manufacture time and used by running SW.			
	Version	ubyte	1	Version of this parameter. Must be set to 0.
	DeviceType	ubyte	1	Device type reported by discovery: – 0x00 - generic STP device – 0x01 - 0x7f - user defined – 0x80 - 0xfe - Reserved – 0xff - factory tester V1.
	HardwareName	string	32	Name of the HW platform, like the PCB name, etc.
	HardwareRevision	string	8	Revision string

mfct.writeprot

Writing this key enables write protection on the manufacturer parameter region and causes the firmware update to honour the permissions set in the memory map.

Table 10. mfct.writeprot

Key	Field	Type	Len.	Description
mfct. writeprot	If this key exists, MFCT param write is disabled. Recommended for the production device.			
	Version	ubyte	1	Version of this parameter. Must be set to 0.

target.memmap

Used internally. This is generated by the SDK by parsing the memmap file. Do not set this value.

target.recovery

This parameter is used to control the recovery mechanism.

Table 11. target.recovery

Key	Field	Type	Len.	Description
target. recovery	Version	ubyte	1	Version of this parameter. Must be set to 0.
	SetImageGoodTimeout	u16-le	2	Number of seconds to wait after the system boot for a set image good signal from Linux. If timeout expires, the system will reboot to factory firmware. If 0, no timeout is used.

4.3.4 HPAV modem configuration

The modem configuration file is an XML file that controls the operation mode of the PLC modem.

The file should be a well formed XML file based on the schemas located in the SDK under $STP_SDK_DIR/system/tools/schema.

Below is an example configuration file enabling the HPGP mode for use as an EVSE, and also specifying the polarity of the line driver used.

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="hpavconfig" xmlns:axd="arkxmldata" xmlns:t="hpavconfig_types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="hpavconfig ../hpavconfig.xsd">
  <config>
    <t:txon-polarity>active_high</t:txon-polarity>
    <t:device-profile>gp_evse</t:device-profile>
  </config>
</root>
```

All of the available parameters are documented in the schema hpavconfig_types.xsd found in the SDK.

Warning: RISK OF INCORRECT PROGRAMMING.
Changing any of these values can lead to the device not operating properly. Please contact ST to verify changes made to the modem configuration.

5 Assembling firmware binaries

To assemble upgrade firmware binaries for use with the network update tool, you will need:

- SDK
- STreamPlug firmware
- Linux kernel (if building a Linux image)

To assemble factory firmware binaries, or any firmware binaries to be used with the serial loader, you will also need (see *Section 3.4: Boot recovery mode on page 11* for a discussion on the board dependency for serial load images):

- A valid 'board' configuration for your hardware and application.

5.1 stpmake - tool overview

To build firmware images, the SDK tool 'stpmake' is used. The stpmake is a command line tool that assembles firmware images. The stpmake can generate binary images for:

- Direct programming into Flash
- Use with the network update tool 'stpnetutil'
- Use with the serial loader 'sfwloader' in the boot recovery mode.

The tool will identify any required options as well as defaults that are selected.

Running the stpmake without any arguments yields:

```
$ stpmake

stpmake v1.11
Info: --type not specified: defaulting to "stpfw_upgrade"
Info: --format not specified: defaulting to "stpupg"

------------------------------------------------------------
Operation failed.

Errors encountered:

  Error: Required option --stpfw_path not specified.
```

One option that is mandatory (based on the assumed defaults of --type=stpfw_upgrade, and --format=stpupg) is the path to the stpfw. To get full help for the tool, run stpmake --help.

Tip: when running the stpmake from the command line, start with few parameters, and allow the error checking to tell you what you are missing. Following this iterative approach to constructing the command line will be the easiest way to use this tool.

5.2 stpmake - common options

5.2.1 --list boards

This will list the available board configurations in both- the SDK and in user supplied boards. Supplied board configurations are in separate directories under: $STP_SDK_DIR/board, and if $STP_SDK_USER_BOARDS is defined, that directory will also be searched for user supplied board configurations.

5.2.2 --outbase

```
--outbase=basename
```

This is an optional argument; it specifies the base path/filename for output images. Note that this is the base file name, so if a path is specified, a trailing / must be used.

5.2.3 -- format

```
-- format=stpupg|raw|sfwloader
```

This option will specify what type of images to generate. If not specified, format defaults to stpupg.

- stpupg
  - Images to be used with the stpnetutil network update tool.
- raw
  - Raw images that can be written directly to Flash.

    - Note: For building raw NAND firmware images, additional processing must be done on the raw image to add NAND block headers needed by the bootloader; and depending on the method used to pre-program the NAND, the NAND spare area (ECC) may need to be added. See Appendix C: Considerations for pre-programming NAND devices with STreamPlug firmware on page 52 for more details.
- sfwloader
  - Images to be used with the low level serial loader tool.

5.2.4 --stpfw_path

```
--stpfw_path=path
```

This options specifies the path to the stpfirmware directory of a release. This directory must include a build_info.dat file.

5.2.5 --vmlinux

```
--vmlinux=path_to_vmlinux
```

When building Linux firmware (Linux firmware is identified to the stpmake by the type key from the build_info.dat in the --stpfw_path folder).

5.2.6 --infile

`--infile=path`

When building a file system image, this specifies the path of the filesytem image.

*Note: When building a UBI file system image, use the *.ubi file, NOT the *.ubifs file.*

5.2.7 --type

`--type=stpfw_factory|stpfw_upgrade|rootfs|aux1fs|aux2fs|raw_<sectionname>`

This option specifies the type of an image being built. The default value is stpfw_upgrade.

- stpfw_factory
- stpfw_upgrade
- rootfs
- aux1fs
- aux2fs
- raw_<sectionname>
  - raw_<section_name> is only valid for an sfwloader image. <section_name> is the name of a section defined in the memmap. This could be useful for writing some raw data to specific locations in Flash.

5.2.8 --board

`--board=board_name`

The option specifies the board configuration name. This can be a name returned by --list boards, or the full path to a board directory. The board directory must contain a board*.cfg file to specify the necessary parameters for the board (see *Section 4.3.1: Board configuration - board.cfg on page 15* for more details on board.cfg).

Tip: If you are unsure of the board name while constructing a command line, use anything like '--board=XXXX'. On error, the tool will list the available options.

5.3 stpmake - usage examples

The following examples will detail the most common uses of stpmake. Each usage example below builds on the previous one so it is advised to understand each example in order.

5.3.1 Using stpmake to build sfwloader and stpupg UPGRADE images for bridge firmware

Building the stpupg (network update file for use with 'stpnetutil' tool) is the easiest; the only required option is the path to the stpfirmware. You can see from the tool output that it defaults to make 'upgrade' and 'stpupg' images.

```
$ stpmake --stpfw_path=firmware/stp-bridge-release-1a0e201c2812/rel-stp-
1.1-bridge-release-1a0e201c2812/stpfirmware


stpmake v1.11
Info: --type not specified: defaulting to "stpfw_upgrade"
Info: --format not specified: defaulting to "stpupg"
```

```
stp_gen_image v1.33
Info: Memmap not specified, cannot check image max size
Building: ./rel-stp-1.1-bridge-release-1a0e201c2812-UPGRADE.stpupg
```

Of course you can be more explicit and specify all parameters. All further examples will be explicit in specifying all options.

```
$ stpmake --type=stpfw_upgrade --format=stpupg --stpfw_path=firmware/stp-bridge-release-1a0e201c2812/rel-stp-1.1-bridge-release-1a0e201c2812/stpfirmware
```

```
stpmake v1.11
stp_gen_image v1.33
Info: Memmap not specified, cannot check image max size
Building: ./rel-stp-1.1-bridge-release-1a0e201c2812-UPGRADE.stpupg
```

When looking at this output notice the message indicating the tool cannot validate that the image would fit. This is because it does not know anything about the memory map. If memory size should be checked, always provide a --board option so the tool can know the size of the upgrade image.

Building the sfwloader version of the image does require the --board option since it needs to know the DDR memory type in order to load and execute the programming algorithm (see *Note:* in *Section 3.4: Boot recovery mode on page 11* for more details on this). If you forget to add the --board option when generating an sfwloader image, the tool will remind you it is necessary. Generating the sfwloader image can be done as follows:

```
$ stpmake --board=test_board --type=stpfw_upgrade --format=sfwloader --stpfw_path=firmware/stp-bridge-release-1a0e201c2812/rel-stp-1.1-bridge-release-1a0e201c2812/stpfirmware
```

```
stpmake v1.11
Info: Using board:  - (My Custom board)
Info: Board path:
stp_gen_image v1.33
Building: ./rel-stp-1.1-bridge-release-1a0e201c2812-UPGRADE.smiprog
```

The output image built has the extension of *.smiprog. The sfwloader tool will accept files of .smiprog or *.nandprog. The extension is simply for convenience to know what is being programmed. The device being programmed is determined by the memory map in the board configuration.

5.3.2 Using stpmake to build sfwloader and stpupg UPGRADE images for Linux firmware

Building sfwloader and stpupg binaries for Linux firmware is nearly identical to the building for bridge firmware (see *Section 5.3.1*) but with the addition of the --vmlinux command line option to specify the Linux kernel. When building an image for Linux, the STreamPlug firmware is combined with the Linux kernel to create one image.

Caution: **IMPORTANT: RISK OF INCOMPATIBLITY.**
There exist version dependencies between the Linux kernel and STreamPlug firmware. Always ensure to use compatible firmware and Linux sources.

Building the stpupg can be done as follows:

```
$ stpmake --type=stpfw_upgrade --format=stpupg --stpfw_path=firmware/stp-
lnx-release-400e201c2a12/rel-stp-1.1-stplnx-release-
400e201c2a12/stpfirmware --vmlinux=linux/prebuilt/vmlinux

stpmake v1.11
Preparing linux image.. step1.
Preparing linux image.. step2.
Info: Lnx Kernel Version: streamplug-1.1a
stp_gen_image v1.33
Info: Memmap not specified, cannot check image max size
Building: ./rel-stp-1.1-stplnx-release-400e201c2a12-UPGRADE.stpupg
```

Building the Linux image has a few more steps as the Linux kernel must be weaved with the firmware image to create the binary. You can note that the tool also retrieves the kernel version from the Linux kernel and displays it here as “streamplug-1.1a”. This version is encoded into the stpupg file, and during runtime can be queried using discovery (see *Section B.3.3: Discovery and specifying a STP device on page 37* for using the stpnetutil for discovery). It is also possible to get the firmware BuildID and Linux version information from an *.stpupg file using the --info option to stpnetutil (see section *Section B.3.4: Examine firmware image on page 39* for more details).

Building the sfwloader file is similar except the --board option and a change of the format to sfwloader is required.

```
$ stpmake --board=test_board --type=stpfw_upgrade --format=sfwloader --
stpfw_path=firmware/stp-lnx-release-400e201c2a12/rel-stp-1.1-stplnx-
release-400e201c2a12/stpfirmware --vmlinux=linux/prebuilt/vmlinux

stpmake v1.11
Info: Using board:  - (My Custom board)
Info: Board path:
Preparing linux image.. step1.
Preparing linux image.. step2.
Info: Lnx Kernel Version: streamplug-1.1a
stp_gen_image v1.33
Building: ./rel-stp-1.1-stplnx-release-400e201c2a12-UPGRADE.smiprog
```

5.3.3 Using stpmake to build sfwloader and stpupg FACTORY images for bridge firmware

The building factory images requires --board to be specified since the tool will build the manufacturer parameter region into the factory image. The only difference between the building sfwloader and stpupg images is the --format command line options. Examples of both are as follows:

stpupg:

```
$ stpmake --board=test_board --type=stpfw_factory --format=stpupg --
stpfw_path=firmware/stp-bridge-release-1a0e201c2812/rel-stp-1.1-bridge-
release-1a0e201c2812/stpfirmware


stpmake v1.11
Info: Using board:  - (My Custom board)
Info: Board path:
stp_gen_image v1.33
Building: ./rel-stp-1.1-bridge-release-1a0e201c2812-FACTORY.stpupg
```

sfwloader:

```
$ stpmake --board=test_board --type=stpfw_factory --format=sfwloader --
stpfw_path=firmware/stp-bridge-release-1a0e201c2812/rel-stp-1.1-bridge-
release-1a0e201c2812/stpfirmware


stpmake v1.11
Info: Using board:  - (My Custom board)
Info: Board path:
stp_gen_image v1.33
Building: ./rel-stp-1.1-bridge-release-1a0e201c2812-FACTORY.smiprog
```

5.3.4 Using stpmake to build sfwloader and stpupg FACTORY images for Linux firmware

The building on the examples above, it should be easy to conclude that the building factory images for Linux firmware would be the same as the bridge with the addition of the --vmlinux option to stpmake.

```
$ stpmake --board=test_board --type=stpfw_factory --format=sfwloader --
stpfw_path=firmware/stp-lnx-release-400e201c2a12/rel-stp-1.1-stplnx-
release-400e201c2a12/stpfirmware --vmlinux=linux/prebuilt/vmlinux


stpmake v1.11
Info: Using board:  - (My Custom board)
Info: Board path:
Preparing linux image.. step1.
Preparing linux image.. step2.
Info: Lnx Kernel Version: streamplug-1.1a
stp_gen_image v1.33
Error: Image ./rel-stp-1.1-stplnx-release-400e201c2a12-FACTORY.fw.bin
exceeds max size allowed (1048576)
Error running stp_gen_image
```

Notice that this build failed! In the board “test_board”, the standard SMI memory map that allocates only 1 MB to the factory firmware region. This build has failed because the resultant image (including the Linux kernel) is too big to fit.

Some options to remedy this situation are:

- Increase the size of the factory firmware region in the memory map.
- Use a bridge firmware image in the factory region, and use the upgrade regions for Linux. This is a reasonable choice as the bridge firmware would allow the network update to occur if there was some failure to complete an update of the Linux firmware.

Note on Ethernet configurations: *the Linux command line specifies the configuration of the Ethernet interface for Linux. There are three settings for Ethernet: on, off, and rtos. These are described in more details in the UM1942. Simply stated, when eth=on, Linux has ownership of the Ethernet interface meaning that Linux sees two Ethernet interfaces (PLC and Ethernet). When eth=rtos, the STreamPlug firmware owns the Ethernet interface and implements L2 bridging while exposing only one interface to Linux. This is discussed here as it relates to the decision to use a bridge firmware image in the factory region. In order for the bridge firmware to the bridge, eth must be set to rtos. If eth is set to 'on', the PLC modem will still function but will not bridge any traffic. It will, however respond to discovery and the firmware update on both interfaces.*

5.3.5 Using stpmake to build sfwloader and stpupg rootfs images

It may be useful to use the sfwloader or stpnetutil network updater to load a Linux file system. This example is for the rootfs, but easily extended to aux1/2 fs. (See documentation on the --type command line option in *Section 5.2.7: --type on page 26*).

```
$ stpmake --type=rootfs --format=stpupg --
infile=linux/prebuilt/rootfs/minimal_smi/rootfs.ubi


stpmake v1.11
Generating ./rootfs.stpupg.
Done.
```

Generating the sfwloader needs the --board option:

```
$ stpmake --board=test_board --type=rootfs --format=sfwloader --
infile=linux/prebuilt/rootfs/minimal_smi/rootfs.ubi


stpmake v1.11
Info: Using board:  - (My Custom board)
Info: Board path:
stp_gen_image v1.33
Info: Using SMI, Bank 1, Offset=0x600000, Len=0xa00000
Building: ./rootfs-DDR2_1x_x16_256MB-release.smiprog
Done.
```

Note regarding Linux file systems: *the UBI Linux file system images need to be built with the correct geometry information. In the prebuilt configurations there are examples for the most common SMI and NAND device geometries. If you are building custom hardware with different memories, it may be necessary to modify the buildroot configuration for the UBI file system generation.*

5.3.6 Using stpmake to build raw FACTORY images for bridge firmware

Building a raw image file is necessary if you wish to pre-program into Flash at production time. The other formats (*.[smi|nand]prog, *.stpupg) all have header and other information that encapsulates the raw firmware binary file.

A raw firmware file has a suffix of ".fw.bin". The following example will build a raw factory image for the board name "test_board":

```
$ stpmake --board=test_board --type=stpfw_factory --format=raw --
stpfw_path=firmware/stp-bridge-release-1a0e201c2812/rel-stp-1.1-bridge-
release-1a0e201c2812/stpfirmware


stpmake v1.11
Info: Using board:  - (My Custom board)
Info: Board path:
stp_gen_image v1.33
$ ll
-rw-rw-r--. 1 macaluso macaluso   586124 Jul 23  2015 rel-stp-1.1-bridge-
release-1a0e201c2812-FACTORY.fw.bin
```

When building a raw file it does not contain any information about where it should be placed into Flash - for this you need to consult the memory map (although it is a rule that factory images resides at the beginning of Flash devices, but for an upgrade image the memory map would need to be consulted).

After building this image, it is possible to use a Flash device programmer to program the .fw.bin file at the address 0x0 of a SMI Flash device.

Caution: **IMPORTANT: RISK OF INCORRECT PROGRAMMING**
fw.bin files CANNOT be directly programmed into NAND Flash.
(See note in *Section 5.2.3: -- format on page 25*, as well as *Appendix C: Considerations for pre-programming NAND devices with STreamPlug firmware on page 52* for more details).

6 Loading firmware

Firmware can be loaded onto the device using either the serial loader, or the network update tool 'stpnetutil'.

A factory image MUST be present on a device in order for it to boot. Upgrade images are optional, and if running Linux firmware, a Linux root file system must be loaded.

6.1 Load using sfwloader

The serial loader uses the *.smiprog files, or the *.nandprog files. For details on how to load using the serial loader see *Appendix A: sfwloader*.

For details on how to load firmware using the network loader, see *Section B.3.6: Firmware update on page 41*.

```
$ sfwloader.exe --port=com1 --baudrate=460800 --image=file.smiprog
```

If loading a new device via the serial loader it is recommended to load the root file system first, then the upgrade image, finally the factory image. Following this method will be the easiest since the system will not be able to properly boot until the factory image is present, loading in the reverse order will avoid having to manually enter boot recovery mode after each image.

6.2 Load using stpnetutil

The network loader uses the *.stpupg files. The network loader encrypts the firmware when sending it to the device. It is important to know the HomePlug network membership password to be able to update the device.

```
# stpnetutil --if-index=1 --remote-mac=02:AB:01:B6:20:22 --update --
nmk=HomePlugAV --firmware=stplnx-release-180a181c6212-UPGRADE.stpupg
```

For details on how to load firmware using the network loader, see *Section B.3.6*.

Appendix A sfwloader

The sfwloader tool is used to feed a firmware image to the STreamPlug while the device is in the boot recovery mode (see *Section : Manually entering boot recovery mode on page 12*), or when set to boot from the UART via BMODE pins (see *Appendix D: STreamPlug boot source selection on page 53*) for more details on the BMODE pin definition.

Baud rates up to 460 k are supported. If using 460 k baud, ensure that the serial port of the PC you are using can support this data rate. USB serial converters with FTDI chipsets have proven to be the most reliable.

The required arguments to sfwloader are: the port and image. On a Windows PC the port is specified as e.g.: “comx”, while on Linux the port is specified as the device e.g.: “/dev/ttyS1”.

An example of running sfwloader is:

```
$ sfwloader.exe --port=com18 --baudrate=460800 --image=rootfs-
DDR2_1x_x16_256MB-release.smiprog
Firmware loader v0.2

Sending BEACON - waiting for response from target device..
Got BEACON response.. Waiting for data requst..
Image 0: (     0/5252076) 0.0% Complete.
Image 0: (   506/5252076) 0.0% Complete.
Image 0: (  1012/5252076) 0.0% Complete.
Image 0: (  1518/5252076) 0.0% Complete.
Image 0: (  2024/5252076) 0.0% Complete.
Image 0: (  2530/5252076) 0.0% Complete.
Image 0: (  3036/5252076) 0.1% Complete.
...
```

Appendix B stpnetutil

B.1 Overview of tool

Stpnetutil provides a set of network utilities that can be used to interact with an STP device. It can be executed from Windows, Linux as well as the on-board STP Linux. Stpnetutil provides utilities to perform:

- Discovery of STP devices
- Firmware update
- Security key (NMK) configuration
- Powerline network discovery
- Bridging information discovery
- Channel performance reporting

B.2 Prerequisites

Stpnetutil requires the pcap libraries to be installed.

For Windows, download and install WinPcap (www.winpcap.org).

For Linux, install the libpcap libraries. If running on a x86_64 machine, make sure to also install the load of the i686 libpcap libraries. Depending on the version installed, it may be necessary to symlink to the shared library that the tool is looking for.

B.3 Stpnetutil usage

Most command line options for stpnetutil can be specified using a long or short option format. 'stpnetutil --help' will display all available options in a long format, such as '--list-interfaces', and a short format, such as '-l'. For this document only the long option format will be used. Use the short format if desired, as shown in 'stpnetutil --help'.

The three distributions of stpnetutil (Windows, Linux, and STP Linux) all operate identically. This documentation can be equally applied across all three.

B.3.1 Help and usage

stpnetutil includes command line help and example usage for quick reference.

To display the help, use 'stpnetutil --help'

```
% stpnetutil --help
stpnetutil v1.4 - STreamPlug network utility

Usage: stpnetutil [OPTION...]

Options:
  -l, --list-interfaces           Display a list of network interfaces.
  -a, --if-name                   Select network interface by name.
  -d, --if-index                  Select network interface by index.
```

```
  -r, --remote-mac=mac-address     Mac address of STreamPlug device.
  -v, --discover-local             Discover only locally connected
STreamPlug
                                   devices.
  -V, --discover-all               Discover all STreamPlug devices.
  -c, --discover-csv-output        output discovery info in CSV, machine
                                   parseable format.
  -f, --firmware=filename          Firmware file to use for upgrade.
      --info                       Display information about firmware file.
  -u, --update                     Update all devices using specified
                                   firmware. Use with -r or -v.
      --set-nmk                    Set NMK on device - must also specify
                                   --nmk=<password>
  -N, --nmk=password               Use NMK password to derive encryption key.

Utility functions:
  -b, --bss-list                   List BSS's seen by a device. Use with -r or
                                   -v
      --chanqual-sub               Subscribe to channel quality measurements.
                                   Can combine with --chanqual-listen, Use
                                   with -r or -v
      --chanqual-unsub             Unsubscribe to channel quality
                                   measurements. Use with -r or -v
      --chanqual-listen            Listen count for chanqual messages.
                                   0=continuous
  -t, --tei-list                   Request simple list of available TEIs on a
                                   network. Use with -r or -v
  -T, --tei-snapshot               Requests list of bridged addresses for a
                                   particular TEI. Use with -r or -v
      --tei-query                  Set the Tei to query --tei-query=<tei>

Help functions:
  -?, --help                       Show this help message
      --usage                      Show usage examples
```

To display the example usage:

```
% stpnetutil --usage
stpnetutil v1.4 - STreamPlug network utility

Example usage
=============
Discover a local STP device:
-------------------------------------
    stpnetutil --if-name=eth0 --discover-local

Update a local STP device:
```

```
-------------------------------------
    stpnetutil --if-name=eth0 --discover-local --update -nmk=HomePlugAV --
firmware=UpgradeImage.stpupg

Update a specific STP device:
-------------------------------------
    stpnetutil --if-name=eth0 --remote-mac=00:11:22:33:44:55 --update --
nmk=HomePlugAV --firmware=UpgradeImage.stpupg

Change the NMK on an STP device:
-------------------------------------
    stpnetutil --if-name=eth0 --remote-mac=00:11:22:33:44:55 --set-nmk --
nmk=Key1
NOTE: The set-nmk function can only be used for locally attached devices.
      The --remote-mac option must be used. set-nmk cannot be used over the
powerline.
```

B.3.2 Finding and selecting network interface

If stpnetutil is run without any options it will first tell you that it needs a network interface to be specified:

```
$ ./stpnetutil
stpnetutil v1.4 - STreamPlug network utility

Error: Interface must be specified by --if-name, OR --if-index.
       use --list-interfaces to display available interfaces.

Exiting with error: Try --help or --usage
```

Stpnetutil must be provided with the network interface to use. The command-line option '--list-interfaces' exists to provide the user with a list of the available interfaces. The output of this option slightly differs between Windows and Linux.

For Windows:

```
C:\>stpnetutil --list-interfaces
stpnetutil v1.4 - STreamPlug network utility

Index: 1
  Name        : \Device\NPF_{4DA37FDC-6947-490D-B3B4-EDCDE90183BF}
  Description: Microsoft
  MAC         : 00:11:22:33:44:55
  Link        : DOWN
  IP Adddr    : 167.4.76.254
  Netmask     : 255.255.255.128
Index: 2
  Name        : \Device\NPF_{46C5767A-1047-4B05-B0B3-1B570F627E14}
  Description: Intel(R) 82577LM Gigabit Network Connection
  MAC         : 00:aa:bb:cc:dd:ee
```

```
  Link        : UP
  IP Adddr    : 167.4.73.46
  Netmask     : 255.255.255.128
```

It is known that the wired LAN adapter is the correct interface and that is shown at the Index 2. When using Windows it is easier to refer to the network interface by the index rather than by the name. To tell stpnetutil to use the Index 2 interface it is specified as follows:

```
stpnetutil --if-index=2
```

For Linux:

```
# stpnetutil --list-interfaces
stpnetutil v1.4 - STreamPlug network utility

Index: 1
  Name        : eth0
  MAC         : 00:11:22:33:44:55
  Link        : UP
  IP Adddr    : 192.168.1.5
  Netmask     : 255.255.255.128
  Netmask     : 0.0.0.0
Index: 2
  Name        : eth1
  MAC         : 00:aa:bb:cc:dd:ee
  Link        : UP
Index: 3
  Name        : lo
  MAC         : 00:00:00:00:00:00
  Link        : UP
  IP Adddr    : 127.0.0.1
  Netmask     : 255.0.0.0
  Netmask     : 0.0.0.0
```

In Linux the --if-index option also can be used, but it is also common to use the interface name. In the above example, eth0 is the interface that is desired; this interface is listed at the Index 1. This interface can be selected by using the name as follows:

```
% stpnetutil --if-name=eth0
```

The interface index can also be used as it was in the Windows example above:

```
% stpnetutil --if-index=1
```

For the remainder of this document the --if-index format will be used.

B.3.3 Discovery and specifying a STP device

stpnetutil is designed to exchange Ethernet packets with the STreamPlug modem. To do this the MAC address of the STP device to talk to is needed. The command-line options --discover-all and --discover-local can be used to reveal all connected STP devices.

The option --discover-local will only reveal STP devices that are connected to the local wired Ethernet interface. The option --discover-all, as the name suggests, will identify the same

devices as --discover-local, as well as any STP devices that are connected over the powerline.

For example, let's assume there are two STP devices, A and B. 'A' is connected to the Ethernet port on my Windows workstation (at -if-index=1) and 'B' is only connected via powerline.

```
% stpnetutil --if-index=1 --discover-local
stpnetutil v1.4 - STreamPlug network utility

Looking for STreamPlug devices on IfNdx:1...
press any key to stop and print summary.

.
Found Device #  1: MAC: 02:AB:01:B6:20:22, BuildID: 420e1a1a0e12, Ver:stp
v1.1(l), KernelId: streamplug-1.1a, AppVer:

Summary of Devices:
Found 1 device.:
------------------------------------------------------------------------
          Device MAC: 02:AB:01:B6:20:22
         Device Type: Generic STreamPlug
    Firmware Version: stp v1.1(l)
    Firmware BuildID: 420e1a1a0e12
Linux Kernel Version: streamplug-1.1a
          Device UID: 001158B2B6202201
       Hardware Name: Tatung M1i
    Hardware Version: A
------------------------------------------------------------------------
```

Notice that there is a locally connected STP device with a modem MAC address 02:AB:01:B6:20:22. The firmware version is “stp v1.1(l)” which means it is the STP Linux version [as noted by (l) in the version name] of the STreamPlug release v1.1. This is the selected device 'A'.

Consider the --discover-all option:

```
% stpnetutil --if-index=1 --discover-all
stpnetutil v1.4 - STreamPlug network utility

Looking for STreamPlug devices on IfNdx:1...
press any key to stop and print summary.

.
Found Device #  1: MAC: 02:AB:01:B6:20:22, BuildID: 420e1a1a0e12, Ver:stp
v1.1(l), KernelId: streamplug-1.1a, AppVer:
Found Device #  2: MAC: 02:AB:01:B6:20:36, BuildID: 220e1a1a0c12, Ver:stp
v1.1(b)
```

```
Summary of Devices:
Found 2 devices.:
------------------------------------------------------------------
          Device MAC: 02:AB:01:B6:20:22
         Device Type: Generic STreamPlug
    Firmware Version: stp v1.1(l)
    Firmware BuildID: 420e1a1a0e12
Linux Kernel Version: streamplug-1.1a
          Device UID: 001158B2B6202201
       Hardware Name: Tatung M1i
    Hardware Version: A
------------------------------------------------------------------
------------------------------------------------------------------
          Device MAC: 02:AB:01:B6:20:36
         Device Type: Generic STreamPlug
    Firmware Version: stp v1.1(b)
    Firmware BuildID: 220e1a1a0c12
          Device UID: 001158B2B6203601
       Hardware Name: Tatung M1i
    Hardware Version: A
------------------------------------------------------------------
```

Two devices are shown. Device 'A' is shown as in the above example, as well as another device with the MAC address 02:AB:01:B6:20:36, running the bridge version of STP release v1.1.

When using the stpnetutil utility functions, as shown in --help, a single STP device to talk to needs to be specified. This can be done by using the '--remote-mac' option, as follows:

```
% stpnetutil --if-index=1 --remote-mac=02:AB:01:B6:20:22
```

B.3.4 Examine firmware image

The '--info' option is useful in displaying version information for a firmware image file.

```
# stpnetutil --info --firmware=generic-128MB-smi-rel-stp-v1.1beta1-bridge-
release-2c06301c1812-FACTORY.stpupg

stpnetutil v1.4 - STreamPlug network utility

File info:
    Type: Factory Firmware
 BuildID: 2c06301c1812
```

Notice that the image type is “Type: Factory Firmware” and that the BuildID is 2c06301c1812.

```
# stpnetutil --info --firmware=bridge-debug-620c34183c12-UPGRADE.stpupg

stpnetutil v1.4 - STreamPlug network utility
```

```
File info:
    Type: Upgrade Firmware
 BuildID: 620c34183c12
```

Notice that the image type is “Type: Upgrade Firmware” and that the BuildID is 620c34183c12.

B.3.5 Setting the security key (NMK)

The --set-nmk option is used to set the NMK key for the STP device. This is a special operation and has the following requirements:

- Can only be used with a locally connected STP device. 'Locally connected' means over a wired Ethernet connection to the device, or via STP Linux. It cannot be used over the powerline.
- Can only be used with '--remote-mac'. You cannot use '--discovery-local' or '--discover-all'.

Example 3 If the MAC address of the device is not known, upon the first use, change the NMK using '--discover-local' as follows:

```
# stpnetutil --if-index=1 --discover-local
stpnetutil v1.4 - STreamPlug network utility

Looking for STreamPlug devices on IfNdx:1...
press any key to stop and print summary.

.
Found Device #  1: MAC: 02:AB:01:B6:20:22, BuildID: 420e1a1a0e12, Ver:stp
v1.1(l), KernelId: streamplug-1.1a, AppVer:

Summary of Devices:
Found 1 device.:
-----------------------------------------------------------------
          Device MAC: 02:AB:01:B6:20:22
         Device Type: Generic STreamPlug
    Firmware Version: stp v1.1(l)
    Firmware BuildID: 420e1a1a0e12
Linux Kernel Version: streamplug-1.1a
          Device UID: 001158B2B6202201
       Hardware Name: Tatung M1i
    Hardware Version: A
-----------------------------------------------------------------
```

Notice there is a locally connected device at the MAC address 02:AB:01:B6:20:22. The the '--set-nmk' command can be formed.

```
# stpnetutil --if-index=1 --remote-mac=02:Ab:01:b6:02:22 --set-nmk --
nmk=HomePlugAV
```

In this example the NMK is forcibly being changed to the default 'HomePlugAV'. This allows for the recovery of a STP device for which the password has been lost.

When the command is run, the result is:

```
# stpnetutil --if-index=1 --remote-mac=02:Ab:01:b6:20:22 --set-nmk -
nmk=HomePlugAV
stpnetutil v1.4 - STreamPlug network utility

Running setNmk
NMK Set successfully.
```

When completed successfully, observe the string: 'NMK Set successfully'. If the operation fails note the following exchange:

```
# stpnetutil --if-index=1 --remote-mac=02:Ab:01:b6:20:21 --set-nmk --
nmk=HomePlugAV
stpnetutil v1.4 - STreamPlug network utility

Running setNmk
Did not get response from device - check MAC address
```

Referring to the help for stpnetutil, it can be seen that the --nmk option takes a password. This is different from the actual NMK. The NMK is a key that is derived (by using an algorithm defined by the IEEE 1901 standard) from the password. For the purposes of convenience when using the stpnetutil tool, the --nmk command line option accepts the network membership password. For more information on converting network membership passwords into keys, see *Appendix E: Hashing passwords to keys on page 54*.

B.3.6 Firmware update

Stpnetutil can be used to update STP firmware using the '--update' option. The STP device can be specified with the '--remote-mac' or '--discover-local' options. When using multiple STP devices it is recommended that you use the --remote-mac option to specify the device you want to update.

Firmware update example:

```
# stpnetutil --if-index=1 --remote-mac=02:AB:01:B6:20:22 --update --
nmk=HomePlugAV --firmware=stplnx-release-180a181c6212-UPGRADE.stpupg
```

In the above example the '--update' option along with the '--nmk' and '--firmware' options is used. '--update' is used to signal the firmware update operation. '--nmk' is used to provide the network membership key that is set on the device. '--firmware' specifies the file containing the update.

If an update fails to start, note the following:

```
# stpnetutil --if-index=1 --remote-mac=02:AB:01:B6:20:22 --update --
nmk=HomePlugAV --firmware=stplnx-release-180a181c6212-UPGRADE.stpupg
stpnetutil v1.4 - STreamPlug network utility

Update starting..


!!! Error 4: Timed out waiting for response from receiver. !!!

Progress: 84/3606832 (0.0%)
Done.
```

A successful update looks like:

```
# stpnetutil --if-index=1 --remote-mac=02:AB:01:B6:20:22 --update --
nmk=HomePlugAV --firmware=stplnx-release-180a181c6212-UPGRADE.stpupg
stpnetutil v1.4 - STreamPlug network utility

Update starting..
Progress: 16984/3606832 (0.5%)
Progress: 33884/3606832 (0.9%)
Progress: 50784/3606832 (1.4%)
Progress: 67684/3606832 (1.9%)
Progress: 84584/3606832 (2.3%)
Progress: 101484/3606832 (2.8%)

...
Progress: 3481484/3606832 (96.5%)
Progress: 3498384/3606832 (97.0%)
Progress: 3515284/3606832 (97.5%)
Progress: 3532184/3606832 (97.9%)
Progress: 3549084/3606832 (98.4%)
Progress: 3565984/3606832 (98.9%)
Progress: 3582884/3606832 (99.3%)
Progress: 3599784/3606832 (99.8%)
Progress: 3606832/3606832 (100.0%)
Done.
```

B.3.7 BSS list - display AV networks visible to a specific node

A request can be sent to a specific STP device and have it report on the powerline networks (BSSs) that it can see using the '--bss-list' option.

```
# stpnetutil --if-index=1 --remote-mac=02:AB:01:B6:20:22 --bss-list
stpnetutil v1.4 - STreamPlug network utility


Running bssList
2 BSSs found.

 |-SNID:06, NID: c7a22b104ff604, REMOTEBSS
 +----->       Beacon Age: 250955
 +----->   Reliability(F):  100
 +----->   Reliability(P):  100
 +----->          Cur AGC:   -8
 +----->          Min AGC:   -8
 +----->          Max AGC:   -8
 +----->      NetworkMode:    2
 +----->       HybridMode:    2
 +----->             Stei:   13
```

```
+----->       BeaconType:    0
+----->             Uncr:    0
+----->             Npsm:    0
+----->         NumSlots:    0
+----->        SlotUsage:  0x1
+----->           SlotID:    0
+----->            Aclss:    0
+----->             Hoip:    0
+----->            Rtsbf:    0
+----->            Bmcap:    0
+----->              Rsf:    0
+----->           Plevel:    0

|-SNID:09, NID: e32d84da7deb05, LOCALBSS
+----->       Beacon Age: 250939
+----->   Reliability(F):  100
+----->   Reliability(P):  100
+----->          Cur AGC:   28
+----->          Min AGC:   28
+----->          Max AGC:   28
+----->      NetworkMode:    2
+----->       HybridMode:    2
+----->             Stei:    9
+----->       BeaconType:    0
+----->             Uncr:    0
+----->             Npsm:    0
+----->         NumSlots:    0
+----->        SlotUsage:  0x1
+----->           SlotID:    0
+----->            Aclss:    0
+----->             Hoip:    0
+----->            Rtsbf:    0
+----->            Bmcap:    0
+----->              Rsf:    0
+----->           Plevel:    0
```

Notice from the data returned that there are two BSSs visible from the device 02:ab:01:b6:20:22, SNID:06, NID: c7a22b104ff604, REMOTEBSS and SNID:09, NID: e32d84da7deb05, LOCALBSS. Note that the SNIDs shown are not permanently assigned and can change across reboots. The NIDs for these networks will not change unless the user changes them.

SNID 9 is the network that the device 02:ab:01:b6:20:22 is connected to which can be seen from the LOCALBSS designation. SNID 6 has a REMOTEBSS designation meaning the device 02:ab:01:b6:20:22 is not attached to it, but it can hear it's beacon.

By examining the AGC values for these networks the signal strengths of the beacons that are received by the device 02:ab:01:b6:20:22 can be seen. SNID has an AGC of -8 dB. The

AGC values range from -12 dB to 48 dB where -12 dB represents high signal strength and 48 dB represents a very low signal. SNID 6 has an AGC of -8 dB. This means the beacon manager for this is network is very close by. SNID 9 has an AGC of 28 dB which is an indication that this beacon manager is farther away than the beacon manager of SNID 6.

The reliability measurements indicate how reliable the reception of this beacon is. Notice that both SNIDs have a reliability (P) measurement of 100 which means the complete beacon (P)ayload has been received 100% of the time. This is the greatest reliability.

B.3.8 Bridging information discovery

Stpnetutil provides a facility to reveal details about the current state of the source-aware bridging that is used. Every device maintains a list of MAC addresses for devices that it is locally bridging for.

There are three options that are used to collect the bridging information:

- --tei-list
- --tei-snapshot
- --tei-query

'--tei-list' asks the indicated STP device to report all the terminal equipment identifiers of which it is aware.

```
# stpnetutil --if-index=1 --remote-mac=02:ab:01:b6:20:22 --tei-list
stpnetutil v1.4 - STreamPlug network utility


Running teiList
TeiCount:2
5
9
```

Notice that two TEIs were returned: 5 and 9. Use the '--tei-snapshot' and '--tei-query' options to query one of these TEIs, 11 as shown below.

```
# stpnetutil --if-index=1 --remote-mac=02:ab:01:b6:20:22 --tei-snapshot --
tei-query=5
stpnetutil v1.4 - STreamPlug network utility


Running teiList
Tei:5 AddrCount:3
02:AB:01:B6:20:22 Local MAC
02:AB:02:B6:20:22 Local Bridged MAC
68:B5:99:F6:50:C0 Local Bridged MAC
```

Notice that there are three addresses in the local bridge table:

- 02:ab:01:b6:20:22 (modem)
- 02:ab:02:b6:20:22 (STP Linux)
- 68:b5:99:f6:50:C0

02:ab:01:b6:20:22 is the device that was queried so now it is clear that TEI 5 belongs to 02:ab:01:b6:20:22 and 68:b5:99:f6:50 is the MAC address of the workstation on the network used to do the query.

This makes a sense since the locally connected device is 02:ab:01:b6:20:22 and that it's running STP Linux. The device at TEI 9 is the remote device. To query the TEI:

```
# stpnetutil --if-index=1 --remote-mac=02:ab:01:b6:20:22 --tei-snapshot --
tei-query=9
stpnetutil v1.4 - STreamPlug network utility


Running teiList
Tei:9 AddrCount:1
02:AB:01:B6:20:36 Remote MAC
```

Notice that the remote MAC address for TEI 9 is 02:ab:01:b6:20:36. The only device being bridged from that address is itself.

```
If a '--tei-query' is run on that device for TEI 9:
# stpnetutil --if-index=1 -r 02:ab:01:b6:20:36 --tei-snapshot --tei-query=9
stpnetutil v1.4 - STreamPlug network utility


Running teiList
Tei:9 AddrCount:1
02:AB:01:B6:20:36 Local MAC
```

The remote device is now being queried and it is reporting its own MAC address as a “Local MAC” as shown. Querying this same device about the TEI if the STP Linux device, TEI 5 results in:

```
# stpnetutil --if-index=1 --remote-mac=02:ab:01:b6:20:36 --tei-snapshot --
tei-query=5
stpnetutil v1.4 - STreamPlug network utility


Running teiList
Tei:5 AddrCount:3
68:B5:99:F6:50:C0 Remote Bridged MAC
02:AB:02:B6:20:22 Remote Bridged MAC
02:AB:01:B6:20:22 Remote MAC
```

The MAC is the same as the earlier example, but now they are displayed as “Remote”.

B.3.9 Channel quality reporting

STP can report the calculated channel quality to stpnetutil. This is done through the use of the '--chanqual-sub', '--chanqual-unsub', and '--chanqual-listen' options. The feature works by asking an STP device to relay all channel quality measurements to stpnetutil. This feature is enabled on per a device basis which means you will receive measurements for all devices that the STP device is communicating with. Channel quality is calculated when a unicast communication is setup between two powerline devices.

The feature works by subscribing to an STP device for measurements and then listening for those measurements to be received on stpnetutil. The example below shows a "--chanqual-sub' option that will subscribe to measurements, followed by the '--chanqual-listen' option with a value of 0 (zero) (which instructs stpnetutil to listen forever).

```
# stpnetutil --if-index=1 -r 02:ab:01:b6:20:36 --chanqual-sub --chanqual-
listen=0
stpnetutil v1.4 - STreamPlug network utility


Running chanQual

Cnf Parse OKAY
Status: 1-Subscribed

Listening for ChanQual Reports ...

Report Received: TxAddr:02:AB:01:B6:20:36 ==> RxAddr:02:AB:01:B6:20:7E
LocalMac: 02:AB:01:B6:20:36
RemoteMac: 02:AB:01:B6:20:7E
Source: Remote
ResponseType: 0
NumTmi: 1
TmiA[0]: 5
NumInt: 0
NewTmi: 5
Rsvd1: 0
CodeRate: 1
Rsvd2: 0
CBLD[0000]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A
0A 0A 0A
CBLD[0024]: 0A 0A 0A 0A 0A 08 06 04 00 04 04 06 08 08 08 08 08 08 08 08 06
06 06 06
CBLD[0048]: 08 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 0F 0F 0F
0F 0F 0F
CBLD[0072]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 08 08
CBLD[0096]: 08 08 08 08 08 08 08 06 08 0A 0A 08 0A 08 0A 0A 0A 0A 0A 0A 0A
0A 0A 0A
CBLD[0120]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 08 0A 0A
0F 0F 0F
CBLD[0144]: 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
0A 0A 0A
CBLD[0168]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
0A 0A 0A
CBLD[0192]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0F 0F 0F 0F
0F 0F 0F
CBLD[0216]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A
0A 0A 0A
```

```
CBLD[0240]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0264]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0288]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0312]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0336]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0360]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0384]: 0A 0A 0A 08 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 08
0A 08 08
CBLD[0408]: 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08
08 08 08
CBLD[0432]: 08 06 06 06 06 06 04 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06
06 06 06
CBLD[0456]: 06 06 06 06 06 06 06 06 06 06 06 04 04 04 04 04 04 04 04 04 04 04 04 04
04 04 04
CBLD[0480]: 02 01 02 04 04 04 04 03 04 04 04 04 04 04 04 03 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[0504]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 01 02 02 02 02 00 02 02 02
02 02 02
CBLD[0528]: 01 02 02 02 02 01 01 00 00 00 02 00 00 00 00 00 00 00 02 01 01 01 01 00
01 00 00
CBLD[0552]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00
CBLD[0576]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00
CBLD[0600]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00
CBLD[0624]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00
CBLD[0648]: 00 00 00 00 00 00 00 01 01 01 01 00 01 01 01 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[0672]: 0F 0F 0F 00 01 01 00 01 02 02 02 02 02 02 02 02 02 02 02 02 01 02 02 02
CBLD[0696]: 02 03 02 02 02 02 02 03 03 02 02 02 02 02 02 02 02 03 03 02 02 03 04 03
CBLD[0720]: 02 03 03 03 03 03 02 02 02 02 02 02 03 02 03 03 03 02 02 02 02 02 02 02
CBLD[0744]: 03 02 03 03 03 03 03 03 02 02 03 02 03 03 03 03 03 03 02 02 03 03 02 03
CBLD[0768]: 02 02 02 02 02 03 03 03 03 03 02 02 03 03 02 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[0792]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 03 03 03 02 02 03
03
CBLD[0816]: 02 02 02 03 03 03 02 02 01 02 02 02 02 03 02 02 02 03 03 03 02 02 02 02
```

```
CBLD[0840]: 02 02 01 01 02 02 01 01 00 02 02 02 02 03 02 02 02 02 02 02 02
02 03 03
CBLD[0864]: 02 02 03 03 02 02 02 02 02 02 02 03 02 02 02 03 02 02 02 02 02
03 03 03
CBLD[0888]: 03 02 02 02 02 02 02 02 03 03 03 03 03 03 03 03 03 03 04 03
02 03 04
CBLD[0912]: 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04
04 04 04
CBLD[0936]: 04 04 04 04 04 04 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 04 06 06
04 06 04
CBLD[0960]: 04 06 06 04 04 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06
06 06 06
CBLD[0984]: 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 08 08 06 06 06 06
06 06 06
CBLD[1008]: 06 06 06 06 06 06 08 06 06 06 06 06 06 06 08 08 06 06 08 06 06
08 08 08
CBLD[1032]: 08 08 08 06 06 06 06 06 06 08 08 06 06 06 06 06 06 06 06 08 08
08 08 08
CBLD[1056]: 08 08 06 06 06 06 06 08 08 08 08 08 08 08 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1080]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1104]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1128]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1152]: 0F 0F 0F

Report Received: TxAddr:02:AB:01:B6:20:7E ==> RxAddr:02:AB:01:B6:20:36
LocalMac: 02:AB:01:B6:20:36
RemoteMac: 02:AB:01:B6:20:7E
Source: Local
ResponseType: 0
NumTmi: 1
TmiA[0]: 5
NumInt: 0
NewTmi: 5
Rsvd1: 0
CodeRate: 1
Rsvd2: 0
CBLD[0000]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A 0A
0A 08 0A
CBLD[0024]: 0A 0A 08 08 08 08 08 06 04 01 00 00 03 04 06 06 06 06 06 06 06
06 06 06
CBLD[0048]: 04 02 04 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 0F 0F 0F
0F 0F 0F
CBLD[0072]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 08 08
```

```
CBLD[0096]: 0A 0A 0A 0A 0A 0A 0A 08 08 06 04 06 08 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0120]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0F 0F 0F
CBLD[0144]: 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0168]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0192]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0F 0F 0F 0F 0F 0F 0F 0F
CBLD[0216]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0240]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0264]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0288]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0312]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0336]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0360]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
CBLD[0384]: 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 08 0A 0A 0A 0A 0A 08 0A 0A 08
CBLD[0408]: 08 0A 0A 08 0A 08 0A 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08
CBLD[0432]: 08 08 08 08 08 08 08 08 08 08 08 06 08 08 08 08 06 06 06 06 06 06 06 06
CBLD[0456]: 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 04 04 04 04 04 04 02 01 00
CBLD[0480]: 00 00 00 00 00 00 02 04 03 04 04 04 04 04 04 04 0F 0F 0F 0F 0F 0F 0F 0F
CBLD[0504]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 02 04 04 04 04 04 04 04 04 03
CBLD[0528]: 03 04 03 03 03 03 02 03 03 03 03 03 03 03 03 02 01 02 02 02 02 02 02 02
CBLD[0552]: 02 01 01 02 01 02 00 01 01 01 01 00 00 01 00 00 00 00 00 00 00 00 00 00
CBLD[0576]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
CBLD[0600]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
CBLD[0624]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 00 01 00 00 01
CBLD[0648]: 01 01 00 00 01 01 02 02 02 02 02 02 02 02 01 0F 0F 0F 0F 0F 0F 0F 0F 0F
CBLD[0672]: 0F 0F 0F 02 02 02 02 03 03 03 02 03 03 03 03 04 04 03 04 03 03 04 03 04
```

```
CBLD[0696]: 04 04 03 03 03 04 04 04 04 04 03 04 04 03 04 04 04 04 04 04
04 04 04
CBLD[0720]: 04 04 04 04 04 04 04 04 04 04 04 04 04 04 03 04 04 04 04 04
04 04 04
CBLD[0744]: 04 04 04 04 04 04 04 04 04 03 04 04 04 04 04 04 04 04 04 04
04 04 04
CBLD[0768]: 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[0792]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 04 04 03
04 04 04
CBLD[0816]: 04 04 04 03 04 04 03 03 03 04 03 03 03 03 03 04 04 04 03 03
02 03 02
CBLD[0840]: 02 02 02 03 03 03 02 02 03 02 02 02 01 03 03 02 02 03 03 02
02 02 03
CBLD[0864]: 02 02 02 02 02 03 03 02 03 02 02 02 02 02 02 02 02 03 02 03
03 03 02
CBLD[0888]: 03 02 03 02 02 03 03 03 03 02 02 02 01 02 03 02 02 03 03 03
00 00 01
CBLD[0912]: 02 03 03 03 03 03 02 02 02 03 03 02 03 04 04 04 04 04 03 03
04 04 03
CBLD[0936]: 03 04 04 04 04 04 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 04 04
04 04 04
CBLD[0960]: 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04
04 06 04
CBLD[0984]: 04 04 04 04 04 04 06 06 06 06 06 06 06 06 06 06 06 06 06 06
06 06 06
CBLD[1008]: 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06
06 06 06
CBLD[1032]: 04 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06
06 06 06
CBLD[1056]: 06 06 06 06 06 08 06 06 06 06 06 06 08 06 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1080]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1104]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1128]: 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F
CBLD[1152]: 0F 0F 0F
```

It is beneficial to understand the basic mechanism of the powerline communication and how it pertains to the channel quality measurements. With any two powerline devices A and B, the communications channel between them is asymmetrical. That is to say that the channel quality from A to B is not necessarily the same as B to A. To accommodate for this, two receive channel quality measurements are performed, one at A and one at B. The channel quality measurements reported to stpnetutil will reflect this with two measurements for each link between any two powerline devices.

In the example data above the notice two reports. The first is labeled:

```
Report Received: TxAddr:02:AB:01:B6:20:36 ==> RxAddr:02:AB:01:B6:20:7E
```

And the second:

```
Report Received: TxAddr:02:AB:01:B6:20:7E ==> RxAddr:02:AB:01:B6:20:36
```

Note that the MAC addresses are mirrored between the two reports. In the first, the TxAddr is 02:ab:01:b6:20:36. This means that the report reveals the channel quality of transmissions from the TxAddr to the RxAddr. The second report contains the mirrored MAC addresses which reveal the channel quality for transmissions in the reverse direction.

The channel quality report represents the raw channel quality data. The measurement is comprised of the CodeRate and the CBLD data.

CodeRate is defined as 0 = ½ rate and 1 = 16/21 rate. The CBLD (“Carrier Bit LoaD”) data represents 1 carrier per byte, where:

Table 12. CBLD data

CBLD	Bits/carrier
0x00	0
0x01	1
0x02	2
0x03	3
0x04	4
0x06	6
0x08	8
0x0A	10
0x0F	Disabled

CBLD data of 0x0F indicates that the carrier is disabled (not include in the tone mask to comply with regulatory requirements). To calculate the raw channel quality, add up all the CBLD data, ignoring the 0x0F values, and then multiply that sum by the CodeRate. For example, if the sum of CBLD was 5796 (for example) and the CodeRate was 1, then:

Raw channel quality = 5796 * 16/21 = 4,416 bits/(OFDM) symbol

Appendix C Considerations for pre-programming NAND devices with STreamPlug firmware

Note: This section is provided for informational purposes. Please contact ST for details on using the tools mentioned in this section.

To manage the reliable boot from NAND devices, a special block header is needed for each NAND block in the firmware image. The NAND block header contains information to allow for block duplication (redundancy) and image identification via an 'Image Number'. The image number assignment is fixed to 0==factory, 1==upgrade.

There is a utility in the SDK to add these block headers to a firmware image - 'nandfwgen'. An example of using this tool is:

```
$ nandfwgen --infile=rel-stp-1.1-bridge-release-1a0e201c2812-FACTORY.fw.bin
--outfile=rel-stp-1.1-bridge-release-1a0e201c2812-FACTORY.fw.bin.nand --
imgnum=0 --pagedatasize=2048 --pagesparesize=64 --pagesperblock=64
nandfwgen v0.1 - Generation utility for ai2100 NAND firmware init file


Writing DATA  - Blk=  0, Seq=0
Writing DATA  - Blk=  1, Seq=1
Writing DATA  - Blk=  2, Seq=2
Writing DATA  - Blk=  3, Seq=3
Writing DATA  - Blk=  4, Seq=4
Writing DATA  - Blk=  5, Seq=5
Writing DATA  - Blk=  6, Seq=6
Writing DATA  - Blk=  7, Seq=7
Writing DATA  - Blk=  8, Seq=8
```

The output of this tool will be a file containing a block header for every block. This file contains only the data portion of the NAND and does not include any spare area data (ECC data).

Depending on the type of programmer, it may be necessary to pre-compute the ECC data using the BCH4 algorithm that the STreamPlug bootloader uses to perform error correction. There also exists a tool to generate the ECC data. Please contact ST for support if mass programming of NAND devices is needed.

Appendix D STreamPlug boot source selection

The boot source device is set with the BMODE pins on the STreamPlug. The values in *Table 13* are supported.

Table 13. Boot sources

SOC_CFG_BMODE[3:0]	Boot source
0000	Serial Flash (3-byte addressing)
0001	Serial Flash (2-byte addressing)
0010	Parallel NAND Flash (8-bit)
0011	Parallel NAND Flash (16-bit)
0100	UART1 (on MFIO group G20)
0101	UART2 (on MFIO group G21)
0110	Parallel NOR Flash (8-bit)
0111	Parallel NOR Flash (16-bit)

Appendix E Hashing passwords to keys

Keys such as the NMK (network membership key), and the DAK (device authorization key) are derived from human readable passwords. The SDK contains a tool that can be used to generate the keys as well as create random DAKs for use in a production environment.

E.1 hpav-keygen tool

Running the tool without any arguments shows help and usage.

```
$ hpav-keygen
hpav-keygen v0.1 - STreamPlug password/key gen tool

Error: One option: --nmk OR --dak OR --devpw must be specified.


Usage: hpav-keygen [OPTION...]
  -n, --nmk          Generate NMK from supplied password
  -d, --dak          Generate DAK from supplied password
  -d, --devpw        Generate random device PW
  -i, --pwstring     Password string
  -i, --pwfile       Password file, omit, or specify - to read from stdin.
  -o, --outfile      Output file to write key to.  omit, or specify - for
                     stdout.
  -h, --help         Show usage.
```

Using the tool can be easily understood looking at the help. Below are some usage examples.

E.1.1 Generating NMK

Using the command line supplied password:

```
$ hpav-keygen --nmk --pwstring=HomePlugAV
hpav-keygen v0.1 - STreamPlug password/key gen tool

50 d3 e4 93 3f 85 5b 70 40 78 4d f8 15 aa 8d b7
```

Using a password supplied from a file:

```
$ echo HomePlugAV > pw.txt
$ hpav-keygen --nmk --pwfile=pw.txt
hpav-keygen v0.1 - STreamPlug password/key gen tool

50 d3 e4 93 3f 85 5b 70 40 78 4d f8 15 aa 8d b7
```

E.1.2 Generating device password and DAK

The hpav-keygen utility can be used to generate a random device password following the recommended rules for device passwords. This example will show how to generate a device password and also to generate a DAK corresponding to the password.

Note: The algorithm for generating the DAK is different from the algorithm to generate the NMK. (Running --dak and --nmk on the same --pwstring will yield different keys).

Generate a random device password:

```
$ hpav-keygen -devpw
hpav-keygen v0.1 - STreamPlug password/key gen tool

CECK-FMLD-77BZ-GHLZ
```

Generate a random password, save to a file, and generate a DAK also saved to a file:

```
$ hpav-keygen --devpw | tee pw.txt | hpav-keygen --dak --outfile=dak.txt
hpav-keygen v0.1 - STreamPlug password/key gen tool

hpav-keygen v0.1 - STreamPlug password/key gen tool

$ cat pw.txt
KEL8-W8D8-K8N8-VRC3

$ cat dak.txt
7d 2e fd ea af 5f 75 b7 fd 1f f1 19 7e c5 c9 81
```

Revision history

Table 14. Document revision history

Date	Revision	Changes
02-Feb-2016	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2016 STMicroelectronics – All rights reserved