Users Guide

SNAPtoolbelt

351 Electronics Blvd. SW // Huntsville, AL 35824 // (877) 982-7888 // synapsewireless.com

CONTENTS

SNAPtoolbelt is a collection of scriptable command line tools for interacting with devices in a **SNAP** network.

CHAPTER

ONE

RELEASE NOTES

CHAPTER

TWO

INSTALLATION

The binary name for **SNAPtoolbelt** is *toolbelt*.

License

Synapse Software Development Kit License Agreement

Precompiled binaries of toolbelt are available for the following platforms:

Resource	Description	MD5
Toolbelt 6.1.1 (Linux/aarch64)	Toolbelt 6.1.1 for Linux aarch64 (e.g. RPi4)	44878783224b88e2dd18b119f55a38
Toolbelt 6.1.1 (Linux/armhf)	Toolbelt 6.1.1 for Linux on armhf (e.g. E12, E20)	b7bc3940e6b996e651d2e05de46b7
Toolbelt 6.1.1 (Linux/x64)	Toolbelt 6.1.1 for Linux x64	a89b1fc82f6724e0650b5da17010c48
Toolbelt 6.1.1 (macOS/aarch64)	Toolbelt 6.1.1 for macOS aarch64 (Apple Silicon, unsigned)	352d4e40cf3b02fe6b5234c2446a6c
Toolbelt 6.1.1 (macOS/x64)	Toolbelt 6.1.1 for macOS on Intel (unsigned)	0dfb350836c68deb045ef8181b0f6a
Toolbelt 6.1.1 (Win/x32)	Toolbelt 6.1.1 for Windows x32	cbafc2581c0251831ec78b0da823be
Toolbelt 6.1.1 (Win/x64)	Toolbelt 6.1.1 for Windows x64	d3d46170da3f0e7d2d99f9a25b62b7

Extract the archive and place the (single file) executable in your path. You may need to *chmod a+x toolbelt* after extracting. On some platforms, you may also need to install libusb. Consult your package manager (e.g. apt, brew).

CHAPTER

THREE

CONVENTIONS

SNAPtoolbelt is scriptable in addition to being user controlled. If you ask it for information, it will be provided on `STDOUT`. Logs and other ancillary chatter are emitted on `STDERR`. If the command succeeds, the exit status will be zero, otherwise the exit status will be non-zero.

All input needed comes either from the configuration files, or from the invocation.

> **Note**
>
> There are no prompts for “Are you sure?”. If you tell it to erase a script, it will do so or `exit(1)` trying.

CHAPTER

FOUR

QUICK START

If you are using an **SN220 SNAPstick**, **SNAPtoolbelt** can scan for your node(s):

```
$ toolbelt config scan
```

SNAPtoolbelt will automagically add new nodes (including sniffer nodes) to your configuration.

To make one of them your default, move it to the default profile:

```
$ toolbelt config move profile SNAPstick0 default
```

CHAPTER

FIVE

COMMAND REFERENCE

SNAPtoolbelt provides access to the fundamental operation on a SNAP network.

5.1 snap config

5.1.1 snap config profile

Description

Display a profile

Usage

```
snap config profile <name> [key [value]]
```

Positional Arguments

name
: Name of profile

key
: (optional) Name of profile parameter to display/set

value
: (optional) Parameter value to set

Examples

Show the `default` profile:

```
snap config profile default
```

Show the value of the `sn220` profile's `device` parameter:

```
snap config profile sn220 device
```

Set the `sn220` profile's `device` parameter to `/dev/snap1`:

```
snap config profile sn220 device /dev/snap1
```

Set the `fast` profile`s `speed` parameter to 115200:

```
snap config profile fast speed 115200
```

5.1.2 snap config network

Description

Display a network

Usage

```
snap config network <name> [key [value]]
```

Positional Arguments

name
: Name of network

key
: (optional) Name of network parameter to display/set

value
: (optional) Parameter value to set

Examples

Show the `default` network:

```
snap config network default
```

Show the `lighting` network's `encryption_type` parameter:

```
snap config network lighting encryption_type
```

Set the `solar` network's `channel` parameter to `7`:

```
snap config network solar channel 7
```

Set the `lighting` network's `encryption_key` parameter:

```
snap config network lighting encryption_key "\x00\x01\x02\x03\x04\x05\x06\x0789abcdef"
```

5.1.3 snap config new profile

Description

Creates a new profile based on the profile type (`serial`, `tcp`) specified

Usage

```
snap config new profile (serial|tcp) <name> [options]
```

Positional Arguments

name
Name of profile to be created

Options

-f, --force Overwrite an existing profile/network

Examples

Create a new serial profile named `sn132`:

```
snap config new profile serial sn132
```

Create a new tcp profile named `e20`:

```
snap config new profile tcp e20
```

5.1.4 snap config new network

Description

Creates a new network

Usage

```
snap config new network <name> [options]
```

Positional Arguments

name
Name of network to be created

Options

-f, --force Overwrite an existing profile/network

Examples

Overwrite the `lighting` network with a new one:

```
snap config new network lighting -f
```

5.1.5 snap config list

Description

List the available profiles/networks

Usage

```
snap config list (profile|network)
```

Examples

List all profiles:

```
snap config list profile
```

5.1.6 snap config delete

Description

Delete a profile/network

Usage

```
snap config delete (profile|network) <name>
```

Positional Arguments

name
Profile/Network to be deleted

Examples

Remove the `sn220` profile:

```
snap config delete profile sn220
```

Remove the `power` network:

```
snap config delete network power
```

5.1.7 snap config copy

Description

Copy a profile or network

Usage

```
snap config copy (profile|network) <from> <to>
```

Positional Arguments

from
Profile/Network to be copied

to
Name of new profile/network

Examples

Copy the `default` profile to `snapstick`:

```
snap config copy profile default snapstick
```

Copy the `lighting` network to `default`:

```
snap config copy network lighting default
```

5.1.8 snap config move

Description

Move/rename a profile or network

Usage

```
snap config move (profile|network) <from> <to>
```

Positional Arguments

from
: Profile/Network to be moved/renamed

to
: New name of profile/network

Examples

Moves the `sn220` profile to the `default` (overriding the current default):

```
snap config move profile sn220 default
```

Move the `solar` network to `testbed`:

```
snap config move network solar testbed
```

5.1.9 snap config scan

Description

Scans serial ports for SNAP nodes and creates profiles for any that don't already exist in your configuration

Usage

```
snap config scan [options]
```

Options

-d, --dry-run	Just scan without creating profiles
-f, --force	Add nodes that have read errors

Examples

Scan for all nodes:

```
snap config scan -f
```

5.1.10 snap config node-cache show

Description

Lists the entries currently in the Node Cache (nodes which will be available when auto-completing node addresses)

Usage

```
snap config node-cache show
```

Examples

Show the Node Cache:

```
snap config node-cache show
```

5.1.11 snap config node-cache clear

Description

Removes the specified entries from the Node Cache

Usage

```
snap config node-cache clear (all|<target>)
```

Examples

Clear the entire cache:

```
snap config node-cache clear all
```

Remove 112233 from the cache:

```
snap config node-cache clear 112233
```

5.1.12 snap config alias

Description

List or define an alias to a node address. Aliases cannot contain spaces, and are case-insensitive: UUT1, uut1, and UuT1 are identical aliases.

Usage

```
snap config alias [<alias>[=<target>]]
```

Examples

List current aliases:

```
snap config alias
```

List single alias:

```
snap config alias UUT1
```

Alias UUT1 to 112233:

```
snap config alias UUT1=112233
```

Alternatively, you can separate the alias and the address with a space:

```
snap config alias UUT2 223344
```

5.1.13 snap config unalias

Description

Remove a previously defined alias

Usage

```
snap config unalias (<alias> | all)
```

Examples

Remove the UUT1 alias:

```
snap config unalias UUT1
```

Remove all aliases:

```
snap config unalias all
```

5.2 snap find

5.2.1 snap find

Description

Finds the target node by sweeping through channels, and optionally moves the node to a new network

> **Note**
>
> Cannot find nodes with different encryption settings from the Network/bridge

Usage

```
snap [global options] find [-h] <target(s)> [-l <list-file>] [-m <move-to-network>]
```

Positional Arguments

target(s)
: SNAP address of target node, or comma-separated list of addresses

Options

-l <list-file>, --list <list-file> File containing the list of nodes to find

-m <network>, --move <network> Moves the node to the specified network

Examples

Move the bridge node to the lighting network:

```
snap find bridge -m lighting
```

Find node 123456:

```
snap find 123456
```

5.3 snap firmware

5.3.1 snap firmware list

Description

Lists all firmware versions

Usage

```
snap [global options] firmware list [-h] [module]
```

Positional Arguments

module
(optional) Type of SNAP module

Examples

List all firmware versions:

```
snap firmware list
```

List firmware for module RF100:

```
snap firmware list RF100
```

5.4 snap network

5.4.1 snap network ping

Description

Send a multicast query to the network and await unicast replies

Usage

```
snap [global options] network ping [-h] [-c <count>] [-t <ttl>]
```

Options

-c <count>, --count <count> Number of pings to send (1 each second, default is 3)

-t <ttl>, --ttl <ttl> Multicast TTL for pings (default is 5)

Examples

Ping the network, sending 5 packets (which will take about 5 seconds), with a ttl of 10:

```
snap network ping -c 5 -t 10
```

5.4.2 snap network dm-ping

Description

Send a multicast query to the network and await directed multicast replies

Usage

```
snap [global options] network dm-ping [-h] [-c <count>] [-t <ttl>]
```

Options

-c <count>, --count <count> Number of pings to send (1 each second, default is 3)

-t <ttl>, --ttl <ttl> Multicast TTL for pings (default is 5)

Examples

Ping the network, sending 5 packets (which will take about 5 seconds):

```
snap network dm-ping -c 5
```

5.4.3 snap network intercept

Description

Use this command to listen for an intercepted node's output. This requires first using the *snap node intercept start* command. If you do not specify the `-t` option, the intercept will run until you use CTRL-C to abort it.

Usage

```
snap [global options] network intercept [-h] [-t <timeout>]
```

Options

-t <seconds>, --timeout <seconds> Number of seconds to intercept messages, default is forever

Examples

Listen for intercepted output:

```
snap network intercept
```

Listen for intercepted output for 10 seconds, then quit:

```
snap network intercept -t 10
```

See Also

- *snap node intercept start*

5.4.4 snap network listen

Description

If you don't have a real sniffer node available, you can always listen for messages that happen to come your way (multicasts, for instance). If you do not specify the `-t` option, the intercept will run until you use CTRL-C to abort it.

Usage

```
snap [global options] network listen [-h] [-t <timeout>]
```

Options

-t <seconds>, --timeout <seconds> Number of seconds to intercept messages, default is forever

Examples

Listen for messages:

```
snap network listen
```

Listen for messages for one minute, then exit:

```
snap network listen -t 60
```

5.4.5 snap network topology

Description

Gathers the topology of the network.

Usage

```
snap [global options] network topology [-h] <root node> [-d [-c]] [-s]
```

Positional Arguments

root node
SNAP address of node to start from (defaults to `bridge`).

Options

-d, --dot	Output DOT digraph format
-c, --colorize-lq	When rendering DOT format, colorize edges based on link quality. (>-30 dBm is excellent, <-80 dBm is bad)
-s, --simplify	Merge digraph into undirected graph by averaging A->B and B->A link qualities.

Examples

Gather the topology, starting from the bridge node:

```
snap network topology
```

Gather the simplified topology and ask `dot` to render it to a PNG with colorized edges:

```
snap network topology -dcs | dot -Tpng -o topology.png
```

5.5 snap node

5.5.1 snap node info

Description

Get summary information about the <target> node. By default it queries:

- Current Channel (i.e. “Where is this node now?”)
- Current Network ID
- NVParam Channel (i.e. “Where will this node go if it’s rebooted?”)
- NVParam Network ID
- Feature Bits
- Vendor settings
- Encryption Type
- Lockdown Bitmask
- Current Script Name
- Current Script CRC
- Current Firmware version
- Current Firmware category

Usage

```
snap [global options] node <target> info [-h] [summary]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its summary information:

```
snap node bridge info
```

5.5.2 snap node info device

Description

Gets device information about the <target> node:

- Address
- Feature Bits
- Device Name
- Platform
- Device Type
- Vendor Settings

Usage

```
snap [global options] node <target> info device [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its device information:

```
snap node bridge info device
```

5.5.3 snap node info firmware

Description

Gets firmware information about the <target> node:

- Core Version
- Platform
- Platform Category

Usage

```
snap [global options] node <target> info firmware [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its firmware information:

```
snap node bridge info firmware
```

5.5.4 snap node info mcast

Description

Gets mcast information about the <target> node:

- Carrier Sense
- Collision Avoidance
- Collision Detect
- CS/CD Threshold
- CSMA Settings
- Multicast Forwarded Groups
- Multicast Processed Groups
- Serial Multicast Forwarded Groups

Usage

```
snap [global options] node <target> info mcast [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its mcast information:

```
snap node bridge info mcast
```

5.5.5 snap node info network

Description

Gets network information about the <target> node:

- Channel
- Default Radio Rate
- Network ID
- Radio LQ Threshold
- Radio Unicast Retries

Usage

```
snap [global options] node <target> info network [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its network information:

```
snap node bridge info network
```

5.5.6 snap node info script

Description

Gets script information about the <target> node:

- Script CRC
- Script Name

Usage

```
snap [global options] node <target> info script [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its script information:

```
snap node bridge info script
```

5.5.7 snap node info security

Description

Gets security information about the <target> node:

- Encryption Type
- Lockdown Bitmask

> **Note**
>
> The Encryption Key is intentionally not queried or displayed. (If you can talk to the node, you already have the correct encryption key.)

Usage

```
snap [global options] node <target> info security [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its security information:

```
snap node bridge info security
```

5.5.8 snap node info stats

Description

Gets stats from the <target> node:

- Null Transmit Buffers
- Transparent Receive Buffers
- Transparent Transmit Buffers
- UART0 Receive Buffers
- UART0 Transmit Buffers
- UART1 Receive Buffers
- UART1 Receive Buffers
- Packet Serial Forwarded Unicasts
- Packet Serial Forwarded Multicasts
- Packet Serial Receive Buffers
- Packet Serial Transmit Buffers
- Radio Forwarded Unicasts
- Radio Forwarded Multicasts
- Radio Receive Buffers
- Radio Transmit Buffers

Usage

```
snap [global options] node <target> info stats [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its stats:

```
snap node bridge info stats
```

5.5.9 snap node info uart

Description

Gets UART information from the <target> node:

- Buffering Threshold
- Buffering Timeout
- Default UART
- Intercharacter Timeout
- UART0 Default Baud Rate
- UART1 Default Baud Rate

Usage

```
snap [global options] node <target> info uart [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its UART information:

```
snap node bridge info uart
```

5.5.10 snap node info all

Description

Gets all information from the <target> node:

- Device
- Firmware
- Mcast
- Mesh
- Network
- Script
- Security
- Stats
- Summary
- UART

Usage

```
snap [global options] node <target> info all [-h]
```

Positional Arguments

target
: SNAP address of target node or `bridge`

Examples

Ask the bridge node for all its information:

```
snap node bridge info all
```

5.5.11 snap node traceroute

Description

Performs a traceroute to the target node. Returns the round trip time and link quality information for each hop along the way. See the SNAP Manual for more details on the traceroute process.

Node Traceroute

Usage

```
snap node <target> traceroute
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Traceroute to `00aabb`:

```
snap node 00aabb traceroute
```

5.5.12 snap node script

Description

Queries the script name and CRC. Erases the script on the target node. Loads the <spy_file> over-the-air on the <target> node.

Usage

```
snap node <target> script (build | erase | info | upload) [options]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge for its script information:

```
snap node bridge script info
```

Erase the script on node `987654`:

```
snap node 987654 script erase
```

Upload `SnapStick.spy` on the bridge:

```
snap node bridge script upload SnapStick.spy
```

5.5.13 snap node script build

Description

Build the script for the TARGET node. Uses **spyc** if available to compile the script to a SPY file for that node's module and core. Will fall back on **SNAPbuild** if **spyc** is not on the PATH.

By default the output .spy/.map files are placed in the current working directory, use -o to override.

> **Note**
>
> This command requires **spyc** or **SNAPbuild** to be installed.

Usage

```
snap [global options] node <target> script build <script> [-h] [-d] [-I include [...
]] [-o output]
```

Positional Arguments

target
: SNAP address of target node or `bridge`

script
: The .py SNAPpy script to build.

Options

-d	Build for debug (**spyc** only)
-I <include>	Additional SNAPpy script import path (repeat to include multiple paths, **SNAPbuild** only)
-o <output_path>	Path to save the .spy and .map files

Examples

Build SnapStick.py for the bridge node:

```
snap node bridge script build SnapStick.py
```

5.5.14 snap node script erase

Description

Erase the script on the <target> node.

Usage

```
snap [global options] node <target> script erase [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Erase the script from the bridge node:

```
snap node bridge script erase
```

5.5.15 snap node script info

Description

Gets script information about the <target> node:

- Script CRC
- Script Name

Usage

```
snap [global options] node <target> script info [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its script information:

```
snap node bridge script info
```

5.5.16 snap node script upload

Description

Uploads the script to the <target> node. Uses **SNAPbuild** to compile the script to a SPY file for that node's module and core. If <script> is already a SPY file, just upload it without rebuilding.

> **Note**
>
> This command requires **SNAPbuild** to be installed.

Usage

```
snap [global options] node <target> script upload <script.[s]py> [-h] [-I include [...]]
```

Positional Arguments

target
: SNAP address of target node or `bridge`

script
: The .py SNAPpy script (or .spy SNAPpy script image) to upload to the node.

Options

-I <include>	Additional SNAPpy script import path (repeat to include multiple paths)

Examples

Upload SnapStick.py to the bridge node:

```
snap node bridge script upload SnapStick.py
```

5.5.17 snap node firmware info

Description

Gets firmware information about the <target> node:

- Core Version
- Platform
- Platform Category

Usage

```
snap [global options] node <target> firmware info [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Examples

Ask the bridge node for its firmware information:

```
snap node bridge firmware info
```

5.5.18 snap node firmware upload

Description

Load the specified firmware on the <target> module.

Usage

```
snap [global options] node <target> firmware upload [-h] (<core> | -f <sfi-file>)
```

Positional Arguments

target
SNAP address of target node or `bridge`

core
The version of **SNAPcore** to load

Options

-f <sfi-file>, --file <sfi-file> If firmware support package is not installed, an SFI file path can be provided

Examples

Load the `2.7.2` firmware on node `012345`:

```
snap node 012345 firmware upload 2.7.2
```

Load the `RF200_AES128_SnapV2.6.2.sfi` firmware on node `00aabb`:

```
snap node 00aabb firmware upload -f RF200_AES128_SnapV2.6.2.sfi
```

5.5.19 snap node energy

Description

Performs an energy scan on a node across all of its channels and returns a list of “clear percentages”. These percentages can be used to determine the least noisy channel to use for communication.

Usage

```
snap node <target> energy [options]
```

Positional Arguments

target
: SNAP address of target node or `bridge`

Options

-n <number>, --num_queries <number> Perform <number> queries and average the results

-d <delay>, --delay <delay> Wait for <delay> seconds between queries

-c <cutoff>, --cutoff <dbm> Cutoff value for the dBm floor

Examples

Ask the bridge node to scan 3 times, waiting 2 seconds between scans:

```
snap node bridge energy -n 3 -d 2
```

5.5.20 snap node reboot

Description

Ask the target node to reboot, with an optional <delay>

Usage

```
snap node <target> reboot [<delay>]
```

Positional Arguments

target
SNAP address of target node or `bridge`

delay
Optional milliseconds the node should wait between receiving the RPC and actually rebooting.

Examples

Reboot the bridge node:

```
snap node bridge reboot
```

5.5.21 snap node nvparam

Description

Either `reset` the <target> node NVParams to factory default, or query the <target> node for the value of NVParam <nvparam_id> and optioanlly set it <value>. Values are parsed like RPC arguments.

You can reference NVParams by name, see `snap help nvparam` for details.

> **Warning**
>
> The changes do not take effect until you reboot the node, see *snap node reboot*

Usage

```
snap node <target> (reset | nvparam <nvparam_id> [<value>]...)
```

Positional Arguments

target
SNAP address of target node or `bridge`

nvparam_id
The ID (integer or name) of the nvparam

value
If provided, set the nvparam to this value.

Examples

Ask the `bridge` node for the value of NVParam `128`:

```
snap node bridge nvparam 128
```

Ask node `123456` for its mesh routing max hop limit:

```
snap node 123456 nvparam mesh.routing.max-hop-limit
```

Set NVParam `110` to `foo` on node `012345`:

```
snap node 012345 nvparam 110 'foo'
```

Set the multicast forwarded groups on the `bridge` node to `5`:

```
snap node bridge nvparam mcast.forwarded-groups 5
```

Read NVParams 0-127 in a single pass:

```
snap node bridge nvparam all
```

See Also

- *snap node reboot*

5.5.22 snap node intercept

Description

Tell the <target> node to `start`/`stop` sending its STDOUT and/or STDERR to us

> **Note**
>
> Use the *snap network intercept* command to listen for the output after telling a node to start.

Usage

```
snap [global options] node <target> intercept [-h] (start [--out-only | --err-only] | stop)
```

Positional Arguments

target
: SNAP address of target node or `bridge`

Options

--out-only	Send only STDOUT.
--err-only	Send only STDERR.

Examples

Ask node `001122` to start sending us both STDOUT and STDERR:

```
snap node 001122 intercept start
```

Ask node `001122` to stop sending us output:

```
snap node 001122 intercept stop
```

See Also

- *snap network intercept*

5.5.23 snap node move

Description

Moves the <target> node to the specified <network>. You must be able to reach the node on the current network. If the <target> node is the bridge, this command also updates the "last used" network for the current profile to the <target> network.

Usage

```
snap [global options] node <target> move [-h] <network>
```

Positional Arguments

target
: SNAP address of target node or `bridge`

network
: Name of SNAP Network the node should move to.

Examples

Move the bridge to the 'lighting' network:

```
snap node bridge move lighting
```

Move node 001122 to the 'power' network:

```
snap node 001122 move power
```

5.5.24 snap node topology

Description

Asks the target node to acquire its immediate topology (census of immediate neighbors). Returns a map of edges and link qualities.

Node Topology

Usage

```
snap [global options] node <target> topology [-h] [-d [-c]] [-s]
```

Positional Arguments

target
SNAP address of target node or `bridge`

Options

-d, --dot	Output DOT digraph format
-c, --colorize-lq	When rendering DOT format, colorize edges based on link quality. (>-30 dBm is excellent, <-80 dBm is bad)
-s, --simplify	Merge digraph into undirected graph by averaging A->B and B->A link qualities.

Examples

Ask `00aabb` for its topology in simplified DOT form, with colors:

```
snap node 00aabb topology -sdc
```

5.6 snap recover

> **Note**
>
> On the E20 (and other gateways) when issuing the recovery commands you may need to use `sudo` `` `which snap` `` to provide sudo the full path to where **SNAPtoolbelt** is installed:

```
sudo `which snap` recover erase-script SM220
```

5.6.1 snap recover erase-script

Description

Erase any script on the connected module

Usage

```
snap [global options] recover erase-script [-h] <module>
```

Positional Arguments

module_type
: The type of SNAP module that is connected via the recover port

Examples

Erase any script on the connected SS200 module:

```
snap recover erase-script SS200
```

5.6.2 snap recover default-nvparam

Description

Restore the NV Params to the factory defaults on the connected module

Usage

```
snap [global options] recover default-nvparam [-h] <module>
```

Positional Arguments

module_type
: The type of SNAP module that is connected via the recover port

Examples

Factory default the NV Params on the connected SS200 module:

```
snap recover default-nvparam SS200
```

5.6.3 snap recover firmware

Description

Load the specified firmware Load the specified firmware on the connected module via the bootloader

Usage

```
snap [global options] recover firmware [-h] <module> (<core> | -f <sfi-file>)
```

Positional Arguments

module_type
: The type of SNAP module that is connected via the recover port

core_version
: The version of **SNAP** to load

Options

-f <sfi-file>, --file <sfi-file> If firmware support package is not installed, an SFI file path can be provided

Examples

Once you have pip installed the `snap-firmware-2.7.2` package you will be able to load **SNAP** version 2.7.2 on the connected SM220UF1 module:

```
snap recover firmware SM220UF1 2.7.2
```

If you have not installed the snap-firmware package, you can still load firmware by specifying an SFI file:

```
snap recover firmware SS200 2.6.2 -f ./sfifiles/RF200_AES128_SnapV2.6.2.sfi
```

5.6.4 snap recover sniffer

Description

Load the sniffer firmware on the connected module via the bootloader

Usage

```
snap [global options] recover sniffer [-h] <module> [-f <sfi-file>]
```

Positional Arguments

module_type
The type of SNAP module that is connected via the recover port

Options

-f <sfi-file>, --file <sfi-file> If firmware support package is not installed, an SFI file path can be provided

Examples

Load sniffer firmware on the connected RF220SU module:

```
snap recover sniffer RF220SU
```

Load sniffer firmware on the connected SS200 module by specifying an SFI file:

```
snap recover sniffer SS200 -f ./sfifiles/ATmega128RFA1_SnifferV1.3.0.sfi
```

5.7 snap rpc

5.7.1 snap rpc call unicast

Description

Callback Unicast RPC

Usage

```
snap [global options] rpc call (u | unicast) <target> <func> [<args>...
] [-h] [-c <callback-name>]
```

Positional Arguments

target
SNAP address of target node or `bridge`

func
Name of function to call on targets

args
A list of arguments to pass to the function (space-delimited, use quotes to pass an argument with whitespace.)

Options

-c <name>, --callback-name <name> Specify which function will be called on a response from the targets

Examples

Ask the bridge node `getInfo(5)`:

```
snap rpc call u bridge getInfo 5
```

Ask the bridge node `vmStat(6)`, expect it to return its result via `tellVmStat`:

```
snap rpc call u bridge vmStat 6 --callback-name tellVmStat
```

5.7.2 snap rpc call multicast

Description

Callback Multicast RPC

Usage

```
snap [global options] rpc call (m | multicast) <targets> <func> [<args>...
] [-h] [-g <group>] [-t <ttl>] [-c <callback-name>]
```

Positional Arguments

targets
A comma separated list of SNAP addresses

func
Name of function to call on targets

args
A list of arguments to pass to the function (space-delimited, use quotes to pass an argument with whitespace.)

Options

-g <group>, --group <group> Override the Network's default multicast group

-t <ttl>, --ttl <ttl> Override the Network's default multicast TTL

-c <name>, --callback-name <name> Specify which function will be called on a response from the targets

Examples

Broadcast a request for `getInfo(5)`, expect responses from `123456` and `789abc`:

```
snap rpc call m 123456,789abc getInfo 5
```

5.7.3 snap rpc call directed-multicast

Description

Callback Directed Multicast RPC

Usage

```
snap [global options] rpc call (d | dm | directed-multicast) <targets> <func> [<args>...
] [-h] [-g <group>] [-t <ttl>] [-d <delay>] [-c <callback-name>]
```

Positional Arguments

targets
: A comma separated list of SNAP addresses

func
: Name of function to call on targets

args
: A list of arguments to pass to the function (space-delimited, use quotes to pass an argument with whitespace.)

Options

-g <group>, --group <group> Override the Network's default multicast group

-t <ttl>, --ttl <ttl> Override the Network's default multicast TTL

-d <delay>, --delay <delay> Specify a response delay in milliseconds from the targets, default is 0

-c <name>, --callback-name <name> Specify which function will be called on a response from the targets

Examples

Directed Multicast a request for `getInfo(5)` to `123456` and `789abc`, ask them to delay their responses into 100ms windows:

```
snap rpc call dm 123456,789abc getInfo 5 --delay 100
```

5.7.4 snap rpc send unicast

Description

Send Unicast RPC

Usage

```
snap [global options] rpc send (u | unicast) <target> <func> [<args>...] [-h]
```

Positional Arguments

target
: SNAP address of target node or `bridge`

func
: Name of function to call on targets

args
: A list of arguments to pass to the function (space-delimited, use quotes to pass an argument with whitespace.)

Examples

Send `ping()` to the bridge node:

```
snap rpc send u bridge ping
```

5.7.5 snap rpc send multicast

Description

Send Multicast RPC

Usage

```
snap [global options] rpc send (m | multicast) <func> [<args>...
] [-h] [-g <group>] [-t <ttl>]
```

Positional Arguments

func
: Name of function to call on targets

args
: A list of arguments to pass to the function (space-delimited, use quotes to pass an argument with whitespace.)

Options

-g <group>, --group <group> Override the Network's default multicast group

-t <ttl>, --ttl <ttl> Override the Network's default multicast TTL

Examples

Broadcast `hunt()`:

```
snap rpc send m hunt
```

5.7.6 snap rpc send directed-multicast

Description

Send Directed Multicast RPC

Usage

```
snap [global options] rpc send (d | dm | directed-multicast) <targets> <func> [<args>...
] [-h] [-g <group>] [-t <ttl>] [-d <delay>]
```

Positional Arguments

targets
A comma separated list of SNAP addresses

func
Name of function to call on targets

args
A list of arguments to pass to the function (space-delimited, use quotes to pass an argument with whitespace.)

Options

-g <group>, --group <group> Override the Network's default multicast group

-t <ttl>, --ttl <ttl> Override the Network's default multicast TTL

-d <delay>, --delay <delay> Specify a response delay in milliseconds from the targets, default is 0

Examples

Send nodes `001122`, `334455`, and `667788` the RPC `getData(17, 'blue', 22)`:

```
snap rpc send dm 001122,334455,667788 getData 17 'blue' 22
```

Note

- Multicast Group, TTL, and DMCast Delay are all given default values in the Network configuration, but may be overridden per-command here.
- All target addresses must be in 6-character form: 0deCAF, 012345.
- The special address “bridge” can be used to specify your connection’s bridge node, even if the exact address is not known. (One way to discover your bridge node’s address is to use *snap node info*.)
- Arguments in quotes (`"test"`, `'abc'`) are parsed as strings.
- `""` and `''` are both parsed as empty strings.
- All numbers are parsed as integers: (0x10 == 16, 100 == 100)
- Booleans (`True`, `False`) are parsed as booleans
- `None` is parsed as `None` (Pythonic None, not the string `'None'`)
- Strings with unprintable characters need to use C/Python-style escape sequences: `"\xab\xcd\xef" == '\xab\xcd\xef'`

CHAPTER

SIX

MIGRATION FROM LEGACY SNAPTOOLBELT (PY2)

This page includes notable changes from **SNAPtoolbelt (Py2)**.

6.1 Output format is always JSON

SNAPtoolbelt now relies more on external tools for formatting its output.

Example unformatted output:

```
$ toolbelt node bridge info
{"current.channel":"12","current.network-id":"0x1234","device.address":
↪"00:1c:2c:00:26:06:a7:f2","device.feature-bits":"0x0d1f","device.name":null,"device.
↪platform":"SN220","device.script-crc":"0x3771","device.script-name":"temp","device.type
↪":"thing","device.vendor-config-bits":"0x4003","firmware.core":"2.8.2","firmware.cpu":
↪"ATMEGA","firmware.platform":"SM220/RF220UF1/SN220","firmware.rev":"(Apr  4 2017 /␣
↪6f4980c)","security.enable-encryption":"AES-128","security.lockdown-bitmask":null}
```

6.1.1 jq

The jq utility can - among other things - be used to pretty-print this output:

```
$ toolbelt node bridge info | jq .
{
    "current.channel": "12",
    "current.network-id": "0x1234",
    "device.address": "00:1c:2c:00:26:06:a7:f2",
    "device.feature-bits": "0x0d1f",
    "device.name": null,
    "device.platform": "SN220",
    "device.script-crc": "0x3771",
    "device.script-name": "temp",
    "device.type": "thing",
    "device.vendor-config-bits": "0x4003",
    "firmware.core": "2.8.2",
    "firmware.cpu": "ATMEGA",
    "firmware.platform": "SM220/RF220UF1/SN220",
    "firmware.rev": "(Apr  4 2017 / 6f4980c)",
    "security.enable-encryption": "AES-128",
```

(continues on next page)

(continued from previous page)

```
    "security.lockdown-bitmask": null
}
```

See https://jqlang.github.io/jq/ for more info.

6.1.2 mlr

Miller's `xtab` format is the closest to what **SNAPtoolbelt (Py2)** called `dkvp` (delimited key-value pairs).

```
$ toolbelt node bridge info | mlr --ijson --oxtab cat
current.channel            12
current.network-id         0x1234
device.address             00:1c:2c:00:26:06:a7:f2
device.feature-bits        0x0d1f
device.name                null
device.platform            SN220
device.script-crc          0x3771
device.script-name         temp
device.type                thing
device.vendor-config-bits  0x4003
firmware.core              2.8.2
firmware.cpu               ATMEGA
firmware.platform          SM220/RF220UF1/SN220
firmware.rev               (Apr  4 2017 / 6f4980c)
security.enable-encryption AES-128
security.lockdown-bitmask  null
```

See https://miller.readthedocs.io/en/latest/ for more info.

6.2 Configuration changes

6.2.1 Configuration is now a SQLite DB

This helped us fix a long-standing bug in **SNAPtoolbelt (Py2)** where file I/O glitches would sometimes result in a change to one profile or network causing the deletion of all other profiles or networks.

Use *toolbelt config import* to import the **SNAPtoolbelt (Py2)** configuration into **SNAPtoolbelt**.

6.2.2 *config scan* updates profiles w/ matching names

Since 2.10+ Core has improved the UART handling, it's not unreasonable to run w/ your bridge at 115200 instead of 38400. When you re-scan, if we find the bridge at 115200, we'll update the profile to indicate that, rather than create a new profile.

6.2.3 Logging is controlled by *RUST_LOG*

This is on the "things to fix" list, but for now, if you need more logging info, you'll need to set the `RUST_LOG` environment variable, rather than using the `-l` flag. (That flag is still recognized, but its value isn't used, because the logging library we started with doesn't support reconfiguring the log level on the fly.)

For example:

```
$ RUST_LOG=debug toolbelt node bridge ping
```

6.3 All RPCs are dmcast by default

All operations (node info, &c.) are done via dmcast RPCs.

Exceptions: You can still send unicast RPCs with `toolbelt rpc send unicast` and `toolbelt rpc call unicast`, and you can do unicast firmware and script uploads with `toolbelt node $TARGET firmware unicast-upload` and `toolbelt node $TARGET script unicast-upload`.

License

Synapse Software Development Kit License Agreement