Users Guide

SNAPtoolbelt

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	a89b1fc82f6724e0650b5da17010c4
Toolbelt 6.1.1 (macOS/aarch64)	Toolbelt 6.1.1 for macOS aarch64 (Apple Silicon, unsigned)	352d4e40cf3b02fe6b5234c2446a6
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 a serially-attached node like the **SN220 SNAPstick** or **SN132 SNAPstick**, **SNAPtoolbelt** can scan for your node(s):

```
$ toolbelt config scan -u default
```

SNAPtoolbelt will automagically add new nodes (including sniffer nodes) to your configuration. The `-u default` option tells **SNAPtoolbelt** to make the last one discovered your default.

To scan without updating a profile, omit the `-u` option:

```
$ toolbelt config scan
```

You can also move the profile later:

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

CHAPTER

FIVE

COMMAND REFERENCE

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

5.1 toolbelt config

5.1.1 toolbelt config profile

Description

Display a profile

Usage

```
toolbelt 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:

```
toolbelt config profile default
```

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

```
toolbelt config profile sn220 device
```

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

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

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

```
toolbelt config profile fast speed 115200
```

5.1.2 toolbelt config network

Description

Display a network

Usage

```
toolbelt 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:

```
toolbelt config network default
```

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

```
toolbelt config network lighting encryption_type
```

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

```
toolbelt config network solar channel 7
```

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

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

5.1.3 toolbelt config new profile

Description

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

Usage

```
toolbelt 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`:

```
toolbelt config new profile serial sn132
```

Create a new tcp profile named `e20`:

```
toolbelt config new profile tcp e20
```

5.1.4 toolbelt config new network

Description

Creates a new network

Usage

```
toolbelt config new network <name>
```

Positional Arguments

name
Name of network to be created

5.1.5 toolbelt config list

Description

List the available profiles/networks

Usage

`toolbelt config list (profile|network)`

Examples

List all profiles:

```
toolbelt config list profile
```

5.1.6 toolbelt config show

Description

Show a profile/network. (Same as `toolbelt config profile <name>` or `toolbelt config network <name>`)

Usage

`toolbelt config cat (profile|network) <name>`

Examples

Show the default profile:

```
toolbelt config cat profile default
```

5.1.7 toolbelt config delete

Description

Delete a profile/network

Usage

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

Positional Arguments

name
    Profile/Network to be deleted

Examples

Remove the `sn220` profile:

```
toolbelt config delete profile sn220
```

Remove the `power` network:

```
toolbelt config delete network power
```

5.1.8 toolbelt config copy

Description

Copy a profile or network

Usage

```
toolbelt 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`:

```
toolbelt config copy profile default snapstick
```

Copy the `lighting` network to `default`:

```
toolbelt config copy network lighting default
```

5.1.9 toolbelt config move

Description

Move/rename a profile or network

Usage

```
toolbelt 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):

```
toolbelt config move profile sn220 default
```

Move the `solar` network to `testbed`:

```
toolbelt config move network solar testbed
```

5.1.10 toolbelt config scan

Description

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

Usage

```
toolbelt config scan [options]
```

Options

-f, --force	Add nodes that have read errors
-u <profile>, --update <profile>	Update an additional profile (e.g. default)
-m <regex>, --matching <regex>	Limit search to paths matching the provided regex [default: .*]

Examples

Scan for all nodes:

```
toolbelt config scan -f
```

Scan for `/dev/tty.usbserial` nodes, and update the default profile w/ the last match found:

```
toolbelt config scan -u default -m /dev/tty.usbserial
```

5.2 toolbelt find

5.2.1 toolbelt 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

```
toolbelt [global options] find [-h] <target(s)> [-m <move-to-network>] [-t <ttl>]
```

Positional Arguments

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

Options

-m <network>, --move <network>	Moves the node to the specified network
-t <ttl>, --ttl <ttl>	Ping TTL

Examples

Move the bridge node to the lighting network:

```
toolbelt find bridge -m lighting
```

Find node `123456`:

```
toolbelt find 123456
```

5.3 toolbelt network

5.3.1 toolbelt network ping

Description

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

Usage

```
toolbelt [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:

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

5.3.2 toolbelt network expect

Description

Use this command to wait for a single RPC by function regex.

Usage

```
toolbelt [global options] network expect <func_re> [-h] [-t <timeout>]
```

Arguments

<func_re> RPC function regex

Options

-t <seconds>, --timeout <seconds> Number of seconds to wait, default is 30 seconds.

Examples

Expect a `button` rpc:

```
toolbelt network expect button
```

Expect a `evt.*` rpc within the next 5 seconds:

```
toolbelt network expect "evt.*" -t 5
```

5.3.3 toolbelt network listen

Description

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

Usage

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

Options

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

Examples

Listen for RPCs:

```
toolbelt network listen
```

Listen for RPCs for one minute, then exit:

```
toolbelt network listen -t 60
```

5.3.4 toolbelt network sniff

Description

Like `toolbelt network listen`, but not limited to RPCs. Allows you to listen for any packet that happens to come your way. If you do not specify the `-t` option, the session will run until you use CTRL-C to abort it.

Usage

```
toolbelt [global options] network sniff [-h] [-t <timeout>]
```

Options

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

Examples

Listen for packets:

```
toolbelt network sniff
```

Listen for packets for one minute, then exit:

```
toolbelt network sniff -t 60
```

5.4 toolbelt node

5.4.1 toolbelt 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

```
toolbelt [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:

```
toolbelt node bridge info
```

5.4.2 toolbelt node info device

Description

Gets device information about the <target> node:

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

Usage

```
toolbelt [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:

```
toolbelt node bridge info device
```

5.4.3 toolbelt node info firmware

Description

Gets firmware information about the <target> node:

- Core Version
- Platform
- Platform Category

Usage

```
toolbelt [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:

```
toolbelt node bridge info firmware
```

5.4.4 toolbelt 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

```
toolbelt [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:

```
toolbelt node bridge info mcast
```

5.4.5 toolbelt node info network

Description

Gets network information about the <target> node:

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

Usage

```
toolbelt [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:

```
toolbelt node bridge info network
```

5.4.6 toolbelt node info script

Description

Gets script information about the <target> node:

- Script CRC
- Script Name

Usage

```
toolbelt [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:

```
toolbelt node bridge info script
```

5.4.7 toolbelt 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

```
toolbelt [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:

```
toolbelt node bridge info security
```

5.4.8 toolbelt 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

```
toolbelt [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:

```
toolbelt node bridge info stats
```

5.4.9 toolbelt 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

```
toolbelt [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:

```
toolbelt node bridge info uart
```

5.4.10 toolbelt node info all

Description

Gets all information from the <target> node:

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

Usage

```
toolbelt [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:

```
toolbelt node bridge info all
```

5.4.11 toolbelt node script erase

Description

Erase the script on the <target> node.

Usage

```
toolbelt [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:

```
toolbelt node bridge script erase
```

5.4.12 toolbelt node script info

Description

Gets script information about the <target> node:

- Script CRC
- Script Name

Usage

```
toolbelt [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:

```
toolbelt node bridge script info
```

5.4.13 toolbelt node script upload

Description

Uploads a SPY file (**SNAPpy** image) to the <target> node. You must provide the SPY file, **SNAPtoolbelt** does not attempt to compile the script. Use **SNAPcompiler** to compile a **SNAPpy** script to a **SNAPpy** image.

Usage

```
toolbelt [global options] node <target> script upload <script.spy> [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

script
The .spy (**SNAPpy** image) to upload to the node.

Examples

Upload SnapStick.spy to the bridge node:

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

5.4.14 toolbelt node script unicast-upload

Description

Uploads a SPY file (**SNAPpy** image) to the <target> node using unicast RPCs. You must provide the SPY file, **SNAPtoolbelt** does not attempt to compile the script. Use **SNAPcompiler** to compile a **SNAPpy** script to a **SNAPpy** image.

Usage

```
toolbelt [global options] node <target> script unicast-upload <script.spy> [-h]
```

Positional Arguments

target
SNAP address of target node or `bridge`

script
The .spy (**SNAPpy** image) to upload to the node.

Examples

Upload SnapStick.spy to the bridge node:

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

5.4.15 toolbelt node firmware info

Description

Gets firmware information about the <target> node:

- Core Version
- Platform
- Platform Category

Usage

```
toolbelt [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:

```
toolbelt node bridge firmware info
```

5.4.16 toolbelt node firmware upload

Description

Load the specified firmware file on the <target> module.

Usage

```
toolbelt [global options] node <target> firmware upload [-h] <sfi-or-gbl-file>
```

Positional Arguments

target
SNAP address of target node or `bridge`

<sfi-or-gbl-file>
The **SNAPcore** firmware for the target node.

Examples

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

```
toolbelt node 00aabb firmware upload RF200_AES128_SnapV2.6.2.sfi
```

5.4.17 toolbelt 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

```
toolbelt 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:

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

5.4.18 toolbelt node reboot

Description

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

Usage

```
toolbelt 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:

```
toolbelt node bridge reboot
```

5.4.19 toolbelt node nvparam

Description

Query the <target> node for the value of NVParam <nvparam_id> and optioanlly set it <value>. Values are parsed like RPC arguments.

Warning

The changes do not take effect until you reboot the node, see *toolbelt node reboot*

Usage

```
toolbelt node <target> nvparam <nvparam_id> [<value>]...
```

Positional Arguments

target
SNAP address of target node or `bridge`

nvparam_id
The ID (integer, see nv-params) of the nvparam

value
If provided, set the nvparam to this value.

Examples

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

```
toolbelt node bridge nvparam 128
```

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

```
toolbelt node 012345 nvparam 110 'foo'
```

Read NVParams 0-127 in a single pass:

```
toolbelt node bridge nvparam all
```

See Also

- *toolbelt node reboot*

5.4.20 toolbelt 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

```
toolbelt [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:

```
toolbelt node bridge move lighting
```

Move node 001122 to the 'power' network:

```
toolbelt node 001122 move power
```

5.4.21 toolbelt 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

```
toolbelt [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:

```
toolbelt node 00aabb topology -sdc
```

5.5 toolbelt 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 toolbelt` recover erase-script SM220
```

Note

In addition to the full SNAP module names (e.g. `SM220UF1`, `RF520UF1`, `SN220`) you can also use the following module categories:

- `2xx` or `atmel` for all ATmega128RFA1-based modules
- `5xx` or `mg` for all Mighty Gecko-based modules

5.5.1 toolbelt recover reset

Description

Just reset the connected module. On the **SNAPconnect E12** and **SNAPconnect E20**, use the built-in GPIO to reset the **SNAP** module. For the **SN132 SNAPstick** and **SN220 SNAPstick**, use the appropriate FTDI commands to reset the **SNAP** module. Attempts a DTR-based reset otherwise.

Usage

```
toolbelt [global options] recover reset
```

Examples

Reset the module on the default profile:

```
toolbelt recover reset
```

5.5.2 toolbelt recover erase-script

Description

Erase any script on the connected module

Usage

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

Positional Arguments

module
The type of SNAP module

Examples

Erase any script on the connected **SN220 SNAPstick** module:

```
toolbelt recover erase-script SN220
```

5.5.3 toolbelt recover default-nvparam

Description

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

Usage

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

Positional Arguments

module
The type of SNAP module

Examples

Factory default the NV Params on the connected SS200 module:

```
toolbelt recover default-nvparam SS200
```

5.5.4 toolbelt recover unbrick

Description

Erase script and restore the NV Params to the factory defaults on the connected module.

Usage

```
toolbelt [global options] recover unbrick [-h] <module>
```

Positional Arguments

module
The type of SNAP module

Examples

Factory default the NV Params on the connected SS200 module:

```
toolbelt recover default-nvparam SS200
```

5.5.5 toolbelt recover firmware

Description

Load the specified firmware on the connected module via the bootloader

Usage

```
toolbelt [global options] recover firmware [-h] <module> <fw>
```

Positional Arguments

module
The type of SNAP module

fw
The **SNAPcore** file to load (SFI for Atmel-based, or GBL for Mighty Gecko-based).

Examples

To load **SNAPcore** 2.8.2 on an SN220:

```
toolbelt recover firmware SN220 ./sfifiles/SN220_AES128_SnapV2.8.2.sfi
```

5.5.6 toolbelt recover script

Description

Load the specified script on the connected module via the bootloader

Note

This function is currently supported only for Atmel-based modules.

Usage

```
toolbelt [global options] recover script [-h] <module> <spy>
```

Positional Arguments

module
The type of SNAP module

spy
The **SNAPpy** image (SPY file) to load.

Examples

To load the `uut.spy` **SNAPpy** image on an SN220 via the bootloader:

```
toolbelt recover script SN220 uut.spy
```

5.6 toolbelt rpc

5.6.1 toolbelt rpc call unicast

Description

Callback Unicast RPC

Usage

```
toolbelt [global options] rpc call (u | unicast) <target> <func> [<args>...
] [-h] [-c <callback-name>] [-w <wait>] [-a <attempts>]
```

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

-w <wait>, --wait <wait> Callback reply timeout/wait between attempts (ms)

-a <attempts>, --attempts <attempts> Number of attempts

Examples

Ask the bridge node `getInfo(5)`:

```
toolbelt rpc call u bridge getInfo 5
```

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

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

5.6.2 toolbelt rpc call directed-multicast

Description

Callback Directed Multicast RPC

Usage

```
toolbelt [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

-w <wait>, --wait <wait> Callback reply timeout/wait between attempts (ms)

-a <attempts>, --attempts <attempts> Number of attempts

Examples

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

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

5.6.3 toolbelt rpc send unicast

Description

Send Unicast RPC

Usage

```
toolbelt [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:

```
toolbelt rpc send u bridge ping
```

5.6.4 toolbelt rpc send multicast

Description

Send Multicast RPC

Usage

```
toolbelt [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()`:

```
toolbelt rpc send m hunt
```

5.6.5 toolbelt rpc send directed-multicast

Description

Send Directed Multicast RPC

Usage

```
toolbelt [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)`:

```
toolbelt 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 *toolbelt 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

GLOBAL OPTIONS

SNAPtoolbelt commands are nested. Global options must come after the initial `toolbelt`, but before the first subcommand. These options can be used with all commands.

```
toolbelt [global-options] subcommand ...
```

6.1 Profile

```
-p <profile>, --profile <profile>
```

Specify which profile you want to use, `default` is the default.

6.2 Network

```
-n <network>, --network <network>
```

Specify which network you want to use, `default` is the default.

6.3 Output Formats

```
-o json, --output-format json
```

The only supported output format is JSON:

```
{"current.channel": 8, "current.network-id": "0xbbbb", "device.core": "2.7.1", "device.
↪feature-bits": "0x001f", "device.platform": "SS200", "device.script-crc": "0xf50d"}
```

To transform the output into other formats, use tools like jq or Miller.

6.4 Alternative Config DB

```
--rcfile <path-to-config-db>
```

By default, **SNAPtoolbelt** uses `~/.snap/toolbelt_config.db` to store its configuration. You can specify an alternative path using the `--rcfile` option.

CHAPTER

SEVEN

MIGRATION FROM LEGACY SNAPTOOLBELT (PY2)

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

7.1 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}
```

7.1.1 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.

7.1.2 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.

7.2 Configuration changes

7.2.1 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**.

7.2.2 *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.

7.2.3 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
```

7.3 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`.

7.4 Obsolete Commands

- `snap rpc call multicast` (No advantage over `toolbelt rpc call dmcast`.)
- `snap config node-cache` (Node cache is unused.)
- `snap config alias` (Aliasing is unused.)
- Connection sharing: `snap start` / `snap stop` / `snap status` (Improvements in **SNAPtoolbelt** startup time mean connection sharing is no longer needed.)
- `snap node script build` (Use **SNAPcompiler** to build **SNAPpy** scripts into **SNAPpy** images, **SNAP-toolbelt** no longer attempts to guess the right platform/options.)

License