NAME

nmcli-examples − usage examples of nmcli

SYNOPSIS

nmcli [OPTIONS...]

DESCRIPTION

nmcli is a command−line client for NetworkManager. It allows controlling NetworkManager and reporting its status. For more information please refer to **nmcli**(1) manual page.

The purpose of this manual page is to provide you with various examples and usage scenarios of *nmcli*.

EXAMPLES

Example 1. Listing available Wi−Fi APs

```
$ nmcli device wifi list
*  SSID               MODE    CHAN  RATE       SIGNAL  BARS  SECURITY
   netdatacomm_local  Infra   6     54 Mbit/s  37      __    WEP
*  F1                 Infra   11    54 Mbit/s  98            WPA1
   LoremCorp          Infra   1     54 Mbit/s  62      _     WPA2 802.1X
   Internet           Infra   6     54 Mbit/s  29      ___   WPA1
   HPB110a.F2672A     Ad−Hoc  6     54 Mbit/s  22      ___   −−
   Jozinet            Infra   1     54 Mbit/s  19      ___   WEP
   VOIP               Infra   1     54 Mbit/s  20      ___   WEP
   MARTINA            Infra   4     54 Mbit/s  32      __    WPA2
   N24PU1             Infra   7     11 Mbit/s  22      ___   −−
   alfa               Infra   1     54 Mbit/s  67      _     WPA2
   bertnet            Infra   5     54 Mbit/s  20      ___   WPA1 WPA2
```

This command shows how to list available Wi−Fi networks (APs). You can also use *−−fields* option for displaying different columns. **nmcli −f all dev wifi list** will show all of them.

Example 2. Connect to a password−protected wifi network

```
$ nmcli device wifi connect "$SSID" password "$PASSWORD"
```

```
$ nmcli −−ask device wifi connect "$SSID"
```

Example 3. Showing general information and properties for a Wi−Fi interface

```
$ nmcli −p −f general,wifi−properties device show wlan0
===============================================================================
                           Device details (wlan0)
===============================================================================
GENERAL.DEVICE:            wlan0
GENERAL.TYPE:              wifi
GENERAL.VENDOR:            Intel Corporation
GENERAL.PRODUCT:           PRO/Wireless 5100 AGN [Shiloh] Network Connection
GENERAL.DRIVER:            iwlwifi
GENERAL.DRIVER−VERSION:    3.8.13−100.fc17.x86_64
GENERAL.FIRMWARE−VERSION: 8.83.5.1 build 33692
GENERAL.HWADDR:            00:1E:65:37:A1:D3
GENERAL.MTU:               1500
GENERAL.STATE:             100 (connected)
GENERAL.REASON:            0 (No reason given)
GENERAL.UDI:               /sys/devices/pci0000:00/0000:00:1c.1/net/wlan0
GENERAL.IP−IFACE:          wlan0
GENERAL.IS−SOFTWARE:       no
GENERAL.NM−MANAGED:        yes
GENERAL.AUTOCONNECT:      yes
GENERAL.FIRMWARE−MISSING: no
GENERAL.CONNECTION:       My Alfa WiFi
GENERAL.CON−UUID:         85194f4c−d496−4eec−bae0−d880b4cbcf26
GENERAL.CON−PATH:         /org/freedesktop/NetworkManager/ActiveConnection/
10
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
WIFI−PROPERTIES.WEP:      yes
WIFI−PROPERTIES.WPA:      yes
WIFI−PROPERTIES.WPA2:     yes
WIFI−PROPERTIES.TKIP:     yes
WIFI−PROPERTIES.CCMP:     yes
WIFI−PROPERTIES.AP:       no
WIFI−PROPERTIES.ADHOC:    yes
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
```

This command shows information about a Wi−Fi device.

Example 4. Listing NetworkManager polkit permissions

```
$ nmcli general permissions
PERMISSION                                                  VALUE
org.freedesktop.NetworkManager.enable−disable−network       yes
org.freedesktop.NetworkManager.enable−disable−wifi          yes
org.freedesktop.NetworkManager.enable−disable−wwan          yes
org.freedesktop.NetworkManager.enable−disable−wimax         yes
org.freedesktop.NetworkManager.sleep−wake                   no
org.freedesktop.NetworkManager.network−control              yes
org.freedesktop.NetworkManager.wifi.share.protected         yes
org.freedesktop.NetworkManager.wifi.share.open              yes
org.freedesktop.NetworkManager.settings.modify.system       yes
org.freedesktop.NetworkManager.settings.modify.own          yes
org.freedesktop.NetworkManager.settings.modify.hostname     auth
org.freedesktop.NetworkManager.settings.modify.global−dns   auth
org.freedesktop.NetworkManager.reload                       auth
```

This command shows configured polkit permissions for various NetworkManager operations. These permissions or actions (using polkit language) are configured by a system administrator and are not meant to be changed by users. The usual place for the polkit configuration is /usr/share/polkit−1/actions/org.freedesktop.NetworkManager.policy. *pkaction* command can display description for polkit actions.

```
  pkaction −−action−id org.freedesktop.NetworkManager.network−control −−verbose
```

More information about polkit can be found at http://www.freedesktop.org/wiki/Software/polkit.

Example 5. Listing NetworkManager log level and domains

```
$ nmcli general logging
LEVEL  DOMAINS
INFO   PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,WIFI_SCAN,IP4,IP6,A
UTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OLPC,
WIMAX,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,DBUS_PROPS,TEAM,CONCHECK,DC
B,DISPATCH
```

This command shows current NetworkManager logging status.

Example 6. Changing NetworkManager logging

```
$ nmcli g log level DEBUG domains CORE,ETHER,IP
$ nmcli g log level INFO domains DEFAULT
```

The first command makes NetworkManager log in DEBUG level, and only for CORE, ETHER and IP domains. The second command restores the default logging state. Please refer to the **NetworkManager.conf**(5) manual page for available logging levels and domains.

Example 7. Activating a VPN connection profile requiring interactive password input

```
$ nmcli −−ask con up my−vpn−con
```

This command activates a VPN connection profile enabling nmcli to interact with the user ('−−ask'): this will allow nmcli to prompt for the VPN password on the command line when the *password−flags* are set to '0x02' ('always ask', see **nm-settings**(5)). This is particularly useful for OTP based VPNs, as the user needs to be prompted for the password each time the connection is activated.

Example 8. Adding a bonding master and two slave connection profiles

```
$ nmcli con add type bond ifname mybond0 mode active−backup
$ nmcli con add type ethernet ifname eth1 master mybond0
$ nmcli con add type ethernet ifname eth2 master mybond0
```

This example demonstrates adding a bond master connection and two slaves. The first command adds a master bond connection, naming the bonding interface *mybond0* and using *active−backup* mode. The next two commands add slaves connections, both enslaved to *mybond0*. The first slave will be bound to *eth1* interface, the second to *eth2*.

Example 9. Adding a team master and two slave connection profiles

```
$ nmcli con add type team con−name Team1 ifname Team1 config team1−master−json.conf
$ nmcli con add type ethernet con−name Team1−slave1 ifname em1 master Team1
$ nmcli con add type ethernet con−name Team1−slave2 ifname em2 master Team1
```

This example demonstrates adding a team master connection profile and two slaves. It is very similar to the bonding example. The first command adds a master team profile, naming the team interface and the profile *Team1*. The team configuration for the master is read from *team1−master−json.conf* file. Later, you can change the configuration with *modify* command (**nmcli con modify Team1 team.config team1−master−another−json.conf**). The last two commands add slaves profiles, both enslaved to *Team1*. The first slave will be bound to the *em1* interface, the second to *em2*. The slaves don't specify *config* and thus *teamd* will use its default configuration. You will activate the whole setup by activating both slaves:

```
  $ nmcli con up Team1−slave1
  $ nmcli con up Team1−slave2
```

By default, the created profiles are marked for auto−activation. But if another connection has been activated on the device, the new profile won't activate automatically and you need to activate it manually.

Example 10. Adding a bridge and two slave profiles

```
$ nmcli con add type bridge con−name TowerBridge ifname TowerBridge
$ nmcli con add type ethernet con−name br−slave−1 ifname ens3 master TowerBridge
$ nmcli con add type ethernet con−name br−slave−2 ifname ens4 master TowerBridge
$ nmcli con modify TowerBridge bridge.stp no
```

This example demonstrates adding a bridge master connection and two slaves. The first command adds a master bridge connection, naming the bridge interface and the profile as *TowerBridge*. The next two commands add slaves profiles, both will be enslaved to *TowerBridge*. The first slave will be tied to *ens3* interface, the second to *ens4*. The last command will disable 802.1D STP for the TowerBridge profile.

Example 11. Adding an ethernet connection profile with manual IP configuration

```
$ nmcli con add con−name my−con−em1 ifname em1 type ethernet \
  ip4 192.168.100.100/24 gw4 192.168.100.1 ip4 1.2.3.4 ip6 abbe::cafe
$ nmcli con mod my−con−em1 ipv4.dns "8.8.8.8 8.8.4.4"
$ nmcli con mod my−con−em1 +ipv4.dns 1.2.3.4
$ nmcli con mod my−con−em1 ipv6.dns "2001:4860:4860::8888 2001:4860:4860::8844"
$ nmcli −p con show my−con−em1
```

The first command adds an Ethernet connection profile named *my−con−em1* that is bound to interface name *em1*. The profile is configured with static IP addresses. Three addresses are added, two IPv4 addresses and one IPv6. The first IP 192.168.100.100 has a prefix of 24 (netmask equivalent of 255.255.255.0). Gateway entry will become the default route if this profile is activated on em1 interface (and there is no connection with higher priority). The next two addresses do not specify a prefix, so a default prefix will be used, i.e. 32 for IPv4 and 128 for IPv6. The second, third and fourth commands modify DNS parameters of the new connection profile. The last *con show* command displays the profile so that all parameters can be reviewed.

Example 12. Convenient field values retrieval for scripting

```
$ nmcli −g ip4.address connection show my−con−eth0
192.168.1.12/24

$ nmcli −g ip4.address,ip4.dns connection show my−con−eth0
192.168.1.12/24
192.168.1.1

$ nmcli −g ip4 connection show my−con−eth0
IP4:192.168.1.12/24:192.168.1.1::192.168.1.1::
```

This example shows retrieval of ip4 connection field values via the −−get−values option. Multiple comma separated fields can be provided: they will be printed one per line. If a whole section is provided instead of a single field, the name of the section will be printed followed by all the related field values on the same line. See also −−terse, −−mode, −−fields and −−escape options in **nmcli**(1) manual page for more customized output.

Example 13. Adding an Ethernet connection and configuring SR−IOV VFs

```
$ nmcli con add type ethernet con−name EthernetPF ifname em1
$ nmcli con modify EthernetPF sriov.total−vfs 3 sriov.autoprobe−drivers false
$ nmcli con modify EthernetPF sriov.vfs '0 mac=00:11:22:33:44:55 vlans=10, 1 trust=true spoof−check=false'
$ nmcli con modify EthernetPF +sriov.vfs '2 max−tx−rate=20'
```

This example demonstrates adding an Ethernet connection for physical function (PF) *ens4* and configuring 3 SR−IOV virtual functions (VFs) on it. The first VF is configured with MAC address 00:11:22:33:44:55 and VLAN 10, the second one has the *trust* and *spoof−check* features respectively enabled and disabled. VF number 2 has a maximum transmission rate of 20Mbps. The kernel is instructed to not automatically instantiate a network interface for the VFs.

Example 14. Escaping colon characters in tabular mode

```
$ nmcli −t −f general −e yes −m tab dev show eth0
GENERAL:eth0:ethernet:Intel Corporation:82567LM Gigabit Network Connection:
e1000e:2.1.4−k:1.8−3:00\:22\:68\:15\:29\:21:1500:100 (connected):0 (No reas
on given):/sys/devices/pci0000\:00/0000\:00\:19.0/net/eth0:eth0:yes:yes:no:
ethernet−13:89cbcbc6−dc85−456c−9c8b−bd828fee3917:/org/freedesktop/NetworkMa
nager/ActiveConnection/9
```

This example shows escaping colon characters in tabular mode. It may be useful for script processing, because ':' is used as a field separator.

Example 15. nmcli usage in a NetworkManager dispatcher script to make Ethernet and Wi−Fi mutually exclusive

```
#!/bin/bash
export LC_ALL=C

enable_disable_wifi ()
{
   result=$(nmcli dev | grep "ethernet" | grep −w "connected")
   if [ −n "$result" ]; then
      nmcli radio wifi off
   else
      nmcli radio wifi on
   fi
}

if [ "$2" = "up" ]; then
   enable_disable_wifi
fi

if [ "$2" = "down" ]; then
   enable_disable_wifi
fi
```

This dispatcher script makes Wi−Fi mutually exclusive with wired networking. When a wired interface is connected, Wi−Fi will be set to airplane mode (rfkilled). When the wired interface is disconnected, Wi−Fi will be turned back on. Name this script e.g. 70−wifi−wired−exclusive.sh and put it into /etc/NetworkManager/dispatcher.d/ directory. See **NetworkManager**(8) manual page for more information about NetworkManager dispatcher scripts.

Example sessions of interactive connection editor

Example 16. Adding an ethernet connection profile in interactive editor (a)

```
$ nmcli connection edit type ethernet

===| nmcli interactive connection editor |===

Adding a new '802−3−ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802−3−ethernet (ethernet),
802−1x, ipv4, ipv6, dcb
nmcli> print
===============================================================================
                          Connection details
===============================================================================
connection.id:                    ethernet−4
connection.uuid:                  de89cdeb−a3e1−4d53−8fa0−c22546c775f4
connection.interface−name:        −−
connection.type:                  802−3−ethernet
connection.autoconnect:           yes
connection.autoconnect−priority:    0
connection.timestamp:               0
connection.read−only:               no
connection.permissions:
connection.zone:                    −−
connection.master:                  −−
connection.slave−type:              −−
connection.secondaries:
connection.gateway−ping−timeout:    0
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
802−3−ethernet.port:                −−
802−3−ethernet.speed:               0
802−3−ethernet.duplex:              −−
802−3−ethernet.auto−negotiate:      yes
802−3−ethernet.mac−address:         −−
802−3−ethernet.cloned−mac−address:  −−
802−3−ethernet.mac−address−blacklist:
802−3−ethernet.mtu:                 auto
802−3−ethernet.s390−subchannels:
802−3−ethernet.s390−nettype:        −−
802−3−ethernet.s390−options:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ipv4.method:                        auto
ipv4.dns:
ipv4.dns−search:
ipv4.addresses:
ipv4.gateway:                       −−
ipv4.routes:
ipv4.route−metric:                  −1
ipv4.ignore−auto−routes:            no
ipv4.ignore−auto−dns:               no
ipv4.dhcp−client−id:                −−
ipv4.dhcp−send−hostname:            yes
ipv4.dhcp−hostname:                 −−
ipv4.never−default:                 no
ipv4.may−fail:                      yes
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ipv6.method:                        auto
ipv6.dns:
ipv6.dns−search:
ipv6.addresses:
ipv6.gateway:                       −−
ipv6.routes:
ipv6.route−metric:                  −1
ipv6.ignore−auto−routes:            no
ipv6.ignore−auto−dns:               no
ipv6.never−default:                 no
ipv6.may−fail:                      yes
ipv6.ip6−privacy:                   −1 (unknown)
ipv6.dhcp−hostname:                 −−
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
nmcli> goto ethernet
You may edit the following properties: port, speed, duplex, auto−negotiate,
 mac−address, cloned−mac−address, mac−address−blacklist, mtu, s390−subchann
```

```
els, s390−nettype, s390−options
nmcli 802−3−ethernet> set mtu 1492
nmcli 802−3−ethernet> b
nmcli> goto ipv4.addresses
nmcli ipv4.addresses> desc

=== [addresses] ===
[NM property description]
Array of IP addresses.

[nmcli specific description]
Enter a list of IPv4 addresses formatted as:
  ip[/prefix], ip[/prefix],...
Missing prefix is regarded as prefix of 32.

Example: 192.168.1.5/24, 10.0.0.11/24

nmcli ipv4.addresses> set 192.168.1.100/24
Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes
nmcli ipv4.addresses>
nmcli ipv4.addresses> print
addresses: 192.168.1.100/24
nmcli ipv4.addresses> back
nmcli ipv4> b
nmcli> set ipv4.gateway 192.168.1.1
nmcli> verify
Verify connection: OK
nmcli> print
===============================================================================
                    Connection details
===============================================================================
connection.id:                          ethernet−4
connection.uuid:                        de89cdeb−a3e1−4d53−8fa0−c22546c775f4
connection.interface−name:              −−
connection.type:                        802−3−ethernet
connection.autoconnect:                 yes
connection.autoconnect−priority:        0
connection.timestamp:                   0
connection.read−only:                   no
connection.permissions:
connection.zone:                        −−
connection.master:                      −−
connection.slave−type:                  −−
connection.secondaries:
connection.gateway−ping−timeout:        0
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
802−3−ethernet.port:                    −−
802−3−ethernet.speed:                   0
802−3−ethernet.duplex:                  −−
802−3−ethernet.auto−negotiate:          yes
802−3−ethernet.mac−address:             −−
802−3−ethernet.cloned−mac−address:      −−
802−3−ethernet.mac−address−blacklist:
802−3−ethernet.mtu:                     1492
```

```
802−3−ethernet.s390−subchannels:
802−3−ethernet.s390−nettype:        −−
802−3−ethernet.s390−options:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ipv4.method:                        manual
ipv4.dns:
ipv4.dns−search:
ipv4.addresses:                     192.168.1.100/24
ipv4.gateway:                       192.168.1.1
ipv4.routes:
ipv4.route−metric:                  −1
ipv4.ignore−auto−routes:            no
ipv4.ignore−auto−dns:               no
ipv4.dhcp−client−id:                −−
ipv4.dhcp−send−hostname:            yes
ipv4.dhcp−hostname:                 −−
ipv4.never−default:                 no
ipv4.may−fail:                      yes
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ipv6.method:                        auto
ipv6.dns:
ipv6.dns−search:
ipv6.addresses:
ipv6.routes:
ipv6.route−metric:                  −1
ipv6.ignore−auto−routes:            no
ipv6.ignore−auto−dns:               no
ipv6.never−default:                 no
ipv6.may−fail:                      yes
ipv6.ip6−privacy:                   −1 (unknown)
ipv6.dhcp−hostname:                 −−
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
nmcli> set ipv4.dns 8.8.8.8 8.8.4.4
nmcli> print
===============================================================================
                 Connection details
===============================================================================
connection.id:                      ethernet−4
connection.uuid:                    de89cdeb−a3e1−4d53−8fa0−c22546c775f4
connection.interface−name:          −−
connection.type:                    802−3−ethernet
connection.autoconnect:             yes
connection.autoconnect−priority:    0
connection.timestamp:               0
connection.read−only:               no
connection.permissions:
connection.zone:                    −−
connection.master:                  −−
connection.slave−type:              −−
connection.secondaries:
connection.gateway−ping−timeout:    0
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
802−3−ethernet.port:                −−
802−3−ethernet.speed:               0
```

```
802−3−ethernet.duplex:              −−
802−3−ethernet.auto−negotiate:      yes
802−3−ethernet.mac−address:         −−
802−3−ethernet.cloned−mac−address:  −−
802−3−ethernet.mac−address−blacklist:
802−3−ethernet.mtu:                 1492
802−3−ethernet.s390−subchannels:
802−3−ethernet.s390−nettype:        −−
802−3−ethernet.s390−options:
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ipv4.method:                        manual
ipv4.dns:                           8.8.8.8,8.8.4.4
ipv4.dns−search:
ipv4.addresses:                     192.168.1.100/24
ipv4.gateway:                       192.168.1.1
ipv4.routes:
ipv4.route−metric:                  −1
ipv4.ignore−auto−routes:            no
ipv4.ignore−auto−dns:               no
ipv4.dhcp−client−id:                −−
ipv4.dhcp−send−hostname:            yes
ipv4.dhcp−hostname:                 −−
ipv4.never−default:                 no
ipv4.may−fail:                      yes
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ipv6.method:                        auto
ipv6.dns:
ipv6.dns−search:
ipv6.addresses:
ipv6.gateway:                       −−
ipv6.routes:
ipv6.route−metric:                  −1
ipv6.ignore−auto−routes:            no
ipv6.ignore−auto−dns:               no
ipv6.never−default:                 no
ipv6.may−fail:                      yes
ipv6.ip6−privacy:                   −1 (unknown)
ipv6.dhcp−hostname:                 −−
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
nmcli> verify
Verify connection: OK
nmcli> save
Connection 'ethernet−4' (de89cdeb−a3e1−4d53−8fa0−c22546c775f4) successfully
 saved.
nmcli> quit
```

Example session in the nmcli interactive connection editor. The scenario creates an Ethernet connection profile with static addressing (IPs and DNS).

Example 17. Bluetooth connection profiles

NetworkManger supports both connecting to NAP and DUN devices as a client. It also supports sharing the network via a NAP server.

For NAP client connections, NetworkManager automatically creates a suitable in−memory profile for paired devices if none is available. You may use that generated profile directly, but you may also modify and persist it, which will prevent to automatically re−create it. You may also create a profile from scratch.

For example, the following uses DHCP and IPv6 autoconf for address configuration:

$ **nmcli connection add type bluetooth con−name "Profile for My Bluetooth Device (NAP)" autoconnect no blueto**

For DUN connections, the user needs to configure modem settings and hence no profile gets created automatically. The modem settings depend on your device and you either need a "gsm" or a "csma" section. For example,

$ **nmcli connection add type bluetooth con−name "Profile for My Bluetooth Device (DUN)" autoconnect no blueto**

Finally, you can create a bluetooth hotspot. BlueZ implements those as a bridge device, so such profiles also have a bridge section. Also, you probably want to set IP methods as "shared", so that clients get automatic IP addressing. Note that the "shared" IPv4 method requires dnsmasq to be available.

$ **nmcli connection add type bluetooth con−name "My Bluetooth Hotspot" autoconnect no ifname btnap0 bluetoot**

SEE ALSO

nmcli(1), **NetworkManager**(8), **NetworkManager.conf**(5), **nm-settings**(5), **nm-online**(1), **nm-applet**(1), **nm-connection-editor**(1)