NAME

NetworkManager.conf − NetworkManager configuration file

SYNOPSIS

/etc/NetworkManager/NetworkManager.conf, /etc/NetworkManager/conf.d/*name*.conf, /run/NetworkManager/conf.d/*name*.conf, /usr/lib/NetworkManager/conf.d/*name*.conf, /var/lib/NetworkManager/NetworkManager−intern.conf

DESCRIPTION

NetworkManager.conf is the configuration file for NetworkManager. It is used to set up various aspects of NetworkManager's behavior. The location of the main file and configuration directories may be changed through use of the **−−config**, **−−config−dir**, **−−system−config−dir**, and **−−intern−config** argument for NetworkManager, respectively.

If a default NetworkManager.conf is provided by your distribution's packages, you should not modify it, since your changes may get overwritten by package updates. Instead, you can add additional .conf files to the /etc/NetworkManager/conf.d directory. These will be read in order, with later files overriding earlier ones. Packages might install further configuration snippets to /usr/lib/NetworkManager/conf.d. This directory is parsed first, even before NetworkManager.conf. Scripts can also put per−boot configuration into /run/NetworkManager/conf.d. This directory is parsed second, also before NetworkManager.conf. The loading of a file /run/NetworkManager/conf.d/*name*.conf can be prevented by adding a file /etc/NetworkManager/conf.d/*name*.conf. Likewise, a file /usr/lib/NetworkManager/conf.d/*name*.conf can be shadowed by putting a file of the same name to either /etc/NetworkManager/conf.d or /run/NetworkManager/conf.d.

NetworkManager can overwrite certain user configuration options via D−Bus or other internal operations. In this case it writes those changes to /var/lib/NetworkManager/NetworkManager−intern.conf. This file is not intended to be modified by the user, but it is read last and can shadow user configuration from NetworkManager.conf.

Certain settings from the configuration can be reloaded at runtime either by sending SIGHUP signal or via D−Bus' Reload call.

FILE FORMAT

The configuration file format is so−called key file (sort of ini−style format). It consists of sections (groups) of key−value pairs. Lines beginning with a '#' and blank lines are considered comments. Sections are started by a header line containing the section enclosed in '[' and ']', and ended implicitly by the start of the next section or the end of the file. Each key−value pair must be contained in a section.

For keys that take a list of devices as their value, you can specify devices by their MAC addresses or interface names, or "*" to specify all devices. See the section called “Device List Format” below.

Minimal system settings configuration file looks like this:

```
[main]
plugins=keyfile
```

As an extension to the normal keyfile format, you can also append a value to a previously−set list−valued key by doing:

```
plugins+=another−plugin
plugins−=remove−me
```

MAIN SECTION

plugins

Lists system settings plugin names separated by ','. These plugins are used to read and write system−wide connection profiles. When multiple plugins are specified, the connections are read from all listed plugins. When writing connections, the plugins will be asked to save the connection in the order listed here; if the first plugin cannot write out that connection type (or can't write out any

connections) the next plugin is tried, etc. If none of the plugins can save the connection, an error is returned to the user.

The default value and the number of available plugins is distro−specific. See the section called “PLUGINS” below for the available plugins. Note that NetworkManager's native keyfile plugin is always appended to the end of this list (if it doesn't already appear earlier in the list).

monitor−connection−files
This setting is deprecated and has no effect.

auth−polkit
Whether the system uses PolicyKit for authorization. If true, non−root requests are authorized using PolicyKit. Requests from root (user ID zero) are always granted without asking PolicyKit. If false, all requests will be allowed and PolicyKit is not used. If set to root−only PolicyKit is not used and all requests except root are denied. The default value is true.

dhcp
This key sets up what DHCP client NetworkManager will use. Allowed values are dhclient, dhcpcd, and internal. The dhclient and dhcpcd options require the indicated clients to be installed. The internal option uses a built−in DHCP client which is not currently as featureful as the external clients.

If this key is missing, it defaults to internal. It the chosen plugin is not available, clients are looked for in this order: dhclient, dhcpcd, internal.

no−auto−default
Specify devices for which NetworkManager shouldn't create default wired connection (Auto eth0). By default, NetworkManager creates a temporary wired connection for any Ethernet device that is managed and doesn't have a connection configured. List a device in this option to inhibit creating the default connection for the device. May have the special value * to apply to all devices.

When the default wired connection is deleted or saved to a new persistent connection by a plugin, the device is added to a list in the file /var/lib/NetworkManager/no−auto−default.state to prevent creating the default connection for that device again.

See the section called “Device List Format” for the syntax how to specify a device.

Example:

no−auto−default=00:22:68:5c:5d:c4,00:1e:65:ff:aa:ee
no−auto−default=eth0,eth1
no−auto−default=*

ignore−carrier
This setting is deprecated for the per−device setting ignore−carrier which overwrites this setting if specified (See *ignore−carrier*). Otherwise, it is a list of matches to specify for which device carrier should be ignored. See the section called “Device List Format” for the syntax how to specify a device. Note that master types like bond, bridge, and team ignore carrier by default. You can however revert that default using the "except:" specifier (or better, use the per−device setting instead of the deprecated setting).

assume−ipv6ll−only
Specify devices for which NetworkManager will try to generate a connection based on initial configuration when the device only has an IPv6 link−local address.

See the section called “Device List Format” for the syntax how to specify a device.

configure−and−quit
When set to 'true', NetworkManager quits after performing initial network configuration but spawns

small helpers to preserve DHCP leases and IPv6 addresses. This is useful in environments where network setup is more or less static or it is desirable to save process time but still handle some dynamic configurations. When this option is true, network configuration for Wi−Fi, WWAN, Bluetooth, ADSL, and PPPoE interfaces cannot be preserved due to their use of external services, and these devices will be deconfigured when NetworkManager quits even though other interface's configuration may be preserved. Also, to preserve DHCP addresses the 'dhcp' option must be set to 'internal'. The default value of the 'configure−and−quit' option is 'false', meaning that NetworkManager will continue running after initial network configuration and continue responding to system and hardware events, D−Bus requests, and user commands.

hostname−mode

Set the management mode of the hostname. This parameter will affect only the transient hostname. If a valid static hostname is set, NetworkManager will skip the update of the hostname despite the value of this option. An hostname empty or equal to 'localhost', 'localhost6', 'localhost.localdomain' or 'localhost6.localdomain' is considered invalid.

default: NetworkManager will update the hostname with the one provided via DHCP on the main connection (the one with a default route). If not present, the hostname will be updated to the last one set outside NetworkManager. If it is not valid, NetworkManager will try to recover the hostname from the reverse lookup of the IP address of the main connection. If this fails too, the hostname will be set to 'localhost.localdomain'.

dhcp: NetworkManager will update the transient hostname only with information coming from DHCP. No fallback nor reverse lookup will be performed, but when the dhcp connection providing the hostname is deactivated, the hostname is reset to the last hostname set outside NetworkManager or 'localhost' if none valid is there.

none: NetworkManager will not manage the transient hostname and will never set it.

dns

Set the DNS processing mode.

If the key is unspecified, default is used, unless /etc/resolv.conf is a symlink to /run/systemd/resolve/stub−resolv.conf, /run/systemd/resolve/resolv.conf, /lib/systemd/resolv.conf or /usr/lib/systemd/resolv.conf. In that case, systemd−resolved is chosen automatically.

default: NetworkManager will update /etc/resolv.conf to reflect the nameservers provided by currently active connections.

dnsmasq: NetworkManager will run dnsmasq as a local caching nameserver, using "Conditional Forwarding" if you are connected to a VPN, and then update resolv.conf to point to the local nameserver. It is possible to pass custom options to the dnsmasq instance by adding them to files in the "/etc/NetworkManager/dnsmasq.d/" directory. Note that when multiple upstream servers are available, dnsmasq will initially contact them in parallel and then use the fastest to respond, probing again other servers after some time. This behavior can be modified passing the 'all−servers' or 'strict−order' options to dnsmasq (see the manual page for more details).

systemd−resolved: NetworkManager will push the DNS configuration to systemd−resolved

unbound: NetworkManager will talk to unbound and dnssec−triggerd, using "Conditional Forwarding" with DNSSEC support. /etc/resolv.conf will be managed by dnssec−trigger daemon.

none: NetworkManager will not modify resolv.conf. This implies rc−manager unmanaged

Note that the plugins dnsmasq, systemd−resolved and unbound are caching local nameservers. Hence, when NetworkManager writes /run/NetworkManager/resolv.conf and /etc/resolv.conf (according to

rc−manager setting below), the name server there will be localhost only. NetworkManager also writes a file /run/NetworkManager/no−stub−resolv.conf that contains the original name servers pushed to the DNS plugin.

When using dnsmasq and systemd−resolved per−connection added dns servers will always be queried using the device the connection has been activated on.

rc−manager
Set the resolv.conf management mode. The default value depends on NetworkManager build options, and this version of NetworkManager was build with a default of "symlink". Regardless of this setting, NetworkManager will always write resolv.conf to its runtime state directory /run/NetworkManager/resolv.conf.

symlink: If /etc/resolv.conf is a regular file, NetworkManager will replace the file on update. If /etc/resolv.conf is instead a symlink, NetworkManager will leave it alone. Unless the symlink points to the internal file /run/NetworkManager/resolv.conf, in which case the symlink will be updated to emit an inotify notification. This allows the user to conveniently instruct NetworkManager not to manage /etc/resolv.conf by replacing it with a symlink.

file: NetworkManager will write /etc/resolv.conf as file. If it finds a symlink to an existing target, it will follow the symlink and update the target instead. In no case will an existing symlink be replaced by a file. Note that older versions of NetworkManager behaved differently and would replace dangling symlinks with a plain file.

resolvconf: NetworkManager will run resolvconf to update the DNS configuration.

netconfig: NetworkManager will run netconfig to update the DNS configuration.

unmanaged: don't touch /etc/resolv.conf.

none: deprecated alias for symlink.

systemd−resolved
Send the connection DNS configuration to systemd−resolved. Defaults to "true".

Note that this setting is complementary to the *dns* setting. You can keep this enabled while using *dns* set to another DNS plugin alongside systemd−resolved, or *dns* set to systemd−resolved to configure the system resolver to use systemd−resolved.

If systemd−resolved is enabled, the connectivity check resolves the hostname per−device.

debug
Comma separated list of options to aid debugging. This value will be combined with the environment variable NM_DEBUG. Currently the following values are supported:

RLIMIT_CORE: set ulimit −c unlimited to write out core dumps. Beware, that a core dump can contain sensitive information such as passwords or configuration settings.

fatal−warnings: set g_log_set_always_fatal() to core dump on warning messages from glib. This is equivalent to the −−g−fatal−warnings command line option.

autoconnect−retries−default
The number of times a connection activation should be automatically tried before switching to another one. This value applies only to connections that can auto−connect and have a connection.autoconnect−retries property set to −1. If not specified, connections will be tried 4 times. Setting this value to 1 means to try activation once, without retry.

slaves−order
This key specifies in which order slave connections are auto−activated on boot or when the master activates them. Allowed values are name (order connection by interface name, the default), or index (order slaves by their kernel index).

KEYFILE SECTION

This section contains keyfile−plugin−specific options, and is normally only used when you are not using any other distro−specific plugin.

hostname
This key is deprecated and has no effect since the hostname is now stored in /etc/hostname or other system configuration files according to build options.

path
The location where keyfiles are read and stored. This defaults to "/etc/NetworkManager/system−connections".

unmanaged−devices
Set devices that should be ignored by NetworkManager.

See the section called “Device List Format” for the syntax how to specify a device.

Example:

unmanaged−devices=interface−name:em4
unmanaged−devices=mac:00:22:68:1c:59:b1;mac:00:1E:65:30:D1:C4;interface−name:eth2

IFUPDOWN SECTION

This section contains ifupdown−specific options and thus only has effect when using the ifupdown plugin.

managed
If set to true, then interfaces listed in /etc/network/interfaces are managed by NetworkManager. If set to false, then any interface listed in /etc/network/interfaces will be ignored by NetworkManager. Remember that NetworkManager controls the default route, so because the interface is ignored, NetworkManager may assign the default route to some other interface.

The default value is false.

LOGGING SECTION

This section controls NetworkManager's logging. Any settings here are overridden by the **−−log−level** and **−−log−domains** command−line options.

level
The default logging verbosity level. One of OFF, ERR, WARN, INFO, DEBUG, TRACE. The ERR level logs only critical errors. WARN logs warnings that may reflect operation. INFO logs various informational messages that are useful for tracking state and operations. DEBUG enables verbose logging for debugging purposes. TRACE enables even more verbose logging then DEBUG level. Subsequent levels also log all messages from earlier levels; thus setting the log level to INFO also logs error and warning messages.

domains
The following log domains are available: PLATFORM, RFKILL, ETHER, WIFI, BT, MB, DHCP4, DHCP6, PPP, WIFI_SCAN, IP4, IP6, AUTOIP4, DNS, VPN, SHARING, SUPPLICANT, AGENTS, SETTINGS, SUSPEND, CORE, DEVICE, OLPC, WIMAX, INFINIBAND, FIREWALL, ADSL, BOND, VLAN, BRIDGE, DBUS_PROPS, TEAM, CONCHECK, DCB, DISPATCH, AUDIT, SYSTEMD, VPN_PLUGIN, PROXY.

In addition, these special domains can be used: NONE, ALL, DEFAULT, DHCP, IP.

You can specify per−domain log level overrides by adding a colon and a log level to any domain. E.g., "WIFI:DEBUG,WIFI_SCAN:OFF".

Domain descriptions:

PLATFORM : OS (platform) operations
RFKILL : RFKill subsystem operations
ETHER : Ethernet device operations
WIFI : Wi−Fi device operations
BT : Bluetooth operations
MB : Mobile broadband operations
DHCP4 : DHCP for IPv4
DHCP6 : DHCP for IPv6
PPP : Point−to−point protocol operations
WIFI_SCAN : Wi−Fi scanning operations
IP4 : IPv4−related operations
IP6 : IPv6−related operations
AUTOIP4 : AutoIP operations
DNS : Domain Name System related operations
VPN : Virtual Private Network connections and operations
SHARING : Connection sharing. With TRACE level log queries for dnsmasq instance
SUPPLICANT : WPA supplicant related operations
AGENTS : Secret agents operations and communication
SETTINGS : Settings/config service operations
SUSPEND : Suspend/resume
CORE : Core daemon and policy operations
DEVICE : Activation and general interface operations
OLPC : OLPC Mesh device operations
WIMAX : WiMAX device operations
INFINIBAND : InfiniBand device operations
FIREWALL : FirewallD related operations
ADSL : ADSL device operations
BOND : Bonding operations
VLAN : VLAN operations
BRIDGE : Bridging operations
DBUS_PROPS : D−Bus property changes
TEAM : Teaming operations
CONCHECK : Connectivity check
DCB : Data Center Bridging (DCB) operations
DISPATCH : Dispatcher scripts
AUDIT : Audit records
SYSTEMD : Messages from internal libsystemd
VPN_PLUGIN : logging messages from VPN plugins
PROXY : logging messages for proxy handling

NONE : when given by itself logging is disabled
ALL : all log domains
DEFAULT : default log domains
DHCP : shortcut for "DHCP4,DHCP6"
IP : shortcut for "IP4,IP6"

HW : deprecated alias for "PLATFORM"

In general, the logfile should not contain passwords or private data. However, you are always advised to check the file before posting it online or attaching to a bug report. VPN_PLUGIN is special as it might reveal private information of the VPN plugins with verbose levels. Therefore this domain will be excluded when setting ALL or DEFAULT to more verbose levels then INFO.

backend
The logging backend. Supported values are "syslog" and "journal". When NetworkManager is started with "−−debug" in addition all messages will be printed to stderr. If unspecified, the default is "journal".

audit
Whether the audit records are delivered to auditd, the audit daemon. If false, audit records will be sent only to the NetworkManager logging system. If set to true, they will be also sent to auditd. The default value is true.

CONNECTION SECTION

Specify default values for connections.

Example:

```
[connection]
ipv6.ip6−privacy=0
```

Supported Properties

Not all properties can be overwritten, only the following properties are supported to have their default values configured (see **nm-settings**(5) for details). A default value is only consulted if the corresponding per−connection value explicitly allows for that.

802−1x.auth−timeout

cdma.mtu

connection.auth−retries
If left unspecified, the default value is 3 tries before failing the connection.

connection.autoconnect−slaves

connection.lldp

connection.llmnr
If unspecified, the ultimate default values depends on the DNS plugin. With systemd−resolved the default currently is "yes" (2) and for all other plugins "no" (0).

connection.mdns
If unspecified, the ultimate default values depends on the DNS plugin. With systemd−resolved the default currently is "no" (0) and for all other plugins also "no" (0).

connection.stable−id

ethernet.cloned−mac−address
If left unspecified, it defaults to "preserve".

ethernet.generate−mac−address−mask

ethernet.mtu
If configured explicitly to 0, the MTU is not reconfigured during device activation unless it is required due to IPv6 constraints. If left unspecified, a DHCP/IPv6 SLAAC provided value is used or the MTU is not reconfigured during activation.

ethernet.wake−on−lan

gsm.mtu

infiniband.mtu
If configured explicitly to 0, the MTU is not reconfigured during device activation unless it is required due to IPv6 constraints. If left unspecified, a DHCP/IPv6 SLAAC provided value is used or the MTU is left unspecified on activation.

ip−tunnel.mtu
If configured explicitly to 0, the MTU is not reconfigured during device activation unless it is required due to IPv6 constraints. If left unspecified, a DHCP/IPv6 SLAAC provided value is used or a default of 1500.

ipv4.dad−timeout

ipv4.dhcp−client−id

ipv4.dhcp−iaid
If left unspecified, it defaults to "ifname".

ipv4.dhcp−hostname−flags
If left unspecified, the value 3 (fqdn−encoded,fqdn−serv−update) is used.

ipv4.dhcp−timeout
If left unspecified, the default value for the interface type is used.

ipv4.dns−priority
If unspecified or zero, use 50 for VPN profiles and 100 for other profiles.

ipv4.route−metric

ipv4.route−table
If left unspecified, routes are only added to the main table. Note that this is different from explicitly selecting the main table 254, because of how NetworkManager removes extraneous routes from the tables.

ipv6.ra−timeout
If left unspecified, the default value depends on the sysctl solicitation settings.

ipv6.dhcp−duid
If left unspecified, it defaults to "lease".

ipv6.dhcp−iaid
If left unspecified, it defaults to "ifname".

ipv6.dhcp−hostname−flags
If left unspecified, the value 1 (fqdn−serv−update) is used.

ipv6.dhcp−timeout
If left unspecified, the default value for the interface type is used.

ipv6.dns−priority
If unspecified or zero, use 50 for VPN profiles and 100 for other profiles.

ipv6.ip6−privacy
If ipv6.ip6−privacy is unset, use the content of "/proc/sys/net/ipv6/conf/default/use_tempaddr" as last fallback.

ipv6.route−metric

ipv6.route−table
If left unspecified, routes are only added to the main table. Note that this is different from explicitly selecting the main table 254, because of how NetworkManager removes extraneous routes from the tables.

sriov.autoprobe−drivers
If left unspecified, drivers are autoprobed when the SR−IOV VF gets created.

vpn.timeout
If left unspecified, default value of 60 seconds is used.

wifi.cloned−mac−address
If left unspecified, it defaults to "preserve".

wifi.generate−mac−address−mask

wifi.mac−address−randomization
If left unspecified, MAC address randomization is disabled. This setting is deprecated for wifi.cloned−mac−address.

wifi.mtu
If configured explicitly to 0, the MTU is not reconfigured during device activation unless it is required due to IPv6 constraints. If left unspecified, a DHCP/IPv6 SLAAC provided value is used or a default of 1500.

wifi.powersave
If left unspecified, the default value "ignore" will be used.

wifi−sec.pmf
If left unspecified, the default value "optional" will be used.

wifi−sec.fils
If left unspecified, the default value "optional" will be used.

wifi.wake−on−wlan

wireguard.mtu

Sections

You can configure multiple connection sections, by having different sections with a name that all start with "connection". Example:

```
[connection]
ipv6.ip6−privacy=0
connection.autoconnect−slaves=1
vpn.timeout=120

[connection−wifi−wlan0]
match−device=interface−name:wlan0
ipv4.route−metric=50

[connection−wifi−other]
match−device=type:wifi
ipv4.route−metric=55
ipv6.ip6−privacy=1
```

The sections within one file are considered in order of appearance, with the exception that the [connection] section is always considered last. In the example above, this order is [connection−wifi−wlan0], [connection−wlan−other], and [connection]. When checking for a default configuration value, the sections are searched until the requested value is found. In the example above, "ipv4.route−metric" for wlan0 interface is set to 50, and for all other Wi−Fi typed interfaces to 55. Also, Wi−Fi devices would have IPv6 private addresses enabled by default, but other devices would have it disabled. Note that also "wlan0" gets "ipv6.ip6−privacy=1", because although the section "[connection−wifi−wlan0]" matches the device, it does not contain that property and the search continues.

When having different sections in multiple files, sections from files that are read later have higher priority. So within one file the priority of the sections is top−to−bottom. Across multiple files later definitions take precedence.

The following properties further control how a connection section applies.

match−device
An optional device spec that restricts when the section applies. See the section called “Device List Format” for the possible values.

stop−match
An optional boolean value which defaults to no. If the section matches (based on match−device),

further sections will not be considered even if the property in question is not present. In the example above, if [connection−wifi−wlan0] would have stop−match set to yes, the device wlan0 would have ipv6.ip6−privacy property unspecified. That is, the search for the property would not continue in the connection sections [connection−wifi−other] or [connection].

DEVICE SECTION

Contains per−device persistent configuration.

Example:

```
[device]
match−device=interface−name:eth3
managed=1
```

Supported Properties

The following properties can be configured per−device.

managed

Whether the device is managed or not. A device can be marked as managed via udev rules (ENV{NM_UNMANAGED}), or via setting plugins (keyfile.unmanaged−devices). This is yet another way. Note that this configuration can be overruled at runtime via D−Bus. Also, it has higher priority then udev rules.

carrier−wait−timeout

Specify the timeout for waiting for carrier in milliseconds. When the device loses carrier, NetworkManager does not react immediately. Instead, it waits for this timeout before considering the link lost. Also, on startup, NetworkManager considers the device as busy for this time, as long as the device has no carrier. This delays startup−complete signal and NetworkManager−wait−online. Configuring this too high means to block NetworkManager−wait−online longer then necessary. Configuring it too low, means that NetworkManager will declare startup−complete, although carrier is about to come and auto−activation to kick in. The default is 5000 milliseconds.

ignore−carrier

Specify devices for which NetworkManager will (partially) ignore the carrier state. Normally, for device types that support carrier−detect, such as Ethernet and InfiniBand, NetworkManager will only allow a connection to be activated on the device if carrier is present (ie, a cable is plugged in), and it will deactivate the device if carrier drops for more than a few seconds.

A device with carrier ignored will allow activating connections on that device even when it does not have carrier, provided that the connection uses only statically−configured IP addresses. Additionally, it will allow any active connection (whether static or dynamic) to remain active on the device when carrier is lost.

Note that the "carrier" property of NMDevices and device D−Bus interfaces will still reflect the actual device state; it's just that NetworkManager will not make use of that information.

Master types like bond, bridge and team ignore carrier by default, while other device types react on carrier changes by default.

This setting overwrites the deprecated main.ignore−carrier setting above.

wifi.scan−rand−mac−address

Configures MAC address randomization of a Wi−Fi device during scanning. This defaults to yes in which case a random, locally−administered MAC address will be used. The setting wifi.scan−generate−mac−address−mask allows to influence the generated MAC address to use certain vendor OUIs. If disabled, the MAC address during scanning is left unchanged to whatever is configured. For the configured MAC address while the device is associated, see instead the per−connection setting wifi.cloned−mac−address.

wifi.backend
: Specify the Wi−Fi backend used for the device. Currently supported are wpa_supplicant and iwd (experimental).

wifi.scan−generate−mac−address−mask
: Like the per−connection settings ethernet.generate−mac−address−mask and wifi.generate−mac−address−mask, this allows to configure the generated MAC addresses during scanning. See **nm-settings**(5) for details.

sriov−num−vfs
: Specify the number of virtual functions (VF) to enable for a PCI physical device that supports single−root I/O virtualization (SR−IOV).

Sections

The [device] section works the same as the [connection] section. That is, multiple sections that all start with the prefix "device" can be specified. The settings "match−device" and "stop−match" are available to match a device section on a device. The order of multiple sections is also top−down within the file and later files overwrite previous settings. See “Sections” under the section called “CONNECTION SECTION” for details.

CONNECTIVITY SECTION

This section controls NetworkManager's optional connectivity checking functionality. This allows NetworkManager to detect whether or not the system can actually access the internet or whether it is behind a captive portal.

Connectivity checking serves two purposes. For one, it exposes a connectivity state on D−Bus, which other applications may use. For example, Gnome's portal helper uses this as signal to show a captive portal login page. The other use is that default−route of devices without global connectivity get a penalty of +20000 to the route−metric. This has the purpose to give a better default−route to devices that have global connectivity. For example, when being connected to WWAN and to a Wi−Fi network which is behind a captive portal, WWAN still gets preferred until login.

Note that your distribution might set /proc/sys/net/ipv4/conf/*/rp_filter to strict filtering. That works badly with per−device connectivity checking, which uses SO_BINDDEVICE to send requests on all devices. A strict rp_filter setting will reject any response and the connectivity check on all but the best route will fail.

enabled
: Whether connectivity check is enabled. Note that to enable connectivity check, a valid uri must also be configured. The value defaults to true, but since the uri is unset by default, connectivity check may be disabled. The main purpose of this option is to have a single flag to disable connectivity check. Note that this setting can also be set via D−Bus API at runtime. In that case, the value gets stored in /var/lib/NetworkManager/NetworkManager−intern.conf file.

uri
: The URI of a web page to periodically request when connectivity is being checked. This page should return the header "X−NetworkManager−Status" with a value of "online". Alternatively, its body content should be set to "NetworkManager is online". The body content check can be controlled by the response option. If this option is blank or missing, connectivity checking is disabled.

interval
: Specified in seconds; controls how often connectivity is checked when a network connection exists. If set to 0 connectivity checking is disabled. If missing, the default is 300 seconds.

response
: If set, controls what body content NetworkManager checks for when requesting the URI for connectivity checking. Note that this only compares that the HTTP response starts with the specifid text, it does not compare the exact string. This behavior might change in the future, so avoid relying on it. If missing, the response defaults to "NetworkManager is online". If set to empty, the HTTP server is expected to answer with status code 204 or send no data.

GLOBAL−DNS SECTION

This section specifies global DNS settings that override connection−specific configuration.

searches
: A list of search domains to be used during hostname lookup.

options
: A list of options to be passed to the hostname resolver.

GLOBAL−DNS−DOMAIN SECTIONS

Sections with a name starting with the "global−dns−domain−" prefix allow to define global DNS configuration for specific domains. The part of section name after "global−dns−domain−" specifies the domain name a section applies to. More specific domains have the precedence over less specific ones and the default domain is represented by the wildcard "*". A default domain section is mandatory.

servers
: A list of addresses of DNS servers to be used for the given domain.

options
: A list of domain−specific DNS options. Not used at the moment.

.CONFIG SECTIONS

This is a special section that contains options which apply to the configuration file that contains the option.

enable
: Defaults to "true". If "false", the configuration file will be skipped during loading. Note that the main configuration file NetworkManager.conf cannot be disabled.

```
# always skip loading the config file
[.config]
enable=false
```

You can also match against the version of NetworkManager. For example the following are valid configurations:

```
# only load on version 1.0.6
[.config]
enable=nm−version:1.0.6

# load on all versions 1.0.x, but not 1.2.x
[.config]
enable=nm−version:1.0

# only load on versions >= 1.1.6. This does not match
# with version 1.2.0 or 1.4.4. Only the last digit is considered.
[.config]
enable=nm−version−min:1.1.6

# only load on versions >= 1.2. Contrary to the previous
# example, this also matches with 1.2.0, 1.2.10, 1.4.4, etc.
[.config]
enable=nm−version−min:1.2

# Match against the maximum allowed version. The example matches
# versions 1.2.0, 1.2.2, 1.2.4. Again, only the last version digit
# is allowed to be smaller. So this would not match match on 1.1.10.
[.config]
enable=nm−version−max:1.2.6
```

You can also match against the value of the environment variable NM_CONFIG_ENABLE_TAG, like:

```
# always skip loading the file when running NetworkManager with
# environment variable "NM_CONFIG_ENABLE_TAG=TAG1"
[.config]
enable=env:TAG1
```

More then one match can be specified. The configuration will be enabled if one of the predicates matches ("or"). The special prefix "except:" can be used to negate the match. Note that if one except−predicate matches, the entire configuration will be disabled. In other words, a except predicate always wins over other predicates. If the setting only consists of "except:" matches and none of the negative conditions are satisfied, the configuration is still enabled.

```
# enable the configuration either when the environment variable
# is present or the version is at least 1.2.0.
[.config]
enable=env:TAG2,nm−version−min:1.2
```

```
# enable the configuration for version >= 1.2.0, but disable
# it when the environment variable is set to "TAG3"
[.config]
enable=except:env:TAG3,nm−version−min:1.2
```

```
# enable the configuration on >= 1.3, >= 1.2.6, and >= 1.0.16.
# Useful if a certain feature is only present since those releases.
[.config]
enable=nm−version−min:1.3,nm−version−min:1.2.6,nm−version−min:1.0.16
```

PLUGINS

Settings plugins for reading and writing connection profiles. The number of available plugins is distribution specific.

keyfile

The keyfile plugin is the generic plugin that supports all the connection types and capabilities that NetworkManager has. It writes files out in an .ini−style format in /etc/NetworkManager/system−connections. See **nm-settings-keyfile**(5) for details about the file format.

The stored connection file may contain passwords, secrets and private keys in plain text, so it will be made readable only to root, and the plugin will ignore files that are readable or writable by any user or group other than root. See "Secret flag types" in **nm-settings**(5) for how to avoid storing passwords in plain text.

This plugin is always active, and will automatically be used to store any connections that aren't supported by any other active plugin.

ifcfg−rh

This plugin is used on the Fedora and Red Hat Enterprise Linux distributions to read and write configuration from the standard /etc/sysconfig/network−scripts/ifcfg−* files. It currently supports reading Ethernet, Wi−Fi, InfiniBand, VLAN, Bond, Bridge, and Team connections. Enabling ifcfg−rh implicitly enables ibft plugin, if it is available. This can be disabled by adding no−ibft. See /usr/share/doc/initscripts/sysconfig.txt and **nm-settings-ifcfg-rh**(5) for more information about the ifcfg file format.

ifupdown

This plugin is used on the Debian and Ubuntu distributions, and reads Ethernet and Wi−Fi connections

from /etc/network/interfaces.

This plugin is read−only; any connections (of any type) added from within NetworkManager when you are using this plugin will be saved using the keyfile plugin instead.

ibft, *no−ibft*
These plugins are deprecated and their selection has no effect. This is now handled by nm−initrd−generator.

ifcfg−suse, *ifnet*
These plugins are deprecated and their selection has no effect. The keyfile plugin should be used instead.

APPENDIX

Device List Format

The configuration options main.no−auto−default, main.ignore−carrier, keyfile.unmanaged−devices, connection*.match−device and device*.match−device select devices based on a list of matchings. Devices can be specified using the following format:

*
Matches every device.

IFNAME
Case sensitive match of interface name of the device. Globbing is not supported.

HWADDR
Match the permanent MAC address of the device. Globbing is not supported

interface−name:IFNAME, interface−name:~IFNAME
Case sensitive match of interface name of the device. Simple globbing is supported with * and ?. Ranges and escaping is not supported.

interface−name:=IFNAME
Case sensitive match of interface name of the device. Globbing is disabled and IFNAME is taken literally.

mac:HWADDR
Match the permanent MAC address of the device. Globbing is not supported

s390−subchannels:HWADDR
Match the device based on the subchannel address. Globbing is not supported

type:TYPE
Match the device type. Valid type names are as reported by "nmcli −f GENERAL.TYPE device show". Globbing is not supported.

driver:DRIVER
Match the device driver as reported by "nmcli −f GENERAL.DRIVER,GENERAL.DRIVER−VERSION device show". "DRIVER" must match the driver name exactly and does not support globbing. Optionally, a driver version may be specified separated by '/'. Globbing is supported for the version.

dhcp−plugin:DHCP
Match the configured DHCP plugin "main.dhcp".

except:SPEC
Negative match of a device. SPEC must be explicitly qualified with a prefix such as interface−name:. A negative match has higher priority then the positive matches above.

If there is a list consisting only of negative matches, the behavior is the same as if there is also match−all. That means, if none of all the negative matches is satisfied, the overall result is still a positive match. That means, "except:interface−name:eth0" is the same as "*,except:interface−name:eth0".

SPEC[,;]SPEC

Multiple specs can be concatenated with commas or semicolons. The order does not matter as matches are either inclusive or negative (except:), with negative matches having higher priority.

Backslash is supported to escape the separators ';' and ',', and to express special characters such as newline ('\n'), tabulator ('\t'), whitespace ('\s') and backslash ('\\'). The globbing of interface names cannot be escaped. Whitespace is not a separator but will be trimmed between two specs (unless escaped as '\s').

Example:

```
interface−name:em4
mac:00:22:68:1c:59:b1;mac:00:1E:65:30:D1:C4;interface−name:eth2
interface−name:vboxnet*,except:interface−name:vboxnet2
*,except:mac:00:22:68:1c:59:b1
```

SEE ALSO

NetworkManager(8), **nmcli**(1), **nmcli-examples**(7), **nm-online**(1), **nm-settings**(5), **nm-applet**(1), **nm-connection-editor**(1)