Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'nm-settings-nmcli.5' command

$ man nm-settings-nmcli.5

NM-SETTINGS-NMCLI(5) Configuration NM-SETTINGS-NMCLI(5)

NAME

nm-settings-nmcli - Description of settings and properties of NetworkManager connection profiles for nmcli

DESCRIPTION

NetworkManager is based on a concept of connection profiles, sometimes referred to as connections only. These connection profiles contain a network configuration. When NetworkManager activates a connection profile on a network device the configuration will be applied and an active network connection will be established. Users are free to create as many connection profiles as they see fit. Thus they are flexible in having various network configurations for different networking needs. NetworkManager provides an API for configuring connection profiles, for activating them to configure the network, and inspecting the current network configuration. The command line tool nmcli is a client application to NetworkManager that uses this API. See nmcli(1) for details.

With commands like nmcli connection add, nmcli connection modify and nmcli connection show, connection profiles can be created, modified and inspected. A profile consists of properties. On D-Bus this follows the format as described by nm-settings-dbus(5), while this manual page describes the settings format how they are expected by nmcli.

The settings and properties shown in tables below list all available

connection configuration options. However, note that not all settings are applicable to all connection types. nmcli connection editor has also a built-in describe command that can display description of particular settings and properties of this page.

The setting and property can be abbreviated provided they are unique. The list below also shows aliases that can be used unqualified instead of the full name. For example connection.interface-name and ifname refer to the same property.

connection setting

General Connection Profile Settings.

Properties:

auth-retries

The number of retries for the authentication. Zero means to try indefinitely; -1 means to use a global default. If the global default is not set, the authentication retries for 3 times before failing the connection.

Currently, this only applies to 802-1x authentication.

Format: int32

autoconnect

Alias: autoconnect

Whether or not the connection should be automatically connected by NetworkManager when the resources for the connection are available. TRUE to automatically activate the connection, FALSE to require manual intervention to activate the connection.

Autoconnect happens when the circumstances are suitable. That means for example that the device is currently managed and not active. Autoconnect thus never replaces or competes with an already active profile.

Note that autoconnect is not implemented for VPN profiles. See "secondaries" as an alternative to automatically connect VPN profiles.

If multiple profiles are ready to autoconnect on the same device, the one with the better "connection.autoconnect-priority" is

chosen. If the priorities are equal, then the most recently connected profile is activated. If the profiles were not connected earlier or their "connection.timestamp" is identical, the choice is undefined.

Depending on "connection.multi-connect", a profile can (auto)connect only once at a time or multiple times.

Format: boolean

autoconnect-priority

The autoconnect priority in range -999 to 999. If the connection is set to autoconnect, connections with higher priority will be preferred. The higher number means higher priority. Defaults to 0. Note that this property only matters if there are more than one candidate profile to select for autoconnect. In case of equal priority, the profile used most recently is chosen.

Format: int32

autoconnect-retries

The number of times a connection should be tried when autoactivating before giving up. Zero means forever, -1 means the global default (4 times if not overridden). Setting this to 1 means to try activation only once before blocking autoconnect. Note that after a timeout, NetworkManager will try to autoconnect again.

Format: int32

autoconnect-slaves

Whether or not slaves of this connection should be automatically brought up when NetworkManager activates this connection. This only has a real effect for master connections. The properties "autoconnect", "autoconnect-priority" and "autoconnect-retries" are unrelated to this setting. The permitted values are: 0: leave slave connections untouched, 1: activate all the slave connections with this connection, -1: default. If -1 (default) is set, global connection.autoconnect-slaves is read to determine the real value. If it is default as well, this fallbacks to 0.

Format: NMSettingConnectionAutoconnectSlaves (int32)

dns-over-tls

Whether DNSOverTls (dns-over-tls) is enabled for the connection. DNSOverTls is a technology which uses TLS to encrypt dns traffic. The permitted values are: "yes" (2) use DNSOverTls and disabled fallback, "opportunistic" (1) use DNSOverTls but allow fallback to unencrypted resolution, "no" (0) don't ever use DNSOverTls. If unspecified "default" depends on the plugin used. Systemd-resolved uses global setting. This feature requires a plugin which supports DNSOverTls. Otherwise, the setting has no effect. One such plugin is dns-systemd-resolved.

Format: int32

gateway-ping-timeout

If greater than zero, delay success of IP addressing until either the timeout is reached, or an IP gateway replies to a ping.

Format: uint32

id

Alias: con-name

A human readable unique identifier for the connection, like "Work Wi-Fi" or "T-Mobile 3G".

Format: string

interface-name

Alias: ifname

The name of the network interface this connection is bound to. If not set, then the connection can be attached to any interface of the appropriate type (subject to restrictions imposed by other settings). For software devices this specifies the name of the created device. For connection types where interface names cannot easily be made persistent (e.g. mobile broadband or USB Ethernet), this property should not be used. Setting this property restricts the interfaces a connection can be used with, and if interface names change or are reordered the connection may be applied to the wrong interface.

Format: string

lldp

Whether LLDP is enabled for the connection.

Format: int32

llmnr

Whether Link-Local Multicast Name Resolution (LLMNR) is enabled for the connection. LLMNR is a protocol based on the Domain Name System (DNS) packet format that allows both IPv4 and IPv6 hosts to perform name resolution for hosts on the same local link.

The permitted values are: "yes" (2) register hostname and resolving for the connection, "no" (0) disable LLMNR for the interface, "resolve" (1) do not register hostname but allow resolving of LLMNR host names If unspecified, "default" ultimately depends on the DNS plugin (which for systemd-resolved currently means "yes").

This feature requires a plugin which supports LLMNR. Otherwise, the setting has no effect. One such plugin is dns-systemd-resolved.

Format: int32

master

Alias: master

Interface name of the master device or UUID of the master connection.

Format: string

mdns

Whether mDNS is enabled for the connection.

The permitted values are: "yes" (2) register hostname and resolving for the connection, "no" (0) disable mDNS for the interface, "resolve" (1) do not register hostname but allow resolving of mDNS host names and "default" (-1) to allow lookup of a global default in NetworkManager.conf. If unspecified, "default" ultimately depends on the DNS plugin (which for systemd-resolved currently means "no").

This feature requires a plugin which supports mDNS. Otherwise, the setting has no effect. One such plugin is dns-systemd-resolved.

Format: int32

metered

Whether the connection is metered.

When updating this property on a currently activated connection, the change takes effect immediately.

Format: NMMetered (int32)

mptcp-flags

Whether to configure MPTCP endpoints and the address flags. If MPTCP is enabled in NetworkManager, it will configure the addresses of the interface as MPTCP endpoints. Note that IPv4 loopback addresses (127.0.0.0/8), IPv4 link local addresses (169.254.0.0/16), the IPv6 loopback address (::1), IPv6 link local addresses (fe80::/10), IPv6 unique local addresses (ULA, fc00::/7) and IPv6 privacy extension addresses (rfc3041, ipv6.ip6-privacy) will be excluded from being configured as endpoints.

If "disabled" (0x1), MPTCP handling for the interface is disabled and no endpoints are registered.

The "enabled" (0x2) flag means that MPTCP handling is enabled. This flag can also be implied from the presence of other flags.

Even when enabled, MPTCP handling will by default still be disabled unless "/proc/sys/net/mptcp/enabled" sysctl is on. NetworkManager does not change the sysctl and this is up to the administrator or distribution. To configure endpoints even if the sysctl is disabled, "also-without-sysctl" (0x4) flag can be used. In that case, NetworkManager doesn't look at the sysctl and configures endpoints regardless.

Even when enabled, NetworkManager will only configure MPTCP endpoints for a certain address family, if there is a unicast default route (0.0.0.0/0 or ::/0) in the main routing table. The flag "also-without-default-route" (0x8) can override that.

When MPTCP handling is enabled then endpoints are configured with the specified address flags "signal" (0x10), "subflow" (0x20), "backup" (0x40), "fullmesh" (0x80). See ip-mptcp(8) manual for

additional information about the flags.

If the flags are zero (0x0), the global connection default from NetworkManager.conf is honored. If still unspecified, the fallback is "enabled,subflow". Note that this means that MPTCP is by default done depending on the "/proc/sys/net/mptcp/enabled" sysctl. NetworkManager does not change the MPTCP limits nor enable MPTCP via "/proc/sys/net/mptcp/enabled". That is a host configuration which the admin can change via sysctl and ip-mptcp.

Strict reverse path filtering (rp_filter) breaks many MPTCP use cases, so when MPTCP handling for IPv4 addresses on the interface is enabled, NetworkManager would loosen the strict reverse path filtering (1) to the loose setting (2).

Format: uint32

mud-url

If configured, set to a Manufacturer Usage Description (MUD) URL that points to manufacturer-recommended network policies for IoT devices. It is transmitted as a DHCPv4 or DHCPv6 option. The value must be a valid URL starting with "https://".

The special value "none" is allowed to indicate that no MUD URL is used.

If the per-profile value is unspecified (the default), a global connection default gets consulted. If still unspecified, the ultimate default is "none".

Format: string

multi-connect

Specifies whether the profile can be active multiple times at a particular moment. The value is of type NMConnectionMultiConnect.

Format: int32

permissions

An array of strings defining what access a given user has to this connection. If this is NULL or empty, all users are allowed to access this connection; otherwise users are allowed if and only if they are in this list. When this is not empty, the connection can

be active only when one of the specified users is logged into an active session. Each entry is of the form "[type]:[id]:[reserved]"; for example, "user:dcbw:blah".

At this time only the "user" [type] is allowed. Any other values are ignored and reserved for future use. [id] is the username that this permission refers to, which may not contain the ":" character. Any [reserved] information present must be ignored and is reserved for future use. All of [type], [id], and [reserved] must be valid UTF-8.

Format: array of string

read-only

FALSE if the connection can be modified using the provided settings service's D-Bus interface with the right privileges, or TRUE if the connection is read-only and cannot be modified.

Format: boolean

secondaries

List of connection UUIDs that should be activated when the base connection itself is activated. Currently, only VPN connections are supported.

Format: array of string

slave-type

Alias: slave-type

Setting name of the device type of this slave's master connection (eg, "bond"), or NULL if this connection is not a slave.

Format: string

stable-id

This represents the identity of the connection used for various purposes. It allows to configure multiple profiles to share the identity. Also, the stable-id can contain placeholders that are substituted dynamically and deterministically depending on the context.

The stable-id is used for generating IPv6 stable private addresses with ipv6.addr-gen-mode=stable-privacy. It is also used to seed the

generated cloned MAC address for ethernet.cloned-mac-address=stable and wifi.cloned-mac-address=stable. It is also used as DHCP client identifier with ipv4.dhcp-client-id=stable and to derive the DHCP DUID with ipv6.dhcp-duid=stable-[llt,ll,uuid].

Note that depending on the context where it is used, other parameters are also seeded into the generation algorithm. For example, a per-host key is commonly also included, so that different systems end up generating different IDs. Or with ipv6.addr-gen-mode=stable-privacy, also the device's name is included, so that different interfaces yield different addresses.

The per-host key is the identity of your machine and stored in /var/lib/NetworkManager/secret_key. See NetworkManager(8) manual about the secret-key and the host identity.

The '$' character is treated special to perform dynamic substitutions at runtime. Currently, supported are "${CONNECTION}", "${DEVICE}", "${MAC}", "${BOOT}", "${RANDOM}". These effectively create unique IDs per-connection, per-device, per-boot, or every time. Note that "${DEVICE}" corresponds to the interface name of the device and "${MAC}" is the permanent MAC address of the device.

Any unrecognized patterns following '$' are treated verbatim, however are reserved for future use. You are thus advised to avoid '$' or escape it as "$$". For example, set it to "${CONNECTION}-${BOOT}-${DEVICE}" to create a unique id for this connection that changes with every reboot and differs depending on the interface where the profile activates.

If the value is unset, a global connection default is consulted. If the value is still unset, the default is similar to "${CONNECTION}" and uses a unique, fixed ID for the connection.

Format: string

timestamp

The time, in seconds since the Unix Epoch, that the connection was last _successfully_ fully activated.

NetworkManager updates the connection timestamp periodically when

the connection is active to ensure that an active connection has the latest timestamp. The property is only meant for reading (changes to this property will not be preserved).

Format: uint64

type

Alias: type

Base type of the connection. For hardware-dependent connections, should contain the setting name of the hardware-type specific setting (ie, "802-3-ethernet" or "802-11-wireless" or "bluetooth", etc), and for non-hardware dependent connections like VPN or otherwise, should contain the setting name of that setting type (ie, "vpn" or "bridge", etc).

Format: string

uuid

A universally unique identifier for the connection, for example generated with libuuid. It should be assigned when the connection is created, and never changed as long as the connection still applies to the same network. For example, it should not be changed when the "id" property or NMSettingIP4Config changes, but might need to be re-created when the Wi-Fi SSID, mobile broadband network provider, or "type" property changes.

The UUID must be in the format "2815492f-7e56-435e-b2e9-246bd7cdc664" (ie, contains only hexadecimal characters and "-").

Format: a valid RFC4122 universally unique identifier (UUID).

wait-activation-delay

Time in milliseconds to wait for connection to be considered activated. The wait will start after the pre-up dispatcher event.

The value 0 means no wait time. The default value is -1, which currently has the same meaning as no wait time.

Format: int32

wait-device-timeout

Timeout in milliseconds to wait for device at startup. During boot,

devices may take a while to be detected by the driver. This property will cause to delay NetworkManager-wait-online.service and nm-online to give the device a chance to appear. This works by waiting for the given timeout until a compatible device for the profile is available and managed.

The value 0 means no wait time. The default value is -1, which currently has the same meaning as no wait time.

Format: int32

zone

The trust level of a the connection. Free form case-insensitive string (for example "Home", "Work", "Public"). NULL or unspecified zone means the connection will be placed in the default zone as defined by the firewall.

When updating this property on a currently activated connection, the change takes effect immediately.

Format: string

6lowpan setting

6LoWPAN Settings.

Properties:

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this 6LowPAN interface should be created.

Format: string

802-1x setting

IEEE 802.1x Authentication Settings.

Properties:

altsubject-matches

List of strings to be matched against the altSubjectName of the certificate presented by the authentication server. If the list is empty, no verification of the server certificate's altSubjectName is performed.

Format: array of string

anonymous-identity

Anonymous identity string for EAP authentication methods. Used as the unencrypted identity with EAP types that support different tunneled identity like EAP-TTLS.

Format: string

auth-timeout

A timeout for the authentication. Zero means the global default; if the global default is not set, the authentication timeout is 25 seconds.

Format: int32

ca-cert

Contains the CA certificate if used by the EAP method specified in the "eap" property.

Certificate data is specified using a "scheme"; three are currently supported: blob, path and pkcs#11 URL. When using the blob scheme this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte. This property can be unset even if the EAP method supports CA certificates, but this allows man-in-the-middle attacks and is NOT recommended.

Note that enabling NMSetting8021x:system-ca-certs will override this setting to use the built-in path, if the built-in path is not a directory.

Format: byte array

ca-cert-password

The password used to access the CA certificate stored in "ca-cert" property. Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

ca-cert-password-flags

Flags indicating how to handle the "ca-cert-password" property.

Format: NMSettingSecretFlags (uint32)

ca-path

UTF-8 encoded path to a directory containing PEM or DER formatted certificates to be added to the verification chain in addition to the certificate specified in the "ca-cert" property.

If NMSetting8021x:system-ca-certs is enabled and the built-in CA path is an existing directory, then this setting is ignored.

Format: string

client-cert

Contains the client certificate if used by the EAP method specified in the "eap" property.

Certificate data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme (which is backwards compatible with NM 0.7.x) this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte.

Format: byte array

client-cert-password

The password used to access the client certificate stored in "client-cert" property. Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

client-cert-password-flags

Flags indicating how to handle the "client-cert-password" property.

Format: NMSettingSecretFlags (uint32)

domain-match

Constraint for server domain name. If set, this list of FQDNs is used as a match requirement for dNSName element(s) of the certificate presented by the authentication server. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this constraint is matched against SubjectName CN using the same comparison. Multiple valid FQDNs can be passed as a ";"

delimited list.

Format: string

domain-suffix-match

Constraint for server domain name. If set, this FQDN is used as a suffix match requirement for dNSName element(s) of the certificate presented by the authentication server. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this constraint is matched against SubjectName CN using same suffix match comparison. Since version 1.24, multiple valid FQDNs can be passed as a ";" delimited list.

Format: string

eap

The allowed EAP method to be used when authenticating to the network with 802.1x. Valid methods are: "leap", "md5", "tls", "peap", "ttls", "pwd", and "fast". Each method requires different configuration using the properties of this setting; refer to wpa_supplicant documentation for the allowed combinations.

Format: array of string

identity

Identity string for EAP authentication methods. Often the user's user or login name.

Format: string

optional

Whether the 802.1X authentication is optional. If TRUE, the activation will continue even after a timeout or an authentication failure. Setting the property to TRUE is currently allowed only for Ethernet connections. If set to FALSE, the activation can continue only after a successful authentication.

Format: boolean

pac-file

UTF-8 encoded file path containing PAC for EAP-FAST.

Format: string

password

UTF-8 encoded password used for EAP authentication methods. If both the "password" property and the "password-raw" property are specified, "password" is preferred.

Format: string

password-flags

Flags indicating how to handle the "password" property.

Format: NMSettingSecretFlags (uint32)

password-raw

Password used for EAP authentication methods, given as a byte array to allow passwords in other encodings than UTF-8 to be used. If both the "password" property and the "password-raw" property are specified, "password" is preferred.

Format: byte array

password-raw-flags

Flags indicating how to handle the "password-raw" property.

Format: NMSettingSecretFlags (uint32)

phase1-auth-flags

Specifies authentication flags to use in "phase 1" outer authentication using NMSetting8021xAuthFlags options. The individual TLS versions can be explicitly disabled. TLS time checks can be also disabled. If a certain TLS disable flag is not set, it is up to the supplicant to allow or forbid it. The TLS options map to tls_disable_tlsv1_x and tls_disable_time_checks settings. See the wpa_supplicant documentation for more details.

Format: uint32

phase1-fast-provisioning

Enables or disables in-line provisioning of EAP-FAST credentials when FAST is specified as the EAP method in the "eap" property. Recognized values are "0" (disabled), "1" (allow unauthenticated provisioning), "2" (allow authenticated provisioning), and "3" (allow both authenticated and unauthenticated provisioning). See the wpa_supplicant documentation for more details.

Format: string

phase1-peaplabel

Forces use of the new PEAP label during key derivation. Some RADIUS servers may require forcing the new PEAP label to interoperate with PEAPv1. Set to "1" to force use of the new PEAP label. See the wpa_supplicant documentation for more details.

Format: string

phase1-peapver

Forces which PEAP version is used when PEAP is set as the EAP method in the "eap" property. When unset, the version reported by the server will be used. Sometimes when using older RADIUS servers, it is necessary to force the client to use a particular PEAP version. To do so, this property may be set to "0" or "1" to force that specific PEAP version.

Format: string

phase2-altsubject-matches

List of strings to be matched against the altSubjectName of the certificate presented by the authentication server during the inner "phase 2" authentication. If the list is empty, no verification of the server certificate's altSubjectName is performed.

Format: array of string

phase2-auth

Specifies the allowed "phase 2" inner authentication method when an EAP method that uses an inner TLS tunnel is specified in the "eap" property. For TTLS this property selects one of the supported non-EAP inner methods: "pap", "chap", "mschap", "mschapv2" while "phase2-autheap" selects an EAP inner method. For PEAP this selects an inner EAP method, one of: "gtc", "otp", "md5" and "tls". Each "phase 2" inner method requires specific parameters for successful authentication; see the wpa_supplicant documentation for more details. Both "phase2-auth" and "phase2-autheap" cannot be specified.

Format: string

phase2-autheap

Specifies the allowed "phase 2" inner EAP-based authentication method when TTLS is specified in the "eap" property. Recognized EAP-based "phase 2" methods are "md5", "mschapv2", "otp", "gtc", and "tls". Each "phase 2" inner method requires specific parameters for successful authentication; see the wpa_supplicant documentation for more details.

Format: string

phase2-ca-cert

Contains the "phase 2" CA certificate if used by the EAP method specified in the "phase2-auth" or "phase2-autheap" properties. Certificate data is specified using a "scheme"; three are currently supported: blob, path and pkcs#11 URL. When using the blob scheme this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte. This property can be unset even if the EAP method supports CA certificates, but this allows man-in-the-middle attacks and is NOT recommended. Note that enabling NMSetting8021x:system-ca-certs will override this setting to use the built-in path, if the built-in path is not a directory.

Format: byte array

phase2-ca-cert-password

The password used to access the "phase2" CA certificate stored in "phase2-ca-cert" property. Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

phase2-ca-cert-password-flags

Flags indicating how to handle the "phase2-ca-cert-password" property.

Format: NMSettingSecretFlags (uint32)

phase2-ca-path

UTF-8 encoded path to a directory containing PEM or DER formatted

certificates to be added to the verification chain in addition to the certificate specified in the "phase2-ca-cert" property. If NMSetting8021x:system-ca-certs is enabled and the built-in CA path is an existing directory, then this setting is ignored.

Format: string

phase2-client-cert

Contains the "phase 2" client certificate if used by the EAP method specified in the "phase2-auth" or "phase2-autheap" properties. Certificate data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme (which is backwards compatible with NM 0.7.x) this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte. This property can be unset even if the EAP method supports CA certificates, but this allows man-in-the-middle attacks and is NOT recommended.

Format: byte array

phase2-client-cert-password

The password used to access the "phase2" client certificate stored in "phase2-client-cert" property. Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

phase2-client-cert-password-flags

Flags indicating how to handle the "phase2-client-cert-password" property.

Format: NMSettingSecretFlags (uint32)

phase2-domain-match

Constraint for server domain name. If set, this list of FQDNs is used as a match requirement for dNSName element(s) of the certificate presented by the authentication server during the inner "phase 2" authentication. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this

constraint is matched against SubjectName CN using the same comparison. Multiple valid FQDNs can be passed as a ";" delimited list.

Format: string

phase2-domain-suffix-match

Constraint for server domain name. If set, this FQDN is used as a suffix match requirement for dNSName element(s) of the certificate presented by the authentication server during the inner "phase 2" authentication. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this constraint is matched against SubjectName CN using same suffix match comparison. Since version 1.24, multiple valid FQDNs can be passed as a ";" delimited list.

Format: string

phase2-private-key

Contains the "phase 2" inner private key when the "phase2-auth" or "phase2-autheap" property is set to "tls".

Key data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme and private keys, this property should be set to the key's encrypted PEM encoded data. When using private keys with the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte. When using PKCS#12 format private keys and the blob scheme, this property should be set to the PKCS#12 data and the "phase2-private-key-password" property must be set to password used to decrypt the PKCS#12 certificate and key. When using PKCS#12 files and the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte, and as with the blob scheme the "phase2-private-key-password" property must be set to the password used to decode the PKCS#12 private key and certificate.

Format: byte array

phase2-private-key-password

The password used to decrypt the "phase 2" private key specified in the "phase2-private-key" property when the private key either uses the path scheme, or is a PKCS#12 format key.

Format: string

phase2-private-key-password-flags

Flags indicating how to handle the "phase2-private-key-password" property.

Format: NMSettingSecretFlags (uint32)

phase2-subject-match

Substring to be matched against the subject of the certificate presented by the authentication server during the inner "phase 2" authentication. When unset, no verification of the authentication server certificate's subject is performed. This property provides little security, if any, and should not be used.

This property is deprecated since version 1.2. Use "phase2-domain-suffix-match" instead.

Format: string

pin

PIN used for EAP authentication methods.

Format: string

pin-flags

Flags indicating how to handle the "pin" property.

Format: NMSettingSecretFlags (uint32)

private-key

Contains the private key when the "eap" property is set to "tls". Key data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme and private keys, this property should be set to the key's encrypted PEM encoded data. When using private keys with the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte. When using PKCS#12 format private keys and the blob

scheme, this property should be set to the PKCS#12 data and the "private-key-password" property must be set to password used to decrypt the PKCS#12 certificate and key. When using PKCS#12 files and the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte, and as with the blob scheme the "private-key-password" property must be set to the password used to decode the PKCS#12 private key and certificate.

WARNING: "private-key" is not a "secret" property, and thus unencrypted private key data using the BLOB scheme may be readable by unprivileged users. Private keys should always be encrypted with a private key password to prevent unauthorized access to unencrypted private key data.

Format: byte array

private-key-password

The password used to decrypt the private key specified in the "private-key" property when the private key either uses the path scheme, or if the private key is a PKCS#12 format key.

Format: string

private-key-password-flags

Flags indicating how to handle the "private-key-password" property.

Format: NMSettingSecretFlags (uint32)

subject-match

Substring to be matched against the subject of the certificate presented by the authentication server. When unset, no verification of the authentication server certificate's subject is performed.

This property provides little security, if any, and should not be used.

This property is deprecated since version 1.2. Use "phase2-domain-suffix-match" instead.

Format: string

system-ca-certs

When TRUE, overrides the "ca-path" and "phase2-ca-path" properties

using the system CA directory specified at configure time with the --system-ca-path switch. The certificates in this directory are added to the verification chain in addition to any certificates specified by the "ca-cert" and "phase2-ca-cert" properties. If the path provided with --system-ca-path is rather a file name (bundle of trusted CA certificates), it overrides "ca-cert" and "phase2-ca-cert" properties instead (sets ca_cert/ca_cert2 options for wpa_supplicant).

Format: boolean

adsl setting

ADSL Settings.

Properties:

encapsulation

Alias: encapsulation

Encapsulation of ADSL connection. Can be "vcmux" or "llc".

Format: string

password

Alias: password

Password used to authenticate with the ADSL service.

Format: string

password-flags

Flags indicating how to handle the "password" property.

Format: NMSettingSecretFlags (uint32)

protocol

Alias: protocol

ADSL connection protocol. Can be "pppoa", "pppoe" or "ipoatm".

Format: string

username

Alias: username

Username used to authenticate with the ADSL service.

Format: string

vci

VCI of ADSL connection

Format: uint32

vpi

VPI of ADSL connection

Format: uint32

bluetooth setting

Bluetooth Settings.

Properties:

bdaddr

Alias: addr

The Bluetooth address of the device.

Format: byte array

type

Alias: bt-type

Either "dun" for Dial-Up Networking connections or "panu" for

Personal Area Networking connections to devices supporting the NAP

profile.

Format: string

bond setting

Bonding Settings.

Properties:

options

Dictionary of key/value pairs of bonding options. Both keys and

values must be strings. Option names must contain only alphanumeric

characters (ie, [a-zA-Z0-9]).

Format: dict of string to string

bridge setting

Bridging Settings.

Properties:

ageing-time

Alias: ageing-time

The Ethernet MAC address aging time, in seconds.

Format: uint32

forward-delay

Alias: forward-delay

The Spanning Tree Protocol (STP) forwarding delay, in seconds.

Format: uint32

group-address

If specified, The MAC address of the multicast group this bridge uses for STP.

The address must be a link-local address in standard Ethernet MAC address format, ie an address of the form 01:80:C2:00:00:0X, with X in [0, 4..F]. If not specified the default value is 01:80:C2:00:00:00.

Format: byte array

group-forward-mask

Alias: group-forward-mask

A mask of group addresses to forward. Usually, group addresses in the range from 01:80:C2:00:00:00 to 01:80:C2:00:00:0F are not forwarded according to standards. This property is a mask of 16 bits, each corresponding to a group address in that range that must be forwarded. The mask can't have bits 0, 1 or 2 set because they are used for STP, MAC pause frames and LACP.

Format: uint32

hello-time

Alias: hello-time

The Spanning Tree Protocol (STP) hello time, in seconds.

Format: uint32

mac-address

Alias: mac

If specified, the MAC address of bridge. When creating a new bridge, this MAC address will be set.

If this field is left unspecified, the "ethernet.cloned-mac-address" is referred instead to generate the initial MAC address. Note that setting "ethernet.cloned-mac-address" anyway overwrites the MAC address of the bridge later while activating the bridge.

This property is deprecated since version 1.12. Use the "cloned-mac-address" property instead.

Format: byte array

max-age

Alias: max-age

The Spanning Tree Protocol (STP) maximum message age, in seconds.

Format: uint32

multicast-hash-max

Set maximum size of multicast hash table (value must be a power of 2).

Format: uint32

multicast-last-member-count

Set the number of queries the bridge will send before stopping forwarding a multicast group after a "leave" message has been received.

Format: uint32

multicast-last-member-interval

Set interval (in deciseconds) between queries to find remaining members of a group, after a "leave" message is received.

Format: uint64

multicast-membership-interval

Set delay (in deciseconds) after which the bridge will leave a group, if no membership reports for this group are received.

Format: uint64

multicast-querier

Enable or disable sending of multicast queries by the bridge. If not specified the option is disabled.

Format: boolean

multicast-querier-interval

If no queries are seen after this delay (in deciseconds) has passed, the bridge will start to send its own queries.

Format: uint64

multicast-query-interval

Interval (in deciseconds) between queries sent by the bridge after the end of the startup phase.

Format: uint64

multicast-query-response-interval

Set the Max Response Time/Max Response Delay (in deciseconds) for IGMP/MLD queries sent by the bridge.

Format: uint64

multicast-query-use-ifaddr

If enabled the bridge's own IP address is used as the source address for IGMP queries otherwise the default of 0.0.0.0 is used.

Format: boolean

multicast-router

Sets bridge's multicast router. Multicast-snooping must be enabled for this option to work.

Supported values are: 'auto', 'disabled', 'enabled' to which kernel assigns the numbers 1, 0, and 2, respectively. If not specified the default value is 'auto' (1).

Format: string

multicast-snooping

Alias: multicast-snooping

Controls whether IGMP snooping is enabled for this bridge. Note that if snooping was automatically disabled due to hash collisions, the system may refuse to enable the feature until the collisions are resolved.

Format: boolean

multicast-startup-query-count

Set the number of IGMP queries to send during startup phase.

Format: uint32

multicast-startup-query-interval

Sets the time (in deciseconds) between queries sent out at startup to determine membership information.

Format: uint64

priority

Alias: priority

Sets the Spanning Tree Protocol (STP) priority for this bridge.

Lower values are "better"; the lowest priority bridge will be

elected the root bridge.

Format: uint32

stp

Alias: stp

Controls whether Spanning Tree Protocol (STP) is enabled for this

bridge.

Format: boolean

vlan-default-pvid

The default PVID for the ports of the bridge, that is the VLAN id

assigned to incoming untagged frames.

Format: uint32

vlan-filtering

Control whether VLAN filtering is enabled on the bridge.

Format: boolean

vlan-protocol

If specified, the protocol used for VLAN filtering.

Supported values are: '802.1Q', '802.1ad'. If not specified the

default value is '802.1Q'.

Format: string

vlan-stats-enabled

Controls whether per-VLAN stats accounting is enabled.

Format: boolean

vlans

Array of bridge VLAN objects. In addition to the VLANs specified

here, the bridge will also have the default-pvid VLAN configured by

the bridge.vlan-default-pvid property.

In nmcli the VLAN list can be specified with the following syntax:

$vid [pvid] [untagged] [, $vid [pvid] [untagged]]...

where $vid is either a single id between 1 and 4094 or a range,

represented as a couple of ids separated by a dash.

Format: array of vardict

bridge-port setting

Bridge Port Settings.

Properties:

hairpin-mode

Alias: hairpin

Enables or disables "hairpin mode" for the port, which allows frames to be sent back out through the port the frame was received on.

Format: boolean

path-cost

Alias: path-cost

The Spanning Tree Protocol (STP) port cost for destinations via this port.

Format: uint32

priority

Alias: priority

The Spanning Tree Protocol (STP) priority of this bridge port.

Format: uint32

vlans

Array of bridge VLAN objects. In addition to the VLANs specified here, the port will also have the default-pvid VLAN configured on the bridge by the bridge.vlan-default-pvid property.

In nmcli the VLAN list can be specified with the following syntax:

$vid [pvid] [untagged] [, $vid [pvid] [untagged]]...

where $vid is either a single id between 1 and 4094 or a range, represented as a couple of ids separated by a dash.

Format: array of vardict

cdma setting

CDMA-based Mobile Broadband Settings.

Properties:

mtu

If non-zero, only transmit packets of the specified size or

smaller, breaking larger packets up into multiple frames.

Format: uint32

number

The number to dial to establish the connection to the CDMA-based mobile broadband network, if any. If not specified, the default number (#777) is used when required.

Format: string

password

Alias: password

The password used to authenticate with the network, if required.

Many providers do not require a password, or accept any password.

But if a password is required, it is specified here.

Format: string

password-flags

Flags indicating how to handle the "password" property.

Format: NMSettingSecretFlags (uint32)

username

Alias: user

The username used to authenticate with the network, if required.

Many providers do not require a username, or accept any username.

But if a username is required, it is specified here.

Format: string

dcb setting

Data Center Bridging Settings.

Properties:

app-fcoe-flags

Specifies the NMSettingDcbFlags for the DCB FCoE application. Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

app-fcoe-mode

The FCoE controller mode; either "fabric" or "vn2vn".

Since 1.34, NULL is the default and means "fabric". Before 1.34,

NULL was rejected as invalid and the default was "fabric".

Format: string

app-fcoe-priority

The highest User Priority (0 - 7) which FCoE frames should use, or

-1 for default priority. Only used when the "app-fcoe-flags"

property includes the NM_SETTING_DCB_FLAG_ENABLE (0x1) flag.

Format: int32

app-fip-flags

Specifies the NMSettingDcbFlags for the DCB FIP application. Flags

may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1),

NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and

NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

app-fip-priority

The highest User Priority (0 - 7) which FIP frames should use, or

-1 for default priority. Only used when the "app-fip-flags"

property includes the NM_SETTING_DCB_FLAG_ENABLE (0x1) flag.

Format: int32

app-iscsi-flags

Specifies the NMSettingDcbFlags for the DCB iSCSI application.

Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1),

NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and

NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

app-iscsi-priority

The highest User Priority (0 - 7) which iSCSI frames should use, or

-1 for default priority. Only used when the "app-iscsi-flags"

property includes the NM_SETTING_DCB_FLAG_ENABLE (0x1) flag.

Format: int32

priority-bandwidth

An array of 8 uint values, where the array index corresponds to the

User Priority (0 - 7) and the value indicates the percentage of

bandwidth of the priority's assigned group that the priority may use. The sum of all percentages for priorities which belong to the same group must total 100 percents.

Format: array of uint32

priority-flow-control

An array of 8 boolean values, where the array index corresponds to the User Priority (0 - 7) and the value indicates whether or not the corresponding priority should transmit priority pause.

Format: array of uint32

priority-flow-control-flags

Specifies the NMSettingDcbFlags for DCB Priority Flow Control (PFC). Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

priority-group-bandwidth

An array of 8 uint values, where the array index corresponds to the Priority Group ID (0 - 7) and the value indicates the percentage of link bandwidth allocated to that group. Allowed values are 0 - 100, and the sum of all values must total 100 percents.

Format: array of uint32

priority-group-flags

Specifies the NMSettingDcbFlags for DCB Priority Groups. Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

priority-group-id

An array of 8 uint values, where the array index corresponds to the User Priority (0 - 7) and the value indicates the Priority Group ID. Allowed Priority Group ID values are 0 - 7 or 15 for the unrestricted group.

Format: array of uint32

priority-strict-bandwidth

An array of 8 boolean values, where the array index corresponds to the User Priority (0 - 7) and the value indicates whether or not the priority may use all of the bandwidth allocated to its assigned group.

Format: array of uint32

priority-traffic-class

An array of 8 uint values, where the array index corresponds to the User Priority (0 - 7) and the value indicates the traffic class (0 - 7) to which the priority is mapped.

Format: array of uint32

ethtool setting

Ethtool Ethernet Settings.

Properties:

coalesce-adaptive-rx

coalesce-adaptive-tx

coalesce-pkt-rate-high

coalesce-pkt-rate-low

coalesce-rx-frames

coalesce-rx-frames-high

coalesce-rx-frames-irq

coalesce-rx-frames-low

coalesce-rx-usecs

coalesce-rx-usecs-high

coalesce-rx-usecs-irq

coalesce-rx-usecs-low

coalesce-sample-interval

coalesce-stats-block-usecs

coalesce-tx-frames

coalesce-tx-frames-high

coalesce-tx-frames-irq

coalesce-tx-frames-low

coalesce-tx-usecs

coalesce-tx-usecs-high

coalesce-tx-usecs-irq

coalesce-tx-usecs-low

feature-esp-hw-offload

feature-esp-tx-csum-hw-offload

feature-fcoe-mtu

feature-gro

feature-gso

feature-highdma

feature-hw-tc-offload

feature-l2-fwd-offload

feature-loopback

feature-lro

feature-macsec-hw-offload

feature-ntuple

feature-rx

feature-rx-all

feature-rx-fcs

feature-rx-gro-hw

feature-rx-gro-list

feature-rx-udp-gro-forwarding

feature-rx-udp_tunnel-port-offload

feature-rx-vlan-filter

feature-rx-vlan-stag-filter

feature-rx-vlan-stag-hw-parse

feature-rxhash

feature-rxvlan

feature-sg

feature-tls-hw-record

feature-tls-hw-rx-offload

feature-tls-hw-tx-offload

feature-tso

feature-tx

feature-tx-checksum-fcoe-crc

feature-tx-checksum-ip-generic

feature-tx-checksum-ipv4

feature-tx-checksum-ipv6

feature-tx-checksum-sctp

feature-tx-esp-segmentation

feature-tx-fcoe-segmentation

feature-tx-gre-csum-segmentation

feature-tx-gre-segmentation

feature-tx-gso-list

feature-tx-gso-partial

feature-tx-gso-robust

feature-tx-ipxip4-segmentation

feature-tx-ipxip6-segmentation

feature-tx-nocache-copy

feature-tx-scatter-gather

feature-tx-scatter-gather-fraglist

feature-tx-sctp-segmentation

feature-tx-tcp-ecn-segmentation

feature-tx-tcp-mangleid-segmentation

feature-tx-tcp-segmentation

feature-tx-tcp6-segmentation

feature-tx-tunnel-remcsum-segmentation

feature-tx-udp-segmentation

feature-tx-udp_tnl-csum-segmentation

feature-tx-udp_tnl-segmentation

feature-tx-vlan-stag-hw-insert

feature-txvlan

pause-autoneg

pause-rx

pause-tx

ring-rx

ring-rx-jumbo

ring-rx-mini

ring-tx

gsm setting

GSM-based Mobile Broadband Settings.

Properties:

apn

Alias: apn

The GPRS Access Point Name specifying the APN used when establishing a data session with the GSM-based network. The APN often determines how the user will be billed for their network usage and whether the user has access to the Internet or just a provider-specific walled-garden, so it is important to use the correct APN for the user's mobile broadband plan. The APN may only be composed of the characters a-z, 0-9, ., and - per GSM 03.60 Section 14.9.

Format: string

auto-config

When TRUE, the settings such as APN, username, or password will default to values that match the network the modem will register to in the Mobile Broadband Provider database.

Format: boolean

device-id

The device unique identifier (as given by the WWAN management service) which this connection applies to. If given, the connection will only apply to the specified device.

Format: string

home-only

When TRUE, only connections to the home network will be allowed. Connections to roaming networks will not be made.

Format: boolean

mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple frames.

Format: uint32

network-id

The Network ID (GSM LAI format, ie MCC-MNC) to force specific network registration. If the Network ID is specified, NetworkManager will attempt to force the device to register only on the specified network. This can be used to ensure that the device does not roam when direct roaming control of the device is not otherwise possible.

Format: string

number

Legacy setting that used to help establishing PPP data sessions for GSM-based modems.

This property is deprecated since version 1.16. User-provided values for this setting are no longer used.

Format: string

password

Alias: password

The password used to authenticate with the network, if required. Many providers do not require a password, or accept any password. But if a password is required, it is specified here.

Format: string

password-flags

Flags indicating how to handle the "password" property.

Format: NMSettingSecretFlags (uint32)

pin

If the SIM is locked with a PIN it must be unlocked before any other operations are requested. Specify the PIN here to allow operation of the device.

Format: string

pin-flags

Flags indicating how to handle the "pin" property.

Format: NMSettingSecretFlags (uint32)

sim-id

The SIM card unique identifier (as given by the WWAN management service) which this connection applies to. If given, the connection will apply to any device also allowed by "device-id" which contains a SIM card matching the given identifier.

Format: string

sim-operator-id

A MCC/MNC string like "310260" or "21601" identifying the specific mobile network operator which this connection applies to. If given, the connection will apply to any device also allowed by "device-id" and "sim-id" which contains a SIM card provisioned by the given operator.

Format: string

username

Alias: user

The username used to authenticate with the network, if required. Many providers do not require a username, or accept any username. But if a username is required, it is specified here.

Format: string

infiniband setting

Infiniband Settings.

Properties:

mac-address

Alias: mac

If specified, this connection will only apply to the IPoIB device whose permanent MAC address matches. This property does not change the MAC address of the device (i.e. MAC spoofing).

Format: byte array

mtu

Alias: mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple frames.

Format: uint32

p-key

Alias: p-key

The InfiniBand P_Key to use for this device. A value of -1 means to use the default P_Key (aka "the P_Key at index 0"). Otherwise, it is a 16-bit unsigned integer, whose high bit 0x8000 is set if it is a "full membership" P_Key. The values 0 and 0x8000 are not allowed.

With the p-key set, the interface name is always "$parent.$p_key". Setting "connection.interface-name" to another name is not supported.

Note that kernel will internally always set the full membership bit, although the interface name does not reflect that. Thus, not setting the high bit is probably not useful.

If the profile is stored in ifcfg-rh format, then the full membership bit is automatically added. To get consistent behavior, it is best to only use p-key values with the full membership bit set.

Format: int32

parent

Alias: parent

The interface name of the parent device of this device. Normally NULL, but if the "p_key" property is set, then you must specify the base device by setting either this property or "mac-address".

Format: string

transport-mode

Alias: transport-mode

The IP-over-InfiniBand transport mode. Either "datagram" or "connected".

Format: string

ipv4 setting

IPv4 Settings.

Properties:

addresses

Alias: ip4

Array of IP addresses.

Format: a comma separated list of addresses

auto-route-ext-gw

VPN connections will default to add the route automatically unless this setting is set to FALSE.

For other connection types, adding such an automatic route is currently not supported and setting this to TRUE has no effect.

Format: NMTernary (int32)

dad-timeout

Timeout in milliseconds used to check for the presence of duplicate IP addresses on the network. If an address conflict is detected, the activation will fail. A zero value means that no duplicate address detection is performed, -1 means the default value (either configuration ipvx.dad-timeout override or zero). A value greater than zero is a timeout in milliseconds.

The property is currently implemented only for IPv4.

Format: int32

dhcp-client-id

A string sent to the DHCP server to identify the local machine which the DHCP server may use to customize the DHCP lease and options. When the property is a hex string ('aa:bb:cc') it is interpreted as a binary client ID, in which case the first byte is assumed to be the 'type' field as per RFC 2132 section 9.14 and the remaining bytes may be an hardware address (e.g. '01:xx:xx:xx:xx:xx:xx' where 1 is the Ethernet ARP type and the rest is a MAC address). If the property is not a hex string it is considered as a non-hardware-address client ID and the 'type' field is set to 0.

The special values "mac" and "perm-mac" are supported, which use the current or permanent MAC address of the device to generate a client identifier with type ethernet (01). Currently, these options only work for ethernet type of links.

The special value "ipv6-duid" uses the DUID from "ipv6.dhcp-duid" property as an RFC4361-compliant client identifier. As IAID it uses

"ipv4.dhcp-iaid" and falls back to "ipv6.dhcp-iaid" if unset.

The special value "duid" generates a RFC4361-compliant client identifier based on "ipv4.dhcp-iaid" and uses a DUID generated by hashing /etc/machine-id.

The special value "stable" is supported to generate a type 0 client identifier based on the stable-id (see connection.stable-id) and a per-host key. If you set the stable-id, you may want to include the "${DEVICE}" or "${MAC}" specifier to get a per-device key.

If unset, a globally configured default is used. If still unset, the default depends on the DHCP plugin.

Format: string

dhcp-fqdn

If the "dhcp-send-hostname" property is TRUE, then the specified FQDN will be sent to the DHCP server when acquiring a lease. This property and "dhcp-hostname" are mutually exclusive and cannot be set at the same time.

Format: string

dhcp-hostname

If the "dhcp-send-hostname" property is TRUE, then the specified name will be sent to the DHCP server when acquiring a lease. This property and "dhcp-fqdn" are mutually exclusive and cannot be set at the same time.

Format: string

dhcp-hostname-flags

Flags for the DHCP hostname and FQDN.

Currently, this property only includes flags to control the FQDN flags set in the DHCP FQDN option. Supported FQDN flags are NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1), NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) and NM_DHCP_HOSTNAME_FLAG_FQDN_NO_UPDATE (0x4). When no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is set, the DHCP FQDN option will contain no flag. Otherwise, if no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is not set,

the standard FQDN flags are set in the request:
NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1),
NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) for IPv4 and
NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1) for IPv6.
When this property is set to the default value
NM_DHCP_HOSTNAME_FLAG_NONE (0x0), a global default is looked up in
NetworkManager configuration. If that value is unset or also
NM_DHCP_HOSTNAME_FLAG_NONE (0x0), then the standard FQDN flags
described above are sent in the DHCP requests.
Format: uint32

dhcp-iaid

A string containing the "Identity Association Identifier" (IAID)
used by the DHCP client. The string can be a 32-bit number (either
decimal, hexadecimal or or as colon separated hexadecimal numbers).
Alternatively it can be set to the special values "mac",
"perm-mac", "ifname" or "stable". When set to "mac" (or
"perm-mac"), the last 4 bytes of the current (or permanent) MAC
address are used as IAID. When set to "ifname", the IAID is
computed by hashing the interface name. The special value "stable"
can be used to generate an IAID based on the stable-id (see
connection.stable-id), a per-host key and the interface name. When
the property is unset, the value from global configuration is used;
if no global default is set then the IAID is assumed to be
"ifname".
For DHCPv4, the IAID is only used with "ipv4.dhcp-client-id" values
"duid" and "ipv6-duid" to generate the client-id.
For DHCPv6, note that at the moment this property is only supported
by the "internal" DHCPv6 plugin. The "dhclient" DHCPv6 plugin
always derives the IAID from the MAC address.
The actually used DHCPv6 IAID for a currently activated interface
is exposed in the lease information of the device.
Format: string

dhcp-reject-servers

Array of servers from which DHCP offers must be rejected. This property is useful to avoid getting a lease from misconfigured or rogue servers.

For DHCPv4, each element must be an IPv4 address, optionally followed by a slash and a prefix length (e.g. "192.168.122.0/24").

This property is currently not implemented for DHCPv6.

Format: array of string

dhcp-send-hostname

If TRUE, a hostname is sent to the DHCP server when acquiring a lease. Some DHCP servers use this hostname to update DNS databases, essentially providing a static hostname for the computer. If the "dhcp-hostname" property is NULL and this property is TRUE, the current persistent hostname of the computer is sent.

Format: boolean

dhcp-timeout

A timeout for a DHCP transaction in seconds. If zero (the default), a globally configured default is used. If still unspecified, a device specific timeout is used (usually 45 seconds).

Set to 2147483647 (MAXINT32) for infinity.

Format: int32

dhcp-vendor-class-identifier

The Vendor Class Identifier DHCP option (60). Special characters in the data string may be escaped using C-style escapes, nevertheless this property cannot contain nul bytes. If the per-profile value is unspecified (the default), a global connection default gets consulted. If still unspecified, the DHCP option is not sent to the server.

Format: string

dns

Array of IP addresses of DNS servers.

For DoT (DNS over TLS), the SNI server name can be specified by appending "#example.com" to the IP address of the DNS server. This currently only has effect when using systemd-resolved.

Format: array of uint32

dns-options

Array of DNS options as described in man 5 resolv.conf.

NULL means that the options are unset and left at the default. In this case NetworkManager will use default options. This is distinct from an empty list of properties.

The currently supported options are "attempts", "debug", "edns0", "inet6", "ip6-bytestring", "ip6-dotint", "ndots", "no-check-names", "no-ip6-dotint", "no-reload", "no-tld-query", "rotate", "single-request", "single-request-reopen", "timeout", "trust-ad", "use-vc".

The "trust-ad" setting is only honored if the profile contributes name servers to resolv.conf, and if all contributing profiles have "trust-ad" enabled.

When using a caching DNS plugin (dnsmasq or systemd-resolved in NetworkManager.conf) then "edns0" and "trust-ad" are automatically added.

Format: array of string

dns-priority

DNS servers priority.

The relative priority for DNS servers specified by this setting. A lower numerical value is better (higher priority).

Negative values have the special effect of excluding other configurations with a greater numerical priority value; so in presence of at least one negative priority, only DNS servers from connections with the lowest priority value will be used. To avoid all DNS leaks, set the priority of the profile that should be used to the most negative value of all active connections profiles.

Zero selects a globally configured default value. If the latter is missing or zero too, it defaults to 50 for VPNs (including WireGuard) and 100 for other connections.

Note that the priority is to order DNS settings for multiple active connections. It does not disambiguate multiple DNS servers within

the same connection profile.

When multiple devices have configurations with the same priority, VPNs will be considered first, then devices with the best (lowest metric) default route and then all other devices.

When using dns=default, servers with higher priority will be on top of resolv.conf. To prioritize a given server over another one within the same connection, just specify them in the desired order.

Note that commonly the resolver tries name servers in /etc/resolv.conf in the order listed, proceeding with the next server in the list on failure. See for example the "rotate" option of the dns-options setting. If there are any negative DNS priorities, then only name servers from the devices with that lowest priority will be considered.

When using a DNS resolver that supports Conditional Forwarding or Split DNS (with dns=dnsmasq or dns=systemd-resolved settings), each connection is used to query domains in its search list. The search domains determine which name servers to ask, and the DNS priority is used to prioritize name servers based on the domain. Queries for domains not present in any search list are routed through connections having the '~.' special wildcard domain, which is added automatically to connections with the default route (or can be added manually). When multiple connections specify the same domain, the one with the best priority (lowest numerical value) wins. If a sub domain is configured on another interface it will be accepted regardless the priority, unless parent domain on the other interface has a negative priority, which causes the sub domain to be shadowed. With Split DNS one can avoid undesired DNS leaks by properly configuring DNS priorities and the search domains, so that only name servers of the desired interface are configured.

Format: int32

dns-search

List of DNS search domains. Domains starting with a tilde ('~') are considered 'routing' domains and are used only to decide the

interface over which a query must be forwarded; they are not used to complete unqualified host names.

When using a DNS plugin that supports Conditional Forwarding or Split DNS, then the search domains specify which name servers to query. This makes the behavior different from running with plain /etc/resolv.conf. For more information see also the dns-priority setting.

When set on a profile that also enabled DHCP, the DNS search list received automatically (option 119 for DHCPv4 and option 24 for DHCPv6) gets merged with the manual list. This can be prevented by setting "ignore-auto-dns". Note that if no DNS searches are configured, the fallback will be derived from the domain from DHCP (option 15).

Format: array of string

gateway

Alias: gw4

The gateway associated with this configuration. This is only meaningful if "addresses" is also set.

Setting the gateway causes NetworkManager to configure a standard default route with the gateway as next hop. This is ignored if "never-default" is set. An alternative is to configure the default route explicitly with a manual route and /0 as prefix length.

Note that the gateway usually conflicts with routing that NetworkManager configures for WireGuard interfaces, so usually it should not be set in that case. See "ip4-auto-default-route".

Format: string

ignore-auto-dns

When "method" is set to "auto" and this property to TRUE, automatically configured name servers and search domains are ignored and only name servers and search domains specified in the "dns" and "dns-search" properties, if any, are used.

Format: boolean

ignore-auto-routes

When "method" is set to "auto" and this property to TRUE, automatically configured routes are ignored and only routes specified in the "routes" property, if any, are used.

Format: boolean

link-local

Enable and disable the IPv4 link-local configuration independently of the ipv4.method configuration. This allows a link-local address (169.254.x.y/16) to be obtained in addition to other addresses, such as those manually configured or obtained from a DHCP server. When set to "auto", the value is dependent on "ipv4.method". When set to "default", it honors the global connection default, before falling back to "auto". Note that if "ipv4.method" is "disabled", then link local addressing is always disabled too. The default is "default".

Format: int32

may-fail

If TRUE, allow overall network configuration to proceed even if the configuration specified by this property times out. Note that at least one IP configuration must succeed or overall network configuration will still fail. For example, in IPv6-only networks, setting this property to TRUE on the NMSettingIP4Config allows the overall network configuration to succeed if IPv4 configuration fails but IPv6 configuration completes successfully.

Format: boolean

method

IP configuration method.

NMSettingIP4Config and NMSettingIP6Config both support "disabled", "auto", "manual", and "link-local". See the subclass-specific documentation for other values.

In general, for the "auto" method, properties such as "dns" and "routes" specify information that is added on to the information returned from automatic configuration. The "ignore-auto-routes" and "ignore-auto-dns" properties modify this behavior.

For methods that imply no upstream network, such as "shared" or "link-local", these properties must be empty.

For IPv4 method "shared", the IP subnet can be configured by adding one manual IPv4 address or otherwise 10.42.x.0/24 is chosen. Note that the shared method must be configured on the interface which shares the internet to a subnet, not on the uplink which is shared.

Format: string

never-default

If TRUE, this connection will never be the default connection for this IP type, meaning it will never be assigned the default route by NetworkManager.

Format: boolean

replace-local-rule

Connections will default to keep the autogenerated priority 0 local rule unless this setting is set to TRUE.

Format: NMTernary (int32)

required-timeout

The minimum time interval in milliseconds for which dynamic IP configuration should be tried before the connection succeeds.

This property is useful for example if both IPv4 and IPv6 are enabled and are allowed to fail. Normally the connection succeeds as soon as one of the two address families completes; by setting a required timeout for e.g. IPv4, one can ensure that even if IP6 succeeds earlier than IPv4, NetworkManager waits some time for IPv4 before the connection becomes active.

Note that if "may-fail" is FALSE for the same address family, this property has no effect as NetworkManager needs to wait for the full DHCP timeout.

A zero value means that no required timeout is present, -1 means the default value (either configuration ipvx.required-timeout override or zero).

Format: int32

route-metric

The default metric for routes that don't explicitly specify a metric. The default value -1 means that the metric is chosen automatically based on the device type. The metric applies to dynamic routes, manual (static) routes that don't have an explicit metric setting, address prefix routes, and the default route. Note that for IPv6, the kernel accepts zero (0) but coerces it to 1024 (user default). Hence, setting this property to zero effectively mean setting it to 1024. For IPv4, zero is a regular value for the metric.

Format: int64

route-table

Enable policy routing (source routing) and set the routing table used when adding routes.

This affects all routes, including device-routes, IPv4LL, DHCP, SLAAC, default-routes and static routes. But note that static routes can individually overwrite the setting by explicitly specifying a non-zero routing table.

If the table setting is left at zero, it is eligible to be overwritten via global configuration. If the property is zero even after applying the global configuration value, policy routing is disabled for the address family of this connection.

Policy routing disabled means that NetworkManager will add all routes to the main table (except static routes that explicitly configure a different table). Additionally, NetworkManager will not delete any extraneous routes from tables except the main table.

This is to preserve backward compatibility for users who manage routing tables outside of NetworkManager.

Format: uint32

routes

A list of IPv4 destination addresses, prefix length, optional IPv4 next hop addresses, optional route metric, optional attribute. The valid syntax is: "ip[/prefix] [next-hop] [metric] [attribute=val]...[,ip[/prefix]...]". For example "192.0.2.0/24

10.1.1.1 77, 198.51.100.0/24".

Various attributes are supported:

? "advmss" - an unsigned 32 bit integer.

? "cwnd" - an unsigned 32 bit integer.

? "initcwnd" - an unsigned 32 bit integer.

? "initrwnd" - an unsigned 32 bit integer.

? "lock-advmss" - a boolean value.

? "lock-cwnd" - a boolean value.

? "lock-initcwnd" - a boolean value.

? "lock-initrwnd" - a boolean value.

? "lock-mtu" - a boolean value.

? "lock-window" - a boolean value.

? "mtu" - an unsigned 32 bit integer.

? "onlink" - a boolean value. The onlink flag is ignored for IPv4 routes without a gateway. That also means, with a positive "weight" the route cannot merge with ECMP routes which are onlink and have a gateway.

? "quickack" - a boolean value.

? "rto_min" - an unsigned 32 bit integer. The value is in milliseconds.

? "scope" - an unsigned 8 bit integer. IPv4 only.

? "src" - an IPv4 address.

? "table" - an unsigned 32 bit integer. The default depends on ipv4.route-table.

? "tos" - an unsigned 8 bit integer. IPv4 only.

? "type" - one of unicast, local, blackhole, unavailable, prohibit, throw. The default is unicast.

? "weight" - an unsigned 32 bit integer ranging from 0 to 256. A non-zero weight indicates that the IPv4 route is an ECMP IPv4 route. NetworkManager will automatically merge compatible ECMP routes into multi-hop routes. Setting to zero or omitting the attribute configures single hop routes that won't get merged. If the route finds no merge partner, it is configured as single

hop route.

Note that in NetworkManager, currently all nexthops of a ECMP route must share the same "onlink" flag in order to be mergable.

? "window" - an unsigned 32 bit integer.

For details see also `man ip-route`.

Format: a comma separated list of routes

routing-rules

A comma separated list of routing rules for policy routing. The format is based on ip rule add syntax and mostly compatible. One difference is that routing rules in NetworkManager always need a fixed priority.

Example: priority 5 from 192.167.4.0/24 table 45

Format: a comma separated list of routing rules

ipv6 setting

IPv6 Settings.

Properties:

addr-gen-mode

Configure method for creating the address for use with RFC4862 IPv6 Stateless Address Autoconfiguration. The permitted values are: NM_SETTING_IP6_CONFIG_ADDR_GEN_MODE_EUI64 (0), NM_SETTING_IP6_CONFIG_ADDR_GEN_MODE_STABLE_PRIVACY (1). NM_SETTING_IP6_CONFIG_ADDR_GEN_MODE_DEFAULT_OR_EUI64 (2) or NM_SETTING_IP6_CONFIG_ADDR_GEN_MODE_DEFAULT (3).

If the property is set to EUI64, the addresses will be generated using the interface tokens derived from hardware address. This makes the host part of the address to stay constant, making it possible to track host's presence when it changes networks. The address changes when the interface hardware is replaced.

The value of stable-privacy enables use of cryptographically secure hash of a secret host-specific key along with the connection's stable-id and the network address as specified by RFC7217. This makes it impossible to use the address track host's presence, and

makes the address stable when the network interface hardware is replaced.

The special values "default" and "default-or-eui64" will fallback to the global connection default in as documented in NetworkManager.conf(5) manual. If the global default is not specified, the fallback value is "stable-privacy" or "eui64", respectively.

For libnm, the property defaults to "default" since 1.40. Previously it defaulted to "stable-privacy". On D-Bus, the absence of an addr-gen-mode setting equals "default". For keyfile plugin, the absence of the setting on disk means "default-or-eui64" so that the property doesn't change on upgrade from older versions.

Note that this setting is distinct from the Privacy Extensions as configured by "ip6-privacy" property and it does not affect the temporary addresses configured with this option.

Format: int32

addresses

Alias: ip6

Array of IP addresses.

Format: a comma separated list of addresses

auto-route-ext-gw

VPN connections will default to add the route automatically unless this setting is set to FALSE.

For other connection types, adding such an automatic route is currently not supported and setting this to TRUE has no effect.

Format: NMTernary (int32)

dhcp-duid

A string containing the DHCPv6 Unique Identifier (DUID) used by the dhcp client to identify itself to DHCPv6 servers (RFC 3315). The DUID is carried in the Client Identifier option. If the property is a hex string ('aa:bb:cc') it is interpreted as a binary DUID and filled as an opaque value in the Client Identifier option.

The special value "lease" will retrieve the DUID previously used

from the lease file belonging to the connection. If no DUID is found and "dhclient" is the configured dhcp client, the DUID is searched in the system-wide dhclient lease file. If still no DUID is found, or another dhcp client is used, a global and permanent DUID-UUID (RFC 6355) will be generated based on the machine-id.

The special values "llt" and "ll" will generate a DUID of type LLT or LL (see RFC 3315) based on the current MAC address of the device. In order to try providing a stable DUID-LLT, the time field will contain a constant timestamp that is used globally (for all profiles) and persisted to disk.

The special values "stable-llt", "stable-ll" and "stable-uuid" will generate a DUID of the corresponding type, derived from the connection's stable-id and a per-host unique key. You may want to include the "${DEVICE}" or "${MAC}" specifier in the stable-id, in case this profile gets activated on multiple devices. So, the link-layer address of "stable-ll" and "stable-llt" will be a generated address derived from the stable id. The DUID-LLT time value in the "stable-llt" option will be picked among a static timespan of three years (the upper bound of the interval is the same constant timestamp used in "llt").

When the property is unset, the global value provided for "ipv6.dhcp-duid" is used. If no global value is provided, the default "lease" value is assumed.

Format: string

dhcp-hostname

If the "dhcp-send-hostname" property is TRUE, then the specified name will be sent to the DHCP server when acquiring a lease. This property and "dhcp-fqdn" are mutually exclusive and cannot be set at the same time.

Format: string

dhcp-hostname-flags

Flags for the DHCP hostname and FQDN.

Currently, this property only includes flags to control the FQDN

flags set in the DHCP FQDN option. Supported FQDN flags are NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1), NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) and NM_DHCP_HOSTNAME_FLAG_FQDN_NO_UPDATE (0x4). When no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is set, the DHCP FQDN option will contain no flag. Otherwise, if no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is not set, the standard FQDN flags are set in the request: NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1), NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) for IPv4 and NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1) for IPv6. When this property is set to the default value NM_DHCP_HOSTNAME_FLAG_NONE (0x0), a global default is looked up in NetworkManager configuration. If that value is unset or also NM_DHCP_HOSTNAME_FLAG_NONE (0x0), then the standard FQDN flags described above are sent in the DHCP requests.

Format: uint32

dhcp-iaid

A string containing the "Identity Association Identifier" (IAID) used by the DHCP client. The string can be a 32-bit number (either decimal, hexadecimal or or as colon separated hexadecimal numbers). Alternatively it can be set to the special values "mac", "perm-mac", "ifname" or "stable". When set to "mac" (or "perm-mac"), the last 4 bytes of the current (or permanent) MAC address are used as IAID. When set to "ifname", the IAID is computed by hashing the interface name. The special value "stable" can be used to generate an IAID based on the stable-id (see connection.stable-id), a per-host key and the interface name. When the property is unset, the value from global configuration is used; if no global default is set then the IAID is assumed to be "ifname".

For DHCPv4, the IAID is only used with "ipv4.dhcp-client-id" values "duid" and "ipv6-duid" to generate the client-id.

For DHCPv6, note that at the moment this property is only supported by the "internal" DHCPv6 plugin. The "dhclient" DHCPv6 plugin always derives the IAID from the MAC address.

The actually used DHCPv6 IAID for a currently activated interface is exposed in the lease information of the device.

Format: string

dhcp-send-hostname

If TRUE, a hostname is sent to the DHCP server when acquiring a lease. Some DHCP servers use this hostname to update DNS databases, essentially providing a static hostname for the computer. If the "dhcp-hostname" property is NULL and this property is TRUE, the current persistent hostname of the computer is sent.

Format: boolean

dhcp-timeout

A timeout for a DHCP transaction in seconds. If zero (the default), a globally configured default is used. If still unspecified, a device specific timeout is used (usually 45 seconds).

Set to 2147483647 (MAXINT32) for infinity.

Format: int32

dns

Array of IP addresses of DNS servers.

For DoT (DNS over TLS), the SNI server name can be specified by appending "#example.com" to the IP address of the DNS server. This currently only has effect when using systemd-resolved.

Format: array of byte array

dns-options

Array of DNS options as described in man 5 resolv.conf.

NULL means that the options are unset and left at the default. In this case NetworkManager will use default options. This is distinct from an empty list of properties.

The currently supported options are "attempts", "debug", "edns0", "inet6", "ip6-bytestring", "ip6-dotint", "ndots", "no-check-names", "no-ip6-dotint", "no-reload", "no-tld-query", "rotate",

"single-request", "single-request-reopen", "timeout", "trust-ad", "use-vc".

The "trust-ad" setting is only honored if the profile contributes name servers to resolv.conf, and if all contributing profiles have "trust-ad" enabled.

When using a caching DNS plugin (dnsmasq or systemd-resolved in NetworkManager.conf) then "edns0" and "trust-ad" are automatically added.

Format: array of string

dns-priority

DNS servers priority.

The relative priority for DNS servers specified by this setting. A lower numerical value is better (higher priority).

Negative values have the special effect of excluding other configurations with a greater numerical priority value; so in presence of at least one negative priority, only DNS servers from connections with the lowest priority value will be used. To avoid all DNS leaks, set the priority of the profile that should be used to the most negative value of all active connections profiles.

Zero selects a globally configured default value. If the latter is missing or zero too, it defaults to 50 for VPNs (including WireGuard) and 100 for other connections.

Note that the priority is to order DNS settings for multiple active connections. It does not disambiguate multiple DNS servers within the same connection profile.

When multiple devices have configurations with the same priority, VPNs will be considered first, then devices with the best (lowest metric) default route and then all other devices.

When using dns=default, servers with higher priority will be on top of resolv.conf. To prioritize a given server over another one within the same connection, just specify them in the desired order.

Note that commonly the resolver tries name servers in /etc/resolv.conf in the order listed, proceeding with the next

server in the list on failure. See for example the "rotate" option of the dns-options setting. If there are any negative DNS priorities, then only name servers from the devices with that lowest priority will be considered.

When using a DNS resolver that supports Conditional Forwarding or Split DNS (with dns=dnsmasq or dns=systemd-resolved settings), each connection is used to query domains in its search list. The search domains determine which name servers to ask, and the DNS priority is used to prioritize name servers based on the domain. Queries for domains not present in any search list are routed through connections having the '~.' special wildcard domain, which is added automatically to connections with the default route (or can be added manually). When multiple connections specify the same domain, the one with the best priority (lowest numerical value) wins. If a sub domain is configured on another interface it will be accepted regardless the priority, unless parent domain on the other interface has a negative priority, which causes the sub domain to be shadowed. With Split DNS one can avoid undesired DNS leaks by properly configuring DNS priorities and the search domains, so that only name servers of the desired interface are configured.

Format: int32

dns-search

List of DNS search domains. Domains starting with a tilde ('~') are considered 'routing' domains and are used only to decide the interface over which a query must be forwarded; they are not used to complete unqualified host names.

When using a DNS plugin that supports Conditional Forwarding or Split DNS, then the search domains specify which name servers to query. This makes the behavior different from running with plain /etc/resolv.conf. For more information see also the dns-priority setting.

When set on a profile that also enabled DHCP, the DNS search list received automatically (option 119 for DHCPv4 and option 24 for

DHCPv6) gets merged with the manual list. This can be prevented by setting "ignore-auto-dns". Note that if no DNS searches are configured, the fallback will be derived from the domain from DHCP (option 15).

Format: array of string

gateway

Alias: gw6

The gateway associated with this configuration. This is only meaningful if "addresses" is also set.

Setting the gateway causes NetworkManager to configure a standard default route with the gateway as next hop. This is ignored if "never-default" is set. An alternative is to configure the default route explicitly with a manual route and /0 as prefix length.

Note that the gateway usually conflicts with routing that NetworkManager configures for WireGuard interfaces, so usually it should not be set in that case. See "ip4-auto-default-route".

Format: string

ignore-auto-dns

When "method" is set to "auto" and this property to TRUE, automatically configured name servers and search domains are ignored and only name servers and search domains specified in the "dns" and "dns-search" properties, if any, are used.

Format: boolean

ignore-auto-routes

When "method" is set to "auto" and this property to TRUE, automatically configured routes are ignored and only routes specified in the "routes" property, if any, are used.

Format: boolean

ip6-privacy

Configure IPv6 Privacy Extensions for SLAAC, described in RFC4941. If enabled, it makes the kernel generate a temporary IPv6 address in addition to the public one generated from MAC address via modified EUI-64. This enhances privacy, but could cause problems in

some applications, on the other hand. The permitted values are: -1: unknown, 0: disabled, 1: enabled (prefer public address), 2: enabled (prefer temporary addresses).

Having a per-connection setting set to "-1" (unknown) means fallback to global configuration "ipv6.ip6-privacy".

If also global configuration is unspecified or set to "-1", fallback to read "/proc/sys/net/ipv6/conf/default/use_tempaddr".

Note that this setting is distinct from the Stable Privacy addresses that can be enabled with the "addr-gen-mode" property's "stable-privacy" setting as another way of avoiding host tracking with IPv6 addresses.

Format: NMSettingIP6ConfigPrivacy (int32)

may-fail

If TRUE, allow overall network configuration to proceed even if the configuration specified by this property times out. Note that at least one IP configuration must succeed or overall network configuration will still fail. For example, in IPv6-only networks, setting this property to TRUE on the NMSettingIP4Config allows the overall network configuration to succeed if IPv4 configuration fails but IPv6 configuration completes successfully.

Format: boolean

method

IP configuration method.

NMSettingIP4Config and NMSettingIP6Config both support "disabled", "auto", "manual", and "link-local". See the subclass-specific documentation for other values.

In general, for the "auto" method, properties such as "dns" and "routes" specify information that is added on to the information returned from automatic configuration. The "ignore-auto-routes" and "ignore-auto-dns" properties modify this behavior.

For methods that imply no upstream network, such as "shared" or "link-local", these properties must be empty.

For IPv4 method "shared", the IP subnet can be configured by adding

one manual IPv4 address or otherwise 10.42.x.0/24 is chosen. Note that the shared method must be configured on the interface which shares the internet to a subnet, not on the uplink which is shared.

Format: string

mtu

Maximum transmission unit size, in bytes. If zero (the default), the MTU is set automatically from router advertisements or is left equal to the link-layer MTU. If greater than the link-layer MTU, or greater than zero but less than the minimum IPv6 MTU of 1280, this value has no effect.

Format: uint32

never-default

If TRUE, this connection will never be the default connection for this IP type, meaning it will never be assigned the default route by NetworkManager.

Format: boolean

ra-timeout

A timeout for waiting Router Advertisements in seconds. If zero (the default), a globally configured default is used. If still unspecified, the timeout depends on the sysctl settings of the device.

Set to 2147483647 (MAXINT32) for infinity.

Format: int32

replace-local-rule

Connections will default to keep the autogenerated priority 0 local rule unless this setting is set to TRUE.

Format: NMTernary (int32)

required-timeout

The minimum time interval in milliseconds for which dynamic IP configuration should be tried before the connection succeeds.

This property is useful for example if both IPv4 and IPv6 are enabled and are allowed to fail. Normally the connection succeeds as soon as one of the two address families completes; by setting a

required timeout for e.g. IPv4, one can ensure that even if IP6 succeeds earlier than IPv4, NetworkManager waits some time for IPv4 before the connection becomes active.

Note that if "may-fail" is FALSE for the same address family, this property has no effect as NetworkManager needs to wait for the full DHCP timeout.

A zero value means that no required timeout is present, -1 means the default value (either configuration ipvx.required-timeout override or zero).

Format: int32

route-metric

The default metric for routes that don't explicitly specify a metric. The default value -1 means that the metric is chosen automatically based on the device type. The metric applies to dynamic routes, manual (static) routes that don't have an explicit metric setting, address prefix routes, and the default route. Note that for IPv6, the kernel accepts zero (0) but coerces it to 1024 (user default). Hence, setting this property to zero effectively mean setting it to 1024. For IPv4, zero is a regular value for the metric.

Format: int64

route-table

Enable policy routing (source routing) and set the routing table used when adding routes.

This affects all routes, including device-routes, IPv4LL, DHCP, SLAAC, default-routes and static routes. But note that static routes can individually overwrite the setting by explicitly specifying a non-zero routing table.

If the table setting is left at zero, it is eligible to be overwritten via global configuration. If the property is zero even after applying the global configuration value, policy routing is disabled for the address family of this connection.

Policy routing disabled means that NetworkManager will add all

routes to the main table (except static routes that explicitly configure a different table). Additionally, NetworkManager will not delete any extraneous routes from tables except the main table. This is to preserve backward compatibility for users who manage routing tables outside of NetworkManager.

Format: uint32

routes

A list of IPv6 destination addresses, prefix length, optional IPv6 next hop addresses, optional route metric, optional attribute. The valid syntax is: "ip[/prefix] [next-hop] [metric] [attribute=val]...[,ip[/prefix]...]".

Various attributes are supported:

- "advmss" - an unsigned 32 bit integer.
- "cwnd" - an unsigned 32 bit integer.
- "from" - an IPv6 address with optional prefix. IPv6 only.
- "initcwnd" - an unsigned 32 bit integer.
- "initrwnd" - an unsigned 32 bit integer.
- "lock-advmss" - a boolean value.
- "lock-cwnd" - a boolean value.
- "lock-initcwnd" - a boolean value.
- "lock-initrwnd" - a boolean value.
- "lock-mtu" - a boolean value.
- "lock-window" - a boolean value.
- "mtu" - an unsigned 32 bit integer.
- "onlink" - a boolean value.
- "quickack" - a boolean value.
- "rto_min" - an unsigned 32 bit integer. The value is in milliseconds.
- "src" - an IPv6 address.
- "table" - an unsigned 32 bit integer. The default depends on ipv6.route-table.
- "type" - one of unicast, local, blackhole, unavailable, prohibit, throw. The default is unicast.

? "window" - an unsigned 32 bit integer.

For details see also `man ip-route`.

Format: a comma separated list of routes

routing-rules

A comma separated list of routing rules for policy routing. The format is based on ip rule add syntax and mostly compatible. One difference is that routing rules in NetworkManager always need a fixed priority.

Example: priority 5 from 1:2:3::5/128 table 45

Format: a comma separated list of routing rules

token

Configure the token for draft-chown-6man-tokenised-ipv6-identifiers-02 IPv6 tokenized interface identifiers. Useful with eui64 addr-gen-mode.

Format: string

ip-tunnel setting

IP Tunneling Settings.

Properties:

encapsulation-limit

How many additional levels of encapsulation are permitted to be prepended to packets. This property applies only to IPv6 tunnels.

Format: uint32

flags

Tunnel flags. Currently, the following values are supported: NM_IP_TUNNEL_FLAG_IP6_IGN_ENCAP_LIMIT (0x1), NM_IP_TUNNEL_FLAG_IP6_USE_ORIG_TCLASS (0x2), NM_IP_TUNNEL_FLAG_IP6_USE_ORIG_FLOWLABEL (0x4), NM_IP_TUNNEL_FLAG_IP6_MIP6_DEV (0x8), NM_IP_TUNNEL_FLAG_IP6_RCV_DSCP_COPY (0x10), NM_IP_TUNNEL_FLAG_IP6_USE_ORIG_FWMARK (0x20). They are valid only for IPv6 tunnels.

Format: uint32

flow-label

The flow label to assign to tunnel packets. This property applies only to IPv6 tunnels.

Format: uint32

fwmark

The fwmark value to assign to tunnel packets. This property can be set to a non zero value only on VTI and VTI6 tunnels.

Format: uint32

input-key

The key used for tunnel input packets; the property is valid only for certain tunnel modes (GRE, IP6GRE). If empty, no key is used.

Format: string

local

Alias: local

The local endpoint of the tunnel; the value can be empty, otherwise it must contain an IPv4 or IPv6 address.

Format: string

mode

Alias: mode

The tunneling mode, for example NM_IP_TUNNEL_MODE_IPIP (1) or NM_IP_TUNNEL_MODE_GRE (2).

Format: uint32

mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple fragments.

Format: uint32

output-key

The key used for tunnel output packets; the property is valid only for certain tunnel modes (GRE, IP6GRE). If empty, no key is used.

Format: string

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID the new device will be bound to so that tunneled packets will

only be routed via that interface.

Format: string

path-mtu-discovery

Whether to enable Path MTU Discovery on this tunnel.

Format: boolean

remote

Alias: remote

The remote endpoint of the tunnel; the value must contain an IPv4

or IPv6 address.

Format: string

tos

The type of service (IPv4) or traffic class (IPv6) field to be set

on tunneled packets.

Format: uint32

ttl

The TTL to assign to tunneled packets. 0 is a special value meaning

that packets inherit the TTL value.

Format: uint32

macsec setting

MACSec Settings.

Properties:

encrypt

Alias: encrypt

Whether the transmitted traffic must be encrypted.

Format: boolean

mka-cak

Alias: cak

The pre-shared CAK (Connectivity Association Key) for MACsec Key

Agreement. Must be a string of 32 hexadecimal characters.

Format: string

mka-cak-flags

Flags indicating how to handle the "mka-cak" property.

Format: NMSettingSecretFlags (uint32)

mka-ckn

Alias: ckn

The pre-shared CKN (Connectivity-association Key Name) for MACsec Key Agreement. Must be a string of hexadecimal characters with a even length between 2 and 64.

Format: string

mode

Alias: mode

Specifies how the CAK (Connectivity Association Key) for MKA (MACsec Key Agreement) is obtained.

Format: int32

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this MACSEC interface should be created. If this property is not specified, the connection must contain an "802-3-ethernet" setting with a "mac-address" property.

Format: string

port

Alias: port

The port component of the SCI (Secure Channel Identifier), between 1 and 65534.

Format: int32

send-sci

Specifies whether the SCI (Secure Channel Identifier) is included in every packet.

Format: boolean

validation

Specifies the validation mode for incoming frames.

Format: int32

macvlan setting

MAC VLAN Settings.

Properties:

mode

Alias: mode

The macvlan mode, which specifies the communication mechanism between multiple macvlans on the same lower device.

Format: uint32

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this MAC-VLAN interface should be created. If this property is not specified, the connection must contain an "802-3-ethernet" setting with a "mac-address" property.

Format: string

promiscuous

Whether the interface should be put in promiscuous mode.

Format: boolean

tap

Alias: tap

Whether the interface should be a MACVTAP.

Format: boolean

match setting

Match settings.

Properties:

driver

A list of driver names to match. Each element is a shell wildcard pattern.

See NMSettingMatch:interface-name for how special characters '|', '&', '!' and '\\' are used for optional and mandatory matches and inverting the pattern.

Format: array of string

interface-name

A list of interface names to match. Each element is a shell wildcard pattern.

An element can be prefixed with a pipe symbol (|) or an ampersand

(&). The former means that the element is optional and the latter means that it is mandatory. If there are any optional elements, than the match evaluates to true if at least one of the optional element matches (logical OR). If there are any mandatory elements, then they all must match (logical AND). By default, an element is optional. This means that an element "foo" behaves the same as "|foo". An element can also be inverted with exclamation mark (!) between the pipe symbol (or the ampersand) and before the pattern. Note that "!foo" is a shortcut for the mandatory match "&!foo". Finally, a backslash can be used at the beginning of the element (after the optional special characters) to escape the start of the pattern. For example, "&\\!a" is an mandatory match for literally "!a".

Format: array of string

kernel-command-line

A list of kernel command line arguments to match. This may be used to check whether a specific kernel command line option is set (or unset, if prefixed with the exclamation mark). The argument must either be a single word, or an assignment (i.e. two words, joined by "="). In the former case the kernel command line is searched for the word appearing as is, or as left hand side of an assignment. In the latter case, the exact assignment is looked for with right and left hand side matching. Wildcard patterns are not supported. See NMSettingMatch:interface-name for how special characters '|', '&', '!' and '\\' are used for optional and mandatory matches and inverting the match.

Format: array of string

path

A list of paths to match against the ID_PATH udev property of devices. ID_PATH represents the topological persistent path of a device. It typically contains a subsystem string (pci, usb, platform, etc.) and a subsystem-specific identifier.

For PCI devices the path has the form

"pci-$domain:$bus:$device.$function", where each variable is an hexadecimal value; for example "pci-0000:0a:00.0".
The path of a device can be obtained with "udevadm info /sys/class/net/$dev | grep ID_PATH=" or by looking at the "path" property exported by NetworkManager ("nmcli -f general.path device show $dev").
Each element of the list is a shell wildcard pattern.
See NMSettingMatch:interface-name for how special characters '|', '&', '!' and '\\' are used for optional and mandatory matches and inverting the pattern.
Format: array of string

802-11-olpc-mesh setting

Alias: olpc-mesh

OLPC Wireless Mesh Settings.

Properties:

channel

Alias: channel

Channel on which the mesh network to join is located.

Format: uint32

dhcp-anycast-address

Alias: dhcp-anycast

Anycast DHCP MAC address used when requesting an IP address via DHCP. The specific anycast address used determines which DHCP server class answers the request.
This is currently only implemented by dhclient DHCP plugin.

Format: byte array

ssid

Alias: ssid

SSID of the mesh network to join.

Format: byte array

ovs-bridge setting

OvsBridge Link Settings.

Properties:

datapath-type

The data path type. One of "system", "netdev" or empty.

Format: string

fail-mode

The bridge failure mode. One of "secure", "standalone" or empty.

Format: string

mcast-snooping-enable

Enable or disable multicast snooping.

Format: boolean

rstp-enable

Enable or disable RSTP.

Format: boolean

stp-enable

Enable or disable STP.

Format: boolean

ovs-dpdk setting

OvsDpdk Link Settings.

Properties:

devargs

Open vSwitch DPDK device arguments.

Format: string

n-rxq

Open vSwitch DPDK number of rx queues. Defaults to zero which means to leave the parameter in OVS unspecified and effectively configures one queue.

Format: uint32

n-rxq-desc

The rx queue size (number of rx descriptors) for DPDK ports. Must be zero or a power of 2 between 1 and 4096, and supported by the hardware. Defaults to zero which means to leave the parameter in OVS unspecified and effectively configures 2048 descriptors.

Format: uint32

n-txq-desc

The tx queue size (number of tx descriptors) for DPDK ports. Must be zero or a power of 2 between 1 and 4096, and supported by the hardware. Defaults to zero which means to leave the parameter in OVS unspecified and effectively configures 2048 descriptors.

Format: uint32

ovs-interface setting

Open vSwitch Interface Settings.

Properties:

ofport-request

Open vSwitch openflow port number. Defaults to zero which means that port number will not be specified and it will be chosen randomly by ovs. OpenFlow ports are the network interfaces for passing packets between OpenFlow processing and the rest of the network. OpenFlow switches connect logically to each other via their OpenFlow ports.

Format: uint32

type

The interface type. Either "internal", "system", "patch", "dpdk", or empty.

Format: string

ovs-patch setting

OvsPatch Link Settings.

Properties:

peer

Specifies the name of the interface for the other side of the patch. The patch on the other side must also set this interface as peer.

Format: string

ovs-port setting

OvsPort Link Settings.

Properties:

bond-downdelay

The time port must be inactive in order to be considered down.

Format: uint32

bond-mode

Bonding mode. One of "active-backup", "balance-slb", or

"balance-tcp".

Format: string

bond-updelay

The time port must be active before it starts forwarding traffic.

Format: uint32

lacp

LACP mode. One of "active", "off", or "passive".

Format: string

tag

The VLAN tag in the range 0-4095.

Format: uint32

trunks

A list of VLAN ranges that this port trunks.

The property is valid only for ports with mode "trunk",

"native-tagged", or "native-untagged port". If it is empty, the

port trunks all VLANs.

Format: array of vardict

vlan-mode

The VLAN mode. One of "access", "native-tagged", "native-untagged",

"trunk", "dot1q-tunnel" or unset.

Format: string

ppp setting

Point-to-Point Protocol Settings.

Properties:

baud

If non-zero, instruct pppd to set the serial port to the specified

baudrate. This value should normally be left as 0 to automatically

choose the speed.

Format: uint32

crtscts

If TRUE, specify that pppd should set the serial port to use hardware flow control with RTS and CTS signals. This value should normally be set to FALSE.

Format: boolean

lcp-echo-failure

If non-zero, instruct pppd to presume the connection to the peer has failed if the specified number of LCP echo-requests go unanswered by the peer. The "lcp-echo-interval" property must also be set to a non-zero value if this property is used.

Format: uint32

lcp-echo-interval

If non-zero, instruct pppd to send an LCP echo-request frame to the peer every n seconds (where n is the specified value). Note that some PPP peers will respond to echo requests and some will not, and it is not possible to autodetect this.

Format: uint32

mppe-stateful

If TRUE, stateful MPPE is used. See pppd documentation for more information on stateful MPPE.

Format: boolean

mru

If non-zero, instruct pppd to request that the peer send packets no larger than the specified size. If non-zero, the MRU should be between 128 and 16384.

Format: uint32

mtu

If non-zero, instruct pppd to send packets no larger than the specified size.

Format: uint32

no-vj-comp

If TRUE, Van Jacobsen TCP header compression will not be requested.

Format: boolean

noauth

If TRUE, do not require the other side (usually the PPP server) to authenticate itself to the client. If FALSE, require authentication from the remote side. In almost all cases, this should be TRUE.

Format: boolean

nobsdcomp

If TRUE, BSD compression will not be requested.

Format: boolean

nodeflate

If TRUE, "deflate" compression will not be requested.

Format: boolean

refuse-chap

If TRUE, the CHAP authentication method will not be used.

Format: boolean

refuse-eap

If TRUE, the EAP authentication method will not be used.

Format: boolean

refuse-mschap

If TRUE, the MSCHAP authentication method will not be used.

Format: boolean

refuse-mschapv2

If TRUE, the MSCHAPv2 authentication method will not be used.

Format: boolean

refuse-pap

If TRUE, the PAP authentication method will not be used.

Format: boolean

require-mppe

If TRUE, MPPE (Microsoft Point-to-Point Encryption) will be required for the PPP session. If either 64-bit or 128-bit MPPE is not available the session will fail. Note that MPPE is not used on mobile broadband connections.

Format: boolean

require-mppe-128

If TRUE, 128-bit MPPE (Microsoft Point-to-Point Encryption) will be

required for the PPP session, and the "require-mppe" property must also be set to TRUE. If 128-bit MPPE is not available the session will fail.

Format: boolean

pppoe setting

PPP-over-Ethernet Settings.

Properties:

parent

Alias: parent

If given, specifies the parent interface name on which this PPPoE connection should be created. If this property is not specified, the connection is activated on the interface specified in "interface-name" of NMSettingConnection.

Format: string

password

Alias: password

Password used to authenticate with the PPPoE service.

Format: string

password-flags

Flags indicating how to handle the "password" property.

Format: NMSettingSecretFlags (uint32)

service

Alias: service

If specified, instruct PPPoE to only initiate sessions with access concentrators that provide the specified service. For most providers, this should be left blank. It is only required if there are multiple access concentrators or a specific service is known to be required.

Format: string

username

Alias: username

Username used to authenticate with the PPPoE service.

Format: string

proxy setting

WWW Proxy Settings.

Properties:

browser-only

Alias: browser-only

Whether the proxy configuration is for browser only.

Format: boolean

method

Alias: method

Method for proxy configuration, Default is

NM_SETTING_PROXY_METHOD_NONE (0)

Format: int32

pac-script

Alias: pac-script

PAC script for the connection. This is an UTF-8 encoded javascript

code that defines a FindProxyForURL() function.

Format: string

pac-url

Alias: pac-url

PAC URL for obtaining PAC file.

Format: string

serial setting

Serial Link Settings.

Properties:

baud

Speed to use for communication over the serial port. Note that this

value usually has no effect for mobile broadband modems as they

generally ignore speed settings and use the highest available

speed.

Format: uint32

bits

Byte-width of the serial communication. The 8 in "8n1" for example.

Format: uint32

parity

Parity setting of the serial port.

Format: NMSettingSerialParity (byte)

send-delay

Time to delay between each byte sent to the modem, in microseconds.

Format: uint64

stopbits

Number of stop bits for communication on the serial port. Either 1

or 2. The 1 in "8n1" for example.

Format: uint32

sriov setting

SR-IOV settings.

Properties:

autoprobe-drivers

Whether to autoprobe virtual functions by a compatible driver.

If set to NM_TERNARY_TRUE (1), the kernel will try to bind VFs to a

compatible driver and if this succeeds a new network interface will

be instantiated for each VF.

If set to NM_TERNARY_FALSE (0), VFs will not be claimed and no

network interfaces will be created for them.

When set to NM_TERNARY_DEFAULT (-1), the global default is used; in

case the global default is unspecified it is assumed to be

NM_TERNARY_TRUE (1).

Format: NMTernary (int32)

total-vfs

The total number of virtual functions to create.

Note that when the sriov setting is present NetworkManager enforces

the number of virtual functions on the interface (also when it is

zero) during activation and resets it upon deactivation. To prevent

any changes to SR-IOV parameters don't add a sriov setting to the

connection.

Format: uint32

vfs

Array of virtual function descriptors.

Each VF descriptor is a dictionary mapping attribute names to GVariant values. The 'index' entry is mandatory for each VF.

When represented as string a VF is in the form:

"INDEX [ATTR=VALUE[ATTR=VALUE]...]".

for example:

"2 mac=00:11:22:33:44:55 spoof-check=true".

Multiple VFs can be specified using a comma as separator.

Currently, the following attributes are supported: mac, spoof-check, trust, min-tx-rate, max-tx-rate, vlans.

The "vlans" attribute is represented as a semicolon-separated list of VLAN descriptors, where each descriptor has the form "ID[.PRIORITY[.PROTO]]".

PROTO can be either 'q' for 802.1Q (the default) or 'ad' for 802.1ad.

Format: array of vardict

tc setting

Linux Traffic Control Settings.

Properties:

qdiscs

Array of TC queueing disciplines. qdisc is a basic block in the Linux traffic control subsystem

Each qdisc can be specified by the following attributes:

handle HANDLE

specifies the qdisc handle. A qdisc, which potentially can have children, gets assigned a major number, called a 'handle', leaving the minor number namespace available for classes. The handle is expressed as '10:'. It is customary to explicitly assign a handle to qdiscs expected to have children.

parent HANDLE

specifies the handle of the parent qdisc the current qdisc must be attached to.

root

specifies that the qdisc is attached to the root of device.

KIND

this is the qdisc kind. NetworkManager currently supports the following kinds: fq_codel, sfq, tbf. Each qdisc kind has a different set of parameters, described below. There are also some kinds like pfifo, pfifo_fast, prio supported by NetworkManager but their parameters are not supported by NetworkManager.

Parameters for 'fq_codel':

limit U32

the hard limit on the real queue size. When this limit is reached, incoming packets are dropped. Default is 10240 packets.

memory_limit U32

sets a limit on the total number of bytes that can be queued in this FQ-CoDel instance. The lower of the packet limit of the limit parameter and the memory limit will be enforced. Default is 32 MB.

flows U32

the number of flows into which the incoming packets are classified. Due to the stochastic nature of hashing, multiple flows may end up being hashed into the same slot. Newer flows have priority over older ones. This parameter can be set only at load time since memory has to be allocated for the hash table. Default value is 1024.

target U32

the acceptable minimum standing/persistent queue delay. This minimum delay is identified by tracking the local minimum queue delay that packets experience. The unit of measurement is microsecond(us). Default value is 5ms.

interval U32

used to ensure that the measured minimum delay does not become too stale. The minimum delay must be experienced in the last

epoch of length .B interval. It should be set on the order of the worst-case RTT through the bottleneck to give endpoints sufficient time to react. Default value is 100ms.

quantum U32

the number of bytes used as 'deficit' in the fair queuing algorithm. Default is set to 1514 bytes which corresponds to the Ethernet MTU plus the hardware header length of 14 bytes.

ecn BOOL

can be used to mark packets instead of dropping them. ecn is turned on by default.

ce_threshold U32

sets a threshold above which all packets are marked with ECN Congestion Experienced. This is useful for DCTCP-style congestion control algorithms that require marking at very shallow queueing thresholds.

Parameters for 'sfq':

divisor U32

can be used to set a different hash table size, available from kernel 2.6.39 onwards. The specified divisor must be a power of two and cannot be larger than 65536. Default value: 1024.

limit U32

Upper limit of the SFQ. Can be used to reduce the default length of 127 packets.

depth U32

Limit of packets per flow. Default to 127 and can be lowered.

perturb_period U32

Interval in seconds for queue algorithm perturbation. Defaults to 0, which means that no perturbation occurs. Do not set too low for each perturbation may cause some packet reordering or losses. Advised value: 60 This value has no effect when external flow classification is used. Its better to increase divisor value to lower risk of hash collisions.

quantum U32

Amount of bytes a flow is allowed to dequeue during a round of the round robin process. Defaults to the MTU of the interface which is also the advised value and the minimum value.

flows U32

Default value is 127.

Parameters for 'tbf':

rate U64

Bandwidth or rate. These parameters accept a floating point number, possibly followed by either a unit (both SI and IEC units supported), or a float followed by a percent character to specify the rate as a percentage of the device's speed.

burst U32

Also known as buffer or maxburst. Size of the bucket, in bytes. This is the maximum amount of bytes that tokens can be available for instantaneously. In general, larger shaping rates require a larger buffer. For 10mbit/s on Intel, you need at least 10kbyte buffer if you want to reach your configured rate! If your buffer is too small, packets may be dropped because more tokens arrive per timer tick than fit in your bucket. The minimum buffer size can be calculated by dividing the rate by HZ.

Token usage calculations are performed using a table which by default has a resolution of 8 packets. This resolution can be changed by specifying the cell size with the burst. For example, to specify a 6000 byte buffer with a 16 byte cell size, set a burst of 6000/16. You will probably never have to set this. Must be an integral power of 2.

limit U32

Limit is the number of bytes that can be queued waiting for tokens to become available.

latency U32

specifies the maximum amount of time a packet can sit in the TBF. The latency calculation takes into account the size of the

bucket, the rate and possibly the peakrate (if set). The latency and limit are mutually exclusive.

Format: GPtrArray(NMTCQdisc)

tfilters

Array of TC traffic filters. Traffic control can manage the packet content during classification by using filters.

Each tfilters can be specified by the following attributes:

handle HANDLE

specifies the tfilters handle. A filter is used by a classful qdisc to determine in which class a packet will be enqueued. It is important to notice that filters reside within qdiscs. Therefore, see qdiscs handle for detailed information.

parent HANDLE

specifies the handle of the parent qdisc the current qdisc must be attached to.

root

specifies that the qdisc is attached to the root of device.

KIND

this is the tfilters kind. NetworkManager currently supports following kinds: mirred, simple. Each filter kind has a different set of actions, described below. There are also some other kinds like matchall, basic, u32 supported by NetworkManager.

Actions for 'mirred':

egress bool

Define whether the packet should exit from the interface.

ingress bool

Define whether the packet should come into the interface.

mirror bool

Define whether the packet should be copied to the destination space.

redirect bool

Define whether the packet should be moved to the destination

space.

Action for 'simple':

sdata char[32]

The actual string to print.

Format: GPtrArray(NMTCTfilter)

team setting

Teaming Settings.

Properties:

config

Alias: config

The JSON configuration for the team network interface. The property should contain raw JSON configuration data suitable for teamd, because the value is passed directly to teamd. If not specified, the default configuration is used. See man teamd.conf for the format details.

Format: string

link-watchers

Link watchers configuration for the connection: each link watcher is defined by a dictionary, whose keys depend upon the selected link watcher. Available link watchers are 'ethtool', 'nsna_ping' and 'arp_ping' and it is specified in the dictionary with the key 'name'. Available keys are: ethtool: 'delay-up', 'delay-down', 'init-wait'; nsna_ping: 'init-wait', 'interval', 'missed-max', 'target-host'; arp_ping: all the ones in nsna_ping and 'source-host', 'validate-active', 'validate-inactive', 'send-always'. See teamd.conf man for more details.

Format: array of vardict

mcast-rejoin-count

Corresponds to the teamd mcast_rejoin.count.

Format: int32

mcast-rejoin-interval

Corresponds to the teamd mcast_rejoin.interval.

Format: int32

notify-peers-count

Corresponds to the teamd notify_peers.count.

Format: int32

notify-peers-interval

Corresponds to the teamd notify_peers.interval.

Format: int32

runner

Corresponds to the teamd runner.name. Permitted values are:

"roundrobin", "broadcast", "activebackup", "loadbalance", "lacp",

"random".

Format: string

runner-active

Corresponds to the teamd runner.active.

Format: boolean

runner-agg-select-policy

Corresponds to the teamd runner.agg_select_policy.

Format: string

runner-fast-rate

Corresponds to the teamd runner.fast_rate.

Format: boolean

runner-hwaddr-policy

Corresponds to the teamd runner.hwaddr_policy.

Format: string

runner-min-ports

Corresponds to the teamd runner.min_ports.

Format: int32

runner-sys-prio

Corresponds to the teamd runner.sys_prio.

Format: int32

runner-tx-balancer

Corresponds to the teamd runner.tx_balancer.name.

Format: string

runner-tx-balancer-interval

Corresponds to the teamd runner.tx_balancer.interval.

Format: int32

runner-tx-hash

Corresponds to the teamd runner.tx_hash.

Format: array of string

team-port setting

Team Port Settings.

Properties:

config

Alias: config

The JSON configuration for the team port. The property should contain raw JSON configuration data suitable for teamd, because the value is passed directly to teamd. If not specified, the default configuration is used. See man teamd.conf for the format details.

Format: string

lacp-key

Corresponds to the teamd ports.PORTIFNAME.lacp_key.

Format: int32

lacp-prio

Corresponds to the teamd ports.PORTIFNAME.lacp_prio.

Format: int32

link-watchers

Link watchers configuration for the connection: each link watcher is defined by a dictionary, whose keys depend upon the selected link watcher. Available link watchers are 'ethtool', 'nsna_ping' and 'arp_ping' and it is specified in the dictionary with the key 'name'. Available keys are: ethtool: 'delay-up', 'delay-down', 'init-wait'; nsna_ping: 'init-wait', 'interval', 'missed-max', 'target-host'; arp_ping: all the ones in nsna_ping and 'source-host', 'validate-active', 'validate-inactive', 'send-always'. See teamd.conf man for more details.

Format: array of vardict

prio

Corresponds to the teamd ports.PORTIFNAME.prio.

Format: int32

queue-id

Corresponds to the teamd ports.PORTIFNAME.queue_id. When set to -1

means the parameter is skipped from the json config.

Format: int32

sticky

Corresponds to the teamd ports.PORTIFNAME.sticky.

Format: boolean

tun setting

Tunnel Settings.

Properties:

group

Alias: group

The group ID which will own the device. If set to NULL everyone

will be able to use the device.

Format: string

mode

Alias: mode

The operating mode of the virtual device. Allowed values are

NM_SETTING_TUN_MODE_TUN (1) to create a layer 3 device and

NM_SETTING_TUN_MODE_TAP (2) to create an Ethernet-like layer 2 one.

Format: uint32

multi-queue

Alias: multi-queue

If the property is set to TRUE, the interface will support multiple

file descriptors (queues) to parallelize packet sending or

receiving. Otherwise, the interface will only support a single

queue.

Format: boolean

owner

Alias: owner

The user ID which will own the device. If set to NULL everyone will

be able to use the device.

Format: string

pi

Alias: pi

If TRUE the interface will prepend a 4 byte header describing the

physical interface to the packets.

Format: boolean

vnet-hdr

Alias: vnet-hdr

If TRUE the IFF_VNET_HDR the tunnel packets will include a virtio

network header.

Format: boolean

vlan setting

VLAN Settings.

Properties:

egress-priority-map

Alias: egress

For outgoing packets, a list of mappings from Linux SKB priorities

to 802.1p priorities. The mapping is given in the format "from:to"

where both "from" and "to" are unsigned integers, ie "7:3".

Format: array of string

flags

Alias: flags

One or more flags which control the behavior and features of the

VLAN interface. Flags include NM_VLAN_FLAG_REORDER_HEADERS (0x1)

(reordering of output packet headers), NM_VLAN_FLAG_GVRP (0x2) (use

of the GVRP protocol), and NM_VLAN_FLAG_LOOSE_BINDING (0x4) (loose

binding of the interface to its master device's operating state).

NM_VLAN_FLAG_MVRP (0x8) (use of the MVRP protocol).

The default value of this property is NM_VLAN_FLAG_REORDER_HEADERS,

but it used to be 0. To preserve backward compatibility, the

default-value in the D-Bus API continues to be 0 and a missing

property on D-Bus is still considered as 0.

Format: NMVlanFlags (uint32)

id

Alias: id

The VLAN identifier that the interface created by this connection should be assigned. The valid range is from 0 to 4094, without the reserved id 4095.

Format: uint32

ingress-priority-map

Alias: ingress

For incoming packets, a list of mappings from 802.1p priorities to Linux SKB priorities. The mapping is given in the format "from:to" where both "from" and "to" are unsigned integers, ie "7:3".

Format: array of string

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this VLAN interface should be created. If this property is not specified, the connection must contain an "802-3-ethernet" setting with a "mac-address" property.

Format: string

protocol

Specifies the VLAN protocol to use for encapsulation. Supported values are: '802.1Q', '802.1ad'. If not specified the default value is '802.1Q'.

Format: string

vpn setting

VPN Settings.

Properties:

data

Dictionary of key/value pairs of VPN plugin specific data. Both keys and values must be strings.

Format: dict of string to string

persistent

If the VPN service supports persistence, and this property is TRUE, the VPN will attempt to stay connected across link changes and outages, until explicitly disconnected.

Format: boolean

secrets

Dictionary of key/value pairs of VPN plugin specific secrets like passwords or private keys. Both keys and values must be strings.

Format: dict of string to string

service-type

Alias: vpn-type

D-Bus service name of the VPN plugin that this setting uses to connect to its network. i.e. org.freedesktop.NetworkManager.vpnc for the vpnc plugin.

Format: string

timeout

Timeout for the VPN service to establish the connection. Some services may take quite a long time to connect. Value of 0 means a default timeout, which is 60 seconds (unless overridden by vpn.timeout in configuration file). Values greater than zero mean timeout in seconds.

Format: uint32

user-name

Alias: user

If the VPN connection requires a user name for authentication, that name should be provided here. If the connection is available to more than one user, and the VPN requires each user to supply a different name, then leave this property empty. If this property is empty, NetworkManager will automatically supply the username of the user which requested the VPN connection.

Format: string

vrf setting

VRF settings.

Properties:

table

Alias: table

The routing table for this VRF.

Format: uint32

vxlan setting

VXLAN Settings.

Properties:

ageing

Specifies the lifetime in seconds of FDB entries learnt by the

kernel.

Format: uint32

destination-port

Alias: destination-port

Specifies the UDP destination port to communicate to the remote

VXLAN tunnel endpoint.

Format: uint32

id

Alias: id

Specifies the VXLAN Network Identifier (or VXLAN Segment

Identifier) to use.

Format: uint32

l2-miss

Specifies whether netlink LL ADDR miss notifications are generated.

Format: boolean

l3-miss

Specifies whether netlink IP ADDR miss notifications are generated.

Format: boolean

learning

Specifies whether unknown source link layer addresses and IP

addresses are entered into the VXLAN device forwarding database.

Format: boolean

limit

Specifies the maximum number of FDB entries. A value of zero means

that the kernel will store unlimited entries.

Format: uint32

local

Alias: local

If given, specifies the source IP address to use in outgoing

packets.

Format: string

parent

Alias: dev

If given, specifies the parent interface name or parent connection

UUID.

Format: string

proxy

Specifies whether ARP proxy is turned on.

Format: boolean

remote

Alias: remote

Specifies the unicast destination IP address to use in outgoing

packets when the destination link layer address is not known in the

VXLAN device forwarding database, or the multicast IP address to

join.

Format: string

rsc

Specifies whether route short circuit is turned on.

Format: boolean

source-port-max

Alias: source-port-max

Specifies the maximum UDP source port to communicate to the remote

VXLAN tunnel endpoint.

Format: uint32

source-port-min

Alias: source-port-min

Specifies the minimum UDP source port to communicate to the remote

VXLAN tunnel endpoint.

Format: uint32

tos

Specifies the TOS value to use in outgoing packets.

Format: uint32

ttl

Specifies the time-to-live value to use in outgoing packets.

Format: uint32

wifi-p2p setting

Wi-Fi P2P Settings.

Properties:

peer

Alias: peer

The P2P device that should be connected to. Currently, this is the only way to create or join a group.

Format: string

wfd-ies

The Wi-Fi Display (WFD) Information Elements (IEs) to set.

Wi-Fi Display requires a protocol specific information element to be set in certain Wi-Fi frames. These can be specified here for the purpose of establishing a connection. This setting is only useful when implementing a Wi-Fi Display client.

Format: byte array

wps-method

Flags indicating which mode of WPS is to be used.

There's little point in changing the default setting as NetworkManager will automatically determine the best method to use.

Format: uint32

wimax setting

WiMax Settings.

Properties:

mac-address

Alias: mac

If specified, this connection will only apply to the WiMAX device whose MAC address matches. This property does not change the MAC address of the device (known as MAC spoofing).

This property is deprecated since version 1.2. WiMAX is no longer supported.

Format: byte array

network-name

Alias: nsp

Network Service Provider (NSP) name of the WiMAX network this connection should use.

This property is deprecated since version 1.2. WiMAX is no longer supported.

Format: string

802-3-ethernet setting

Alias: ethernet

Wired Ethernet Settings.

Properties:

accept-all-mac-addresses

When TRUE, setup the interface to accept packets for all MAC addresses. This is enabling the kernel interface flag IFF_PROMISC. When FALSE, the interface will only accept the packets with the interface destination mac address or broadcast.

Format: NMTernary (int32)

auto-negotiate

When TRUE, enforce auto-negotiation of speed and duplex mode. If "speed" and "duplex" properties are both specified, only that single mode will be advertised and accepted during the link auto-negotiation process: this works only for BASE-T 802.3 specifications and is useful for enforcing gigabits modes, as in these cases link negotiation is mandatory. When FALSE, "speed" and "duplex" properties should be both set or link configuration will be skipped.

Format: boolean

cloned-mac-address

Alias: cloned-mac

If specified, request that the device use this MAC address instead. This is known as MAC cloning or spoofing.

Beside explicitly specifying a MAC address, the special values "preserve", "permanent", "random" and "stable" are supported.

"preserve" means not to touch the MAC address on activation. "permanent" means to use the permanent hardware address if the device has one (otherwise this is treated as "preserve"). "random" creates a random MAC address on each connect. "stable" creates a hashed MAC address based on connection.stable-id and a machine dependent key.

If unspecified, the value can be overwritten via global defaults, see manual of NetworkManager.conf. If still unspecified, it defaults to "preserve" (older versions of NetworkManager may use a different default value).

On D-Bus, this field is expressed as "assigned-mac-address" or the deprecated "cloned-mac-address".

Format: byte array

duplex

When a value is set, either "half" or "full", configures the device to use the specified duplex mode. If "auto-negotiate" is "yes" the specified duplex mode will be the only one advertised during link negotiation: this works only for BASE-T 802.3 specifications and is useful for enforcing gigabits modes, as in these cases link negotiation is mandatory. If the value is unset (the default), the link configuration will be either skipped (if "auto-negotiate" is "no", the default) or will be auto-negotiated (if "auto-negotiate" is "yes") and the local device will advertise all the supported duplex modes. Must be set together with the "speed" property if specified. Before specifying a duplex mode be sure your device supports it.

Format: string

generate-mac-address-mask

With "cloned-mac-address" setting "random" or "stable", by default all bits of the MAC address are scrambled and a locally-administered, unicast MAC address is created. This property allows to specify that certain bits are fixed. Note that the least significant bit of the first MAC address will always be unset to create a unicast MAC address.

If the property is NULL, it is eligible to be overwritten by a default connection setting. If the value is still NULL or an empty string, the default is to create a locally-administered, unicast MAC address.

If the value contains one MAC address, this address is used as mask. The set bits of the mask are to be filled with the current MAC address of the device, while the unset bits are subject to randomization. Setting "FE:FF:FF:00:00:00" means to preserve the OUI of the current MAC address and only randomize the lower 3 bytes using the "random" or "stable" algorithm.

If the value contains one additional MAC address after the mask, this address is used instead of the current MAC address to fill the bits that shall not be randomized. For example, a value of "FE:FF:FF:00:00:00 68:F7:28:00:00:00" will set the OUI of the MAC address to 68:F7:28, while the lower bits are randomized. A value of "02:00:00:00:00:00 00:00:00:00:00:00" will create a fully scrambled globally-administered, burned-in MAC address.

If the value contains more than one additional MAC addresses, one of them is chosen randomly. For example, "02:00:00:00:00:00 00:00:00:00:00:00 02:00:00:00:00:00" will create a fully scrambled MAC address, randomly locally or globally administered.

Format: string

mac-address

Alias: mac

If specified, this connection will only apply to the Ethernet device whose permanent MAC address matches. This property does not

change the MAC address of the device (i.e. MAC spoofing).

Format: byte array

mac-address-blacklist

If specified, this connection will never apply to the Ethernet device whose permanent MAC address matches an address in the list. Each MAC address is in the standard hex-digits-and-colons notation (00:11:22:33:44:55).

Format: array of string

mtu

Alias: mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple Ethernet frames.

Format: uint32

port

Specific port type to use if the device supports multiple attachment methods. One of "tp" (Twisted Pair), "aui" (Attachment Unit Interface), "bnc" (Thin Ethernet) or "mii" (Media Independent Interface). If the device supports only one port type, this setting is ignored.

Format: string

s390-nettype

s390 network device type; one of "qeth", "lcs", or "ctc", representing the different types of virtual network devices available on s390 systems.

Format: string

s390-options

Dictionary of key/value pairs of s390-specific device options. Both keys and values must be strings. Allowed keys include "portno", "layer2", "portname", "protocol", among others. Key names must contain only alphanumeric characters (ie, [a-zA-Z0-9]). Currently, NetworkManager itself does nothing with this information. However, s390utils ships a udev rule which parses this information and applies it to the interface.

Format: dict of string to string

s390-subchannels

Identifies specific subchannels that this network device uses for communication with z/VM or s390 host. Like the "mac-address" property for non-z/VM devices, this property can be used to ensure this connection only applies to the network device that uses these subchannels. The list should contain exactly 3 strings, and each string may only be composed of hexadecimal characters and the period (.) character.

Format: array of string

speed

When a value greater than 0 is set, configures the device to use the specified speed. If "auto-negotiate" is "yes" the specified speed will be the only one advertised during link negotiation: this works only for BASE-T 802.3 specifications and is useful for enforcing gigabit speeds, as in this case link negotiation is mandatory. If the value is unset (0, the default), the link configuration will be either skipped (if "auto-negotiate" is "no", the default) or will be auto-negotiated (if "auto-negotiate" is "yes") and the local device will advertise all the supported speeds. In Mbit/s, ie 100 == 100Mbit/s. Must be set together with the "duplex" property when non-zero. Before specifying a speed value be sure your device supports it.

Format: uint32

wake-on-lan

The NMSettingWiredWakeOnLan options to enable. Not all devices support all options. May be any combination of NM_SETTING_WIRED_WAKE_ON_LAN_PHY (0x2), NM_SETTING_WIRED_WAKE_ON_LAN_UNICAST (0x4), NM_SETTING_WIRED_WAKE_ON_LAN_MULTICAST (0x8), NM_SETTING_WIRED_WAKE_ON_LAN_BROADCAST (0x10), NM_SETTING_WIRED_WAKE_ON_LAN_ARP (0x20), NM_SETTING_WIRED_WAKE_ON_LAN_MAGIC (0x40) or the special values

NM_SETTING_WIRED_WAKE_ON_LAN_DEFAULT (0x1) (to use global settings) and NM_SETTING_WIRED_WAKE_ON_LAN_IGNORE (0x8000) (to disable management of Wake-on-LAN in NetworkManager).

Format: uint32

wake-on-lan-password

If specified, the password used with magic-packet-based Wake-on-LAN, represented as an Ethernet MAC address. If NULL, no password will be required.

Format: string

wireguard setting

WireGuard Settings.

Properties:

fwmark

The use of fwmark is optional and is by default off. Setting it to 0 disables it. Otherwise, it is a 32-bit fwmark for outgoing packets.

Note that "ip4-auto-default-route" or "ip6-auto-default-route" enabled, implies to automatically choose a fwmark.

Format: uint32

ip4-auto-default-route

Whether to enable special handling of the IPv4 default route. If enabled, the IPv4 default route from wireguard.peer-routes will be placed to a dedicated routing-table and two policy routing rules will be added. The fwmark number is also used as routing-table for the default-route, and if fwmark is zero, an unused fwmark/table is chosen automatically. This corresponds to what wg-quick does with Table=auto and what WireGuard calls "Improved Rule-based Routing". Note that for this automatism to work, you usually don't want to set ipv4.gateway, because that will result in a conflicting default route.

Leaving this at the default will enable this option automatically if ipv4.never-default is not set and there are any peers that use a default-route as allowed-ips. Since this automatism only makes

sense if you also have a peer with an /0 allowed-ips, it is usually not necessary to enable this explicitly. However, you can disable it if you want to configure your own routing and rules.

Format: NMTernary (int32)

ip6-auto-default-route

Like ip4-auto-default-route, but for the IPv6 default route.

Format: NMTernary (int32)

listen-port

The listen-port. If listen-port is not specified, the port will be chosen randomly when the interface comes up.

Format: uint32

mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple fragments.

If zero a default MTU is used. Note that contrary to wg-quick's MTU setting, this does not take into account the current routes at the time of activation.

Format: uint32

peer-routes

Whether to automatically add routes for the AllowedIPs ranges of the peers. If TRUE (the default), NetworkManager will automatically add routes in the routing tables according to ipv4.route-table and ipv6.route-table. Usually you want this automatism enabled. If FALSE, no such routes are added automatically. In this case, the user may want to configure static routes in ipv4.routes and ipv6.routes, respectively.

Note that if the peer's AllowedIPs is "0.0.0.0/0" or "::/0" and the profile's ipv4.never-default or ipv6.never-default setting is enabled, the peer route for this peer won't be added automatically.

Format: boolean

private-key

The 256 bit private-key in base64 encoding.

Format: string

private-key-flags

Flags indicating how to handle the "private-key" property.

Format: NMSettingSecretFlags (uint32)

802-11-wireless setting

Alias: wifi

Wi-Fi Settings.

Properties:

ap-isolation

Configures AP isolation, which prevents communication between wireless devices connected to this AP. This property can be set to a value different from NM_TERNARY_DEFAULT (-1) only when the interface is configured in AP mode.

If set to NM_TERNARY_TRUE (1), devices are not able to communicate with each other. This increases security because it protects devices against attacks from other clients in the network. At the same time, it prevents devices to access resources on the same wireless networks as file shares, printers, etc.

If set to NM_TERNARY_FALSE (0), devices can talk to each other.

When set to NM_TERNARY_DEFAULT (-1), the global default is used; in case the global default is unspecified it is assumed to be NM_TERNARY_FALSE (0).

Format: NMTernary (int32)

band

802.11 frequency band of the network. One of "a" for 5GHz 802.11a or "bg" for 2.4GHz 802.11. This will lock associations to the Wi-Fi network to the specific band, i.e. if "a" is specified, the device will not associate with the same network in the 2.4GHz band even if the network's settings are compatible. This setting depends on specific driver capability and may not work with all drivers.

Format: string

bssid

If specified, directs the device to only associate with the given access point. This capability is highly driver dependent and not

supported by all devices. Note: this property does not control the BSSID used when creating an Ad-Hoc network and is unlikely to in the future.

Locking a client profile to a certain BSSID will prevent roaming and also disable background scanning. That can be useful, if there is only one access point for the SSID.

Format: byte array

channel

Wireless channel to use for the Wi-Fi connection. The device will only join (or create for Ad-Hoc networks) a Wi-Fi network on the specified channel. Because channel numbers overlap between bands, this property also requires the "band" property to be set.

Format: uint32

cloned-mac-address

Alias: cloned-mac

If specified, request that the device use this MAC address instead. This is known as MAC cloning or spoofing.

Beside explicitly specifying a MAC address, the special values "preserve", "permanent", "random" and "stable" are supported.

"preserve" means not to touch the MAC address on activation.

"permanent" means to use the permanent hardware address of the device. "random" creates a random MAC address on each connect.

"stable" creates a hashed MAC address based on connection.stable-id and a machine dependent key.

If unspecified, the value can be overwritten via global defaults, see manual of NetworkManager.conf. If still unspecified, it defaults to "preserve" (older versions of NetworkManager may use a different default value).

On D-Bus, this field is expressed as "assigned-mac-address" or the deprecated "cloned-mac-address".

Format: byte array

generate-mac-address-mask

With "cloned-mac-address" setting "random" or "stable", by default

all bits of the MAC address are scrambled and a locally-administered, unicast MAC address is created. This property allows to specify that certain bits are fixed. Note that the least significant bit of the first MAC address will always be unset to create a unicast MAC address.

If the property is NULL, it is eligible to be overwritten by a default connection setting. If the value is still NULL or an empty string, the default is to create a locally-administered, unicast MAC address.

If the value contains one MAC address, this address is used as mask. The set bits of the mask are to be filled with the current MAC address of the device, while the unset bits are subject to randomization. Setting "FE:FF:FF:00:00:00" means to preserve the OUI of the current MAC address and only randomize the lower 3 bytes using the "random" or "stable" algorithm.

If the value contains one additional MAC address after the mask, this address is used instead of the current MAC address to fill the bits that shall not be randomized. For example, a value of "FE:FF:FF:00:00:00 68:F7:28:00:00:00" will set the OUI of the MAC address to 68:F7:28, while the lower bits are randomized. A value of "02:00:00:00:00:00 00:00:00:00:00:00" will create a fully scrambled globally-administered, burned-in MAC address.

If the value contains more than one additional MAC addresses, one of them is chosen randomly. For example, "02:00:00:00:00:00 00:00:00:00:00:00 02:00:00:00:00:00" will create a fully scrambled MAC address, randomly locally or globally administered.

Format: string

hidden

If TRUE, indicates that the network is a non-broadcasting network that hides its SSID. This works both in infrastructure and AP mode.

In infrastructure mode, various workarounds are used for a more reliable discovery of hidden networks, such as probe-scanning the SSID. However, these workarounds expose inherent insecurities with

hidden SSID networks, and thus hidden SSID networks should be used with caution.

In AP mode, the created network does not broadcast its SSID. Note that marking the network as hidden may be a privacy issue for you (in infrastructure mode) or client stations (in AP mode), as the explicit probe-scans are distinctly recognizable on the air.

Format: boolean

mac-address

Alias: mac

If specified, this connection will only apply to the Wi-Fi device whose permanent MAC address matches. This property does not change the MAC address of the device (i.e. MAC spoofing).

Format: byte array

mac-address-blacklist

A list of permanent MAC addresses of Wi-Fi devices to which this connection should never apply. Each MAC address should be given in the standard hex-digits-and-colons notation (eg "00:11:22:33:44:55").

Format: array of string

mac-address-randomization

One of NM_SETTING_MAC_RANDOMIZATION_DEFAULT (0) (never randomize unless the user has set a global default to randomize and the supplicant supports randomization), NM_SETTING_MAC_RANDOMIZATION_NEVER (1) (never randomize the MAC address), or NM_SETTING_MAC_RANDOMIZATION_ALWAYS (2) (always randomize the MAC address).

This property is deprecated since version 1.4. Use the "cloned-mac-address" property instead.

Format: uint32

mode

Alias: mode

Wi-Fi network mode; one of "infrastructure", "mesh", "adhoc" or "ap". If blank, infrastructure is assumed.

Format: string

mtu

Alias: mtu

If non-zero, only transmit packets of the specified size or

smaller, breaking larger packets up into multiple Ethernet frames.

Format: uint32

powersave

One of NM_SETTING_WIRELESS_POWERSAVE_DISABLE (2) (disable Wi-Fi

power saving), NM_SETTING_WIRELESS_POWERSAVE_ENABLE (3) (enable

Wi-Fi power saving), NM_SETTING_WIRELESS_POWERSAVE_IGNORE (1)

(don't touch currently configure setting) or

NM_SETTING_WIRELESS_POWERSAVE_DEFAULT (0) (use the globally

configured value). All other values are reserved.

Format: uint32

rate

If non-zero, directs the device to only use the specified bitrate

for communication with the access point. Units are in Kb/s, ie 5500

= 5.5 Mbit/s. This property is highly driver dependent and not all

devices support setting a static bitrate.

Format: uint32

seen-bssids

A list of BSSIDs (each BSSID formatted as a MAC address like

"00:11:22:33:44:55") that have been detected as part of the Wi-Fi

network. NetworkManager internally tracks previously seen BSSIDs.

The property is only meant for reading and reflects the BSSID list

of NetworkManager. The changes you make to this property will not

be preserved.

Format: array of string

ssid

Alias: ssid

SSID of the Wi-Fi network. Must be specified.

Format: byte array

tx-power

If non-zero, directs the device to use the specified transmit
power. Units are dBm. This property is highly driver dependent and
not all devices support setting a static transmit power.
Format: uint32

wake-on-wlan
The NMSettingWirelessWakeOnWLan options to enable. Not all devices
support all options. May be any combination of
NM_SETTING_WIRELESS_WAKE_ON_WLAN_ANY (0x2),
NM_SETTING_WIRELESS_WAKE_ON_WLAN_DISCONNECT (0x4),
NM_SETTING_WIRELESS_WAKE_ON_WLAN_MAGIC (0x8),
NM_SETTING_WIRELESS_WAKE_ON_WLAN_GTK_REKEY_FAILURE (0x10),
NM_SETTING_WIRELESS_WAKE_ON_WLAN_EAP_IDENTITY_REQUEST (0x20),
NM_SETTING_WIRELESS_WAKE_ON_WLAN_4WAY_HANDSHAKE (0x40),
NM_SETTING_WIRELESS_WAKE_ON_WLAN_RFKILL_RELEASE (0x80),
NM_SETTING_WIRELESS_WAKE_ON_WLAN_TCP (0x100) or the special values
NM_SETTING_WIRELESS_WAKE_ON_WLAN_DEFAULT (0x1) (to use global
settings) and NM_SETTING_WIRELESS_WAKE_ON_WLAN_IGNORE (0x8000) (to
disable management of Wake-on-LAN in NetworkManager).
Format: uint32

802-11-wireless-security setting
Alias: wifi-sec
Wi-Fi Security Settings.
Properties:
auth-alg
When WEP is used (ie, key-mgmt = "none" or "ieee8021x") indicate
the 802.11 authentication algorithm required by the AP here. One of
"open" for Open System, "shared" for Shared Key, or "leap" for
Cisco LEAP. When using Cisco LEAP (ie, key-mgmt = "ieee8021x" and
auth-alg = "leap") the "leap-username" and "leap-password"
properties must be specified.
Format: string

fils
Indicates whether Fast Initial Link Setup (802.11ai) must be

enabled for the connection. One of NM_SETTING_WIRELESS_SECURITY_FILS_DEFAULT (0) (use global default value), NM_SETTING_WIRELESS_SECURITY_FILS_DISABLE (1) (disable FILS), NM_SETTING_WIRELESS_SECURITY_FILS_OPTIONAL (2) (enable FILS if the supplicant and the access point support it) or NM_SETTING_WIRELESS_SECURITY_FILS_REQUIRED (3) (enable FILS and fail if not supported). When set to NM_SETTING_WIRELESS_SECURITY_FILS_DEFAULT (0) and no global default is set, FILS will be optionally enabled.

Format: int32

group

A list of group/broadcast encryption algorithms which prevents connections to Wi-Fi networks that do not utilize one of the algorithms in the list. For maximum compatibility leave this property empty. Each list element may be one of "wep40", "wep104", "tkip", or "ccmp".

Format: array of string

key-mgmt

Key management used for the connection. One of "none" (WEP or no password protection), "ieee8021x" (Dynamic WEP), "owe" (Opportunistic Wireless Encryption), "wpa-psk" (WPA2 + WPA3 personal), "sae" (WPA3 personal only), "wpa-eap" (WPA2 + WPA3 enterprise) or "wpa-eap-suite-b-192" (WPA3 enterprise only). This property must be set for any Wi-Fi connection that uses security.

Format: string

leap-password

The login password for legacy LEAP connections (ie, key-mgmt = "ieee8021x" and auth-alg = "leap").

Format: string

leap-password-flags

Flags indicating how to handle the "leap-password" property.

Format: NMSettingSecretFlags (uint32)

leap-username

The login username for legacy LEAP connections (ie, key-mgmt = "ieee8021x" and auth-alg = "leap").

Format: string

pairwise

A list of pairwise encryption algorithms which prevents connections to Wi-Fi networks that do not utilize one of the algorithms in the list. For maximum compatibility leave this property empty. Each list element may be one of "tkip" or "ccmp".

Format: array of string

pmf

Indicates whether Protected Management Frames (802.11w) must be enabled for the connection. One of NM_SETTING_WIRELESS_SECURITY_PMF_DEFAULT (0) (use global default value), NM_SETTING_WIRELESS_SECURITY_PMF_DISABLE (1) (disable PMF), NM_SETTING_WIRELESS_SECURITY_PMF_OPTIONAL (2) (enable PMF if the supplicant and the access point support it) or NM_SETTING_WIRELESS_SECURITY_PMF_REQUIRED (3) (enable PMF and fail if not supported). When set to NM_SETTING_WIRELESS_SECURITY_PMF_DEFAULT (0) and no global default is set, PMF will be optionally enabled.

Format: int32

proto

List of strings specifying the allowed WPA protocol versions to use. Each element may be one "wpa" (allow WPA) or "rsn" (allow WPA2/RSN). If not specified, both WPA and RSN connections are allowed.

Format: array of string

psk

Pre-Shared-Key for WPA networks. For WPA-PSK, it's either an ASCII passphrase of 8 to 63 characters that is (as specified in the 802.11i standard) hashed to derive the actual key, or the key in form of 64 hexadecimal character. The WPA3-Personal networks use a

passphrase of any length for SAE authentication.

Format: string

psk-flags

Flags indicating how to handle the "psk" property.

Format: NMSettingSecretFlags (uint32)

wep-key-flags

Flags indicating how to handle the "wep-key0", "wep-key1", "wep-key2", and "wep-key3" properties.

Format: NMSettingSecretFlags (uint32)

wep-key-type

Controls the interpretation of WEP keys. Allowed values are NM_WEP_KEY_TYPE_KEY (1), in which case the key is either a 10- or 26-character hexadecimal string, or a 5- or 13-character ASCII password; or NM_WEP_KEY_TYPE_PASSPHRASE (2), in which case the passphrase is provided as a string and will be hashed using the de-facto MD5 method to derive the actual WEP key.

Format: NMWepKeyType (uint32)

wep-key0

Index 0 WEP key. This is the WEP key used in most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-key1

Index 1 WEP key. This WEP index is not used by most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-key2

Index 2 WEP key. This WEP index is not used by most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-key3

Index 3 WEP key. This WEP index is not used by most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-tx-keyidx

When static WEP is used (ie, key-mgmt = "none") and a non-default WEP key index is used by the AP, put that WEP key index here. Valid values are 0 (default key) through 3. Note that some consumer access points (like the Linksys WRT54G) number the keys 1 - 4.

Format: uint32

wps-method

Flags indicating which mode of WPS is to be used if any.

There's little point in changing the default setting as NetworkManager will automatically determine whether it's feasible to start WPS enrollment from the Access Point capabilities.

WPS can be disabled by setting this property to a value of 1.

Format: uint32

wpan setting

IEEE 802.15.4 (WPAN) MAC Settings.

Properties:

channel

Alias: channel

IEEE 802.15.4 channel. A positive integer or -1, meaning "do not set, use whatever the device is already set to".

Format: int32

mac-address

Alias: mac

If specified, this connection will only apply to the IEEE 802.15.4 (WPAN) MAC layer device whose permanent MAC address matches.

Format: string

page

Alias: page

IEEE 802.15.4 channel page. A positive integer or -1, meaning "do

not set, use whatever the device is already set to".

Format: int32

pan-id

Alias: pan-id

IEEE 802.15.4 Personal Area Network (PAN) identifier.

Format: uint32

short-address

Alias: short-addr

Short IEEE 802.15.4 address to be used within a restricted environment.

Format: uint32

bond-port setting

Bond Port Settings.

Properties:

queue-id

Alias: queue-id

The queue ID of this bond port. The maximum value of queue ID is the number of TX queues currently active in device.

Format: uint32

hostname setting

Hostname settings.

Properties:

from-dhcp

Whether the system hostname can be determined from DHCP on this connection.

When set to NM_TERNARY_DEFAULT (-1), the value from global configuration is used. If the property doesn't have a value in the global configuration, NetworkManager assumes the value to be NM_TERNARY_TRUE (1).

Format: NMTernary (int32)

from-dns-lookup

Whether the system hostname can be determined from reverse DNS lookup of addresses on this device.

When set to NM_TERNARY_DEFAULT (-1), the value from global configuration is used. If the property doesn't have a value in the global configuration, NetworkManager assumes the value to be NM_TERNARY_TRUE (1).

Format: NMTernary (int32)

only-from-default

If set to NM_TERNARY_TRUE (1), NetworkManager attempts to get the hostname via DHCPv4/DHCPv6 or reverse DNS lookup on this device only when the device has the default route for the given address family (IPv4/IPv6).

If set to NM_TERNARY_FALSE (0), the hostname can be set from this device even if it doesn't have the default route.

When set to NM_TERNARY_DEFAULT (-1), the value from global configuration is used. If the property doesn't have a value in the global configuration, NetworkManager assumes the value to be NM_TERNARY_FALSE (0).

Format: NMTernary (int32)

priority

The relative priority of this connection to determine the system hostname. A lower numerical value is better (higher priority). A connection with higher priority is considered before connections with lower priority.

If the value is zero, it can be overridden by a global value from NetworkManager configuration. If the property doesn't have a value in the global configuration, the value is assumed to be 100.

Negative values have the special effect of excluding other connections with a greater numerical priority value; so in presence of at least one negative priority, only connections with the lowest priority value will be used to determine the hostname.

Format: int32

loopback setting

Loopback Link Settings.

Properties:

mtu

Alias: mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple Ethernet frames.

Format: uint32

veth setting

Veth Settings.

Properties:

peer

Alias: peer

This property specifies the peer interface name of the veth. This property is mandatory.

Format: string

Secret flag types:

Each password or secret property in a setting has an associated flags property that describes how to handle that secret. The flags property is a bitfield that contains zero or more of the following values logically OR-ed together.

? 0x0 (none) - the system is responsible for providing and storing this secret. This may be required so that secrets are already available before the user logs in. It also commonly means that the secret will be stored in plain text on disk, accessible to root only. For example via the keyfile settings plugin as described in the "PLUGINS" section in NetworkManager.conf(5).

? 0x1 (agent-owned) - a user-session secret agent is responsible for providing and storing this secret; when it is required, agents will be asked to provide it.

? 0x2 (not-saved) - this secret should not be saved but should be requested from the user each time it is required. This flag should be used for One-Time-Pad secrets, PIN codes from hardware tokens, or if the user simply does not want to save the secret.

? 0x4 (not-required) - in some situations it cannot be automatically determined that a secret is required or not. This flag hints that

the secret is not required and should not be requested from the user.

FILES

/etc/NetworkManager/system-connections or distro plugin-specific location

SEE ALSO

nmcli(1), nmcli-examples(7), NetworkManager(8), nm-settings-dbus(5), nm-settings-keyfile(5), NetworkManager.conf(5)