Package: MAC.address (via r-universe)

October 26, 2026

Title Find, Validate and Identify Device MAC Addresses

Version 0.1.0.9000

Description Tools to work with MAC (Media Access Control) addresses: list the MAC addresses of the local network interfaces and of the neighbors in the ARP cache on Linux, macOS and Windows, extract MAC addresses from free text, validate and normalize their notation, check the multicast and locally administered bits, generate random addresses, derive IPv6 interface identifiers (modified EUI-64), and identify the manufacturer using the public IEEE (Institute of Electrical and Electronics Engineers) MA-L, MA-M and MA-S registries <https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries>.

URL https://github.com/EnriquePH/mac-address, https://enriqueph.github.io/mac-address/

BugReports https://github.com/EnriquePH/mac-address/issues

Depends R (>= 4.0.0)

Imports stats, tools, utils

Suggests curl, testthat (>= 3.0.0), withr

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Repository https://enriqueph.r-universe.dev

Date/Publication 2026-10-26 18:44:46 UTC

RemoteUrl https://github.com/enriqueph/mac-address

RemoteRef HEAD

RemoteSha 6c7969982e22eb22d349b7d3b00878d62beb5ff0

Contents

eui64_MAC 2
get_arp_table 3
get_MAC 4
get_oui_registry 4
is_local_MAC 5
is_MAC 6
is_multicast_MAC 7
lookup_vendor 7
normalize_MAC 8
random_MAC 9
search_MAC 9

Index **11**

`eui64_MAC` *Modified EUI-64 interface identifier of MAC addresses*

Description

Derives the IPv6 interface identifier from a MAC address, as used by stateless address autoconfiguration (RFC 4291, appendix A): `FF:FE` is inserted in the middle of the address and the U/L bit is inverted.

Usage

```
eui64_MAC(mac, link_local = FALSE)
```

Arguments

`mac`	A character vector of MAC addresses.
`link_local`	Logical. If `TRUE`, return the IPv6 link-local address (`fe80::/64` prefix) instead of the interface identifier alone.

Value

A character vector in IPv6 notation (lower case, leading zeros omitted, as recommended by RFC 5952). `NA` for invalid addresses.

Examples

```
eui64_MAC("1C:4C:7E:30:D8:13")
eui64_MAC("1C:4C:7E:30:D8:13", link_local = TRUE)
```

get_arp_table — *Get the ARP table of this device*

Description

Lists the IPv4 neighbors (other devices in the local network) that this machine has recently talked to, with their MAC addresses, as stored in the operating system ARP cache. Combined with `lookup_vendor()`, it helps to identify the devices in a network.

Usage

```
get_arp_table()
```

Details

Works on Linux (reads `/proc/net/arp`), macOS and other Unix systems (parses `arp -an`) and Windows (parses `arp -a`). Only the existing cache is read: no packets are sent, so devices that have not communicated with this machine recently are not listed.

Value

A data frame with columns `ip`, `mac` (normalized, see `normalize_MAC()`) and `interface` (on Windows, the IP address of the interface). Incomplete entries are dropped. If the table cannot be read, a warning is given and a data frame with no rows is returned.

See Also

`get_MAC()` for the addresses of this device.

Examples

```
arp <- get_arp_table()
head(arp)


# Manufacturer of each neighbor (needs an internet connection once)
oui <- try(get_oui_registry(cache_dir = tempdir()))
if (is.data.frame(oui)) arp$vendor <- lookup_vendor(arp$mac, oui)
```

get_MAC — *Get the MAC addresses of this device*

Description

Lists the hardware (MAC) address of each network interface of the local machine. Works on Linux (reads `/sys/class/net`), macOS and other Unix systems (parses `ifconfig -a`) and Windows (parses `getmac`).

Usage

```
get_MAC(include_null = FALSE)
```

Arguments

`include_null`	Logical. Keep interfaces whose address is `00:00:00:00:00:00` (e.g. the loopback interface)?

Value

A named character vector of normalized MAC addresses (`XX:XX:XX:XX:XX:XX`), named by interface. If the addresses cannot be read, a warning is given and an empty vector is returned.

Examples

```
get_MAC()
```

get_oui_registry — *Download the IEEE OUI registry*

Description

Downloads the public IEEE registries of MAC address blocks and combines them in one data frame. The result is cached on disk, so later calls do not need an internet connection.

Usage

```
get_oui_registry(
  registry = c("MA-L", "MA-M", "MA-S"),
  refresh = FALSE,
  cache_dir = NULL
)
```

Arguments

`registry`	Which registries to download.
`refresh`	Logical. Download again even if a cached copy exists?
`cache_dir`	Directory where the CSV files are cached. By default, the user cache directory given by `tools::R_user_dir("MAC.address", "cache")`. To remove the cached files, delete the cache directory, e.g. `unlink(tools::R_user_dir("MAC.address' "cache"), recursive = TRUE)`.

Details

Registries:

- `MA-L`: MAC Address Block Large (24-bit prefix, the classic OUI).
- `MA-M`: MAC Address Block Medium (28-bit prefix).
- `MA-S`: MAC Address Block Small (36-bit prefix).

Value

A data frame with columns `registry`, `assignment` (hexadecimal prefix), `organization` and `address`. If a registry cannot be downloaded (e.g. no internet connection), an error with an informative message is given.

Examples

```
# Needs an internet connection; the MA-L registry is about 4 MB
oui <- try(get_oui_registry("MA-L", cache_dir = tempdir()))
if (is.data.frame(oui)) head(oui)
```

`is_local_MAC` *Check whether MAC addresses are locally administered*

Description

Locally administered addresses (e.g. randomized addresses used by phones for Wi-Fi privacy, or virtual machines) have the second least significant bit of the first octet set (the U/L bit).

Usage

```
is_local_MAC(mac)
```

Arguments

`mac`	A character vector of MAC addresses.

Value

A logical vector (NA for invalid addresses).

See Also

is_multicast_MAC(), random_MAC()

Examples

```
is_local_MAC(c("02:00:00:00:00:01", "1C:4C:7E:30:D8:13"))
```

---

is_MAC — *Check whether strings are valid MAC addresses*

Description

Check whether strings are valid MAC addresses

Usage

```
is_MAC(x)
```

Arguments

x	A character vector.

Value

A logical vector, TRUE where the whole string is a MAC address in colon, hyphen or Cisco dotted notation.

Examples

```
is_MAC(c("1C:4C:7E:30:D8:13", "1C:4C:7E:30:D8", "1C:4C-7E:30:D8:13"))
```

is_multicast_MAC — *Check whether MAC addresses are multicast*

Description

Multicast (group) addresses have the least significant bit of the first octet set (the I/G bit). The broadcast address `FF:FF:FF:FF:FF:FF` is a multicast address. Network interfaces always have unicast addresses.

Usage

```
is_multicast_MAC(mac)
```

Arguments

`mac`	A character vector of MAC addresses.

Value

A logical vector (`NA` for invalid addresses).

See Also

`is_local_MAC()`

Examples

```
is_multicast_MAC(c("01:00:5E:00:00:FB", "FF:FF:FF:FF:FF:FF",
                   "1C:4C:7E:30:D8:13"))
```

lookup_vendor — *Find the manufacturer of a MAC address*

Description

Looks up the organization that owns each MAC address block, using the longest matching prefix (MA-S, then MA-M, then MA-L).

Usage

```
lookup_vendor(mac, registry = get_oui_registry())
```

Arguments

`mac`	A character vector of MAC addresses in any supported notation.
`registry`	A data frame as returned by `get_oui_registry()`, with at least the columns `assignment` and `organization`.

Details

Locally administered addresses (see `is_local_MAC()`), such as randomized Wi-Fi addresses, are not part of the IEEE registries and usually return `NA`.

Value

A character vector with the organization names (`NA` if unknown).

Examples

```
# A small registry, to run without internet access
registry <- data.frame(
  registry = c("MA-L", "MA-M"),
  assignment = c("00000C", "70B3D51"),
  organization = c("Cisco Systems, Inc", "Example Medium Co")
)
macs <- c("00-00-0C-12-34-56", "70:B3:D5:1A:00:00", "02:00:00:00:00:01")
lookup_vendor(macs, registry)


# With the full IEEE registries (needs an internet connection)
oui <- try(get_oui_registry(cache_dir = tempdir()))
if (is.data.frame(oui)) lookup_vendor(get_MAC(), oui)
```

`normalize_MAC` *Normalize MAC addresses*

Description

Converts MAC addresses written in any supported notation to a common upper-case format.

Usage

```
normalize_MAC(mac, sep = ":")
```

Arguments

`mac`	A character vector of MAC addresses.
`sep`	Separator to use in the output. Use `""` for plain hexadecimal.

Value

A character vector. Invalid addresses become `NA`.

Examples

```
normalize_MAC(c("1c-4c-7e-30-d8-13", "1c4c.7e30.d813", "not a MAC"))
normalize_MAC("1C:4C:7E:30:D8:13", sep = "-")
```

random_MAC — *Generate random MAC addresses*

Description

Generates random MAC addresses, e.g. for virtual machines, containers or tests. By default they are locally administered unicast addresses, so they never clash with addresses assigned by the IEEE to manufacturers. Uses R's random number generator, so results can be reproduced with `set.seed()`.

Usage

```
random_MAC(n = 1, local = TRUE, multicast = FALSE, sep = ":")
```

Arguments

`n`	Number of addresses.
`local`	Logical. Set the locally administered bit (see `is_local_MAC()`)? If `FALSE`, the bit is cleared.
`multicast`	Logical. Set the multicast bit (see `is_multicast_MAC()`)?
`sep`	Separator to use, as in `normalize_MAC()`.

Value

A character vector of `n` MAC addresses.

Examples

```
set.seed(1)
random_MAC(3)
random_MAC(1, sep = "-")
```

search_MAC — *Search MAC addresses in a text*

Description

Extracts every MAC-48 address found in `txt`. Supports the usual notations: colon (`1C:4C:7E:30:D8:13`), hyphen (`1C-4C-7E-30-D8-13`) and Cisco dotted (`1C4C.7E30.D813`). The separator must be the same inside each address, and longer hexadecimal sequences (e.g. EUI-64 or IPv6) are not partially matched.

Usage

```
search_MAC(
  txt,
  sep = c(":", "-"),
  dotted = TRUE,
  normalize = FALSE,
  unique = FALSE
)
```

Arguments

`txt`	A character vector with the text to search.
`sep`	A character vector with the allowed single-character separators. Use `character()` to search only the dotted notation.
`dotted`	Logical. Also search Cisco dotted notation (`xxxx.xxxx.xxxx`)?
`normalize`	Logical. If `TRUE`, results are returned in canonical `XX:XX:XX:XX:XX:XX` form (see `normalize_MAC()`); otherwise they are returned as found, in upper case.
`unique`	Logical. Remove duplicated addresses?

Value

A character vector with the MAC addresses found (possibly empty).

Examples

```
sample_txt <- "MAC addresses 1C:4C:7E:30:d8:13 and 00-1A-2B-3C-4D-5E"
search_MAC(sample_txt)
search_MAC(sample_txt, normalize = TRUE)
search_MAC("Cisco style: 001a.2b3c.4d5e")
```

Index

eui64_MAC, 2

get_arp_table, 3
get_MAC, 4
get_MAC(), *3*
get_oui_registry, 4
get_oui_registry(), *7*

is_local_MAC, 5
is_local_MAC(), *7–9*
is_MAC, 6
is_multicast_MAC, 7
is_multicast_MAC(), *6*, *9*

lookup_vendor, 7
lookup_vendor(), *3*

normalize_MAC, 8
normalize_MAC(), *3*, *9*, *10*

random_MAC, 9
random_MAC(), *6*

search_MAC, 9
set.seed(), *9*