

Package ‘adaR’

September 22, 2026

Title A Fast 'WHATWG' Compliant URL Parser

Version 0.4.0

Description A wrapper for 'ada-url', a 'WHATWG' compliant and fast URL parser written in modern 'C++'. Also contains auxiliary functions such as a public suffix extractor.

URL <https://gesistsa.github.io/adaR/>, <https://github.com/gesistsa/adaR>

BugReports <https://github.com/gesistsa/adaR/issues>

License MIT + file LICENSE

Encoding UTF-8

LinkingTo Rcpp

Imports Rcpp, triebeard

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.2)

VignetteBuilder knitr

SystemRequirements C++20, GNU make, GCC >= 12 (or Clang with equivalent C++20 support)

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author David Schoch [aut, cre] (ORCID:

<<https://orcid.org/0000-0003-2952-4812>>),

Chung-hong Chan [aut] (ORCID: <<https://orcid.org/0000-0002-6232-7530>>),

Yagiz Nizipli [ctb, cph] (author of ada-url :

<<https://github.com/ada-url/ada>>),

Daniel Lemire [ctb, cph] (author of ada-url :

<<https://github.com/ada-url/ada>>)

Maintainer David Schoch <david@schochastics.net>

Repository CRAN

Date/Publication 2026-09-22 20:40:09 UTC

Contents

ada_clear_port	2
ada_get_href	3
ada_has_credentials	4
ada_set_href	5
ada_url_parse	6
public_suffix	7
url_decode2	8
Index	9

ada_clear_port	<i>Clear a specific component of URL</i>
----------------	--

Description

These functions clears a specific component of URL.

Usage

```
ada_clear_port(url, decode = TRUE)

ada_clear_hash(url, decode = TRUE)

ada_clear_search(url, decode = TRUE)
```

Arguments

url	character. one or more URL to be parsed
decode	logical. Whether to decode the output (see utils::URLdecode()), default to TRUE

Value

character, NA if not a valid URL

Examples

```
url <- "https://user_1:password_1@example.org:8080/dir/../../api?q=1#frag"
ada_clear_port(url)
ada_clear_hash(url)
ada_clear_search(url)
```

ada_get_href	<i>Get a specific component of URL</i>
--------------	--

Description

These functions get a specific component of URL.

Usage

```
ada_get_href(url, decode = TRUE)

ada_get_username(url, decode = TRUE)

ada_get_password(url, decode = TRUE)

ada_get_port(url, decode = TRUE)

ada_get_hash(url, decode = TRUE)

ada_get_host(url, decode = TRUE)

ada_get_hostname(url, decode = TRUE)

ada_get_pathname(url, decode = TRUE)

ada_get_search(url, decode = TRUE)

ada_get_protocol(url, decode = TRUE)

ada_get_domain(url, decode = TRUE, icann_only = FALSE)

ada_get_basename(url, decode = TRUE)
```

Arguments

url	character. one or more URL to be parsed
decode	logical. Whether to decode the output (see utils::URLdecode()), default to TRUE
icann_only	logical. Whether to use only the ICANN section of the public suffix list, ignoring privately registered suffixes such as <code>github.io</code> or <code>blogspot.com</code> . Defaults to FALSE.

Value

character, NA if not a valid URL

Examples

```
url <- "https://user_1:password_1@example.org:8080/dir/../../api?q=1#frag"
ada_get_href(url)
ada_get_username(url)
ada_get_password(url)
ada_get_port(url)
ada_get_hash(url)
ada_get_host(url)
ada_get_hostname(url)
ada_get_pathname(url)
ada_get_search(url)
ada_get_protocol(url)
ada_get_domain(url)
ada_get_basename(url)
## privately registered suffixes count by default
ada_get_domain("https://foo.github.io/page")
ada_get_domain("https://foo.github.io/page", icann_only = TRUE)
## these functions are vectorized
urls <- c("http://www.google.com", "http://www.google.com:80", "noturl")
ada_get_port(urls)
```

ada_has_credentials	<i>Check if URL has a certain component</i>
---------------------	---

Description

These functions check if URL has a certain component.

Usage

```
ada_has_credentials(url)

ada_has_empty_hostname(url)

ada_has_hostname(url)

ada_has_non_empty_username(url)

ada_has_non_empty_password(url)

ada_has_port(url)

ada_has_hash(url)

ada_has_search(url)
```

Arguments

url	character. one or more URL to be parsed
-----	---

Value

logical, NA if not a valid URL.

Examples

```
url <- c("https://user_1:password_1@example.org:8080/dir/../../api?q=1#frag")
ada_has_credentials(url)
ada_has_empty_hostname(url)
ada_has_hostname(url)
ada_has_non_empty_username(url)
ada_has_non_empty_password(url)
ada_has_port(url)
ada_has_hash(url)
ada_has_search(url)
## these functions are vectorized
urls <- c("http://www.google.com", "http://www.google.com:80", "noturl")
ada_has_port(urls)
```

ada_set_href

Set a specific component of URL

Description

These functions set a specific component of URL.

Usage

```
ada_set_href(url, input, decode = TRUE)

ada_set_username(url, input, decode = TRUE)

ada_set_password(url, input, decode = TRUE)

ada_set_port(url, input, decode = TRUE)

ada_set_host(url, input, decode = TRUE)

ada_set_hostname(url, input, decode = TRUE)

ada_set_pathname(url, input, decode = TRUE)

ada_set_protocol(url, input, decode = TRUE)

ada_set_search(url, input, decode = TRUE)

ada_set_hash(url, input, decode = TRUE)
```

Arguments

url	character. one or more URL to be parsed
input	character. containing new component for URL. Vector of length 1 or same length as url.
decode	logical. Whether to decode the output (see utils::URLdecode()), default to TRUE

Value

character, NA if not a valid URL

Examples

```
url <- "https://user_1:password_1@example.org:8080/dir/./api?q=1#frag"
ada_set_href(url, "https://google.de")
ada_set_username(url, "user_2")
ada_set_password(url, "hunter2")
ada_set_port(url, "1234")
ada_set_hash(url, "#section1")
ada_set_host(url, "example.de")
ada_set_hostname(url, "example.de")
ada_set_pathname(url, "path/")
ada_set_search(url, "q=2")
ada_set_protocol(url, "ws:")
```

ada_url_parse	<i>Use ada-url to parse a url</i>
---------------	-----------------------------------

Description

Use ada-url to parse a url

Usage

```
ada_url_parse(url, decode = TRUE)
```

Arguments

url	character. one or more URL to be parsed
decode	logical. Whether to decode the output (see utils::URLdecode()), default to TRUE

Details

For details on the returned components refer to the introductory vignette.

Value

A data frame of the url components: href, protocol, username, password, host, hostname, port, pathname, search, and hash

Examples

```
ada_url_parse("https://user_1:password_1@example.org:8080/dir/../api?q=1#frag")
```

public_suffix

Extract the public suffix from a vector of domains or hostnames

Description

Extract the public suffix from a vector of domains or hostnames

Usage

```
public_suffix(domains, icann_only = FALSE)
```

Arguments

domains	character. vector of domains or hostnames
icann_only	logical. Whether to use only the ICANN section of the public suffix list, ignoring privately registered suffixes such as github.io or blogspot.com. Defaults to FALSE.

Details

domains may be either full URLs or bare hostnames; anything that does not parse as a URL is treated as a hostname. Wildcard (*) and exception (!) rules of the public suffix list are both honoured.

Value

public suffixes of domains as character vector

Examples

```
public_suffix("http://example.com")

# hostnames work too
public_suffix("example.com")

# for general URLs the hostname is extracted first
public_suffix("http://example.com/path/to/file")

# *.kobe.jp is a wildcard rule, !city.kobe.jp an exception to it
public_suffix(c("foo.kobe.jp", "city.kobe.jp"))
```

```
# privately registered suffixes are included by default
public_suffix("foo.github.io")
public_suffix("foo.github.io", icann_only = TRUE)
```

`url_decode2`*Function to percent-decode characters in URLs*

Description

Similar to [utils::URLdecode](#)

Usage

```
url_decode2(url)
```

Arguments

`url` a character vector

Details

A % that is not followed by two hexadecimal digits is not a valid escape sequence and is returned unchanged.

Value

percent decoded URLs as character vector

Examples

```
url_decode2("Hello%20World")
```

Index

ada_clear_hash (ada_clear_port), [2](#)
ada_clear_port, [2](#)
ada_clear_search (ada_clear_port), [2](#)
ada_get_basename (ada_get_href), [3](#)
ada_get_domain (ada_get_href), [3](#)
ada_get_hash (ada_get_href), [3](#)
ada_get_host (ada_get_href), [3](#)
ada_get_hostname (ada_get_href), [3](#)
ada_get_href, [3](#)
ada_get_password (ada_get_href), [3](#)
ada_get_pathname (ada_get_href), [3](#)
ada_get_port (ada_get_href), [3](#)
ada_get_protocol (ada_get_href), [3](#)
ada_get_search (ada_get_href), [3](#)
ada_get_username (ada_get_href), [3](#)
ada_has_credentials, [4](#)
ada_has_empty_hostname
 (ada_has_credentials), [4](#)
ada_has_hash (ada_has_credentials), [4](#)
ada_has_hostname (ada_has_credentials),
 [4](#)
ada_has_non_empty_password
 (ada_has_credentials), [4](#)
ada_has_non_empty_username
 (ada_has_credentials), [4](#)
ada_has_port (ada_has_credentials), [4](#)
ada_has_search (ada_has_credentials), [4](#)
ada_set_hash (ada_set_href), [5](#)
ada_set_host (ada_set_href), [5](#)
ada_set_hostname (ada_set_href), [5](#)
ada_set_href, [5](#)
ada_set_password (ada_set_href), [5](#)
ada_set_pathname (ada_set_href), [5](#)
ada_set_port (ada_set_href), [5](#)
ada_set_protocol (ada_set_href), [5](#)
ada_set_search (ada_set_href), [5](#)
ada_set_username (ada_set_href), [5](#)
ada_url_parse, [6](#)

public_suffix, [7](#)

url_decode2, [8](#)
utils::URLdecode, [8](#)
utils::URLdecode(), [2](#), [3](#), [6](#)