Package: cookie (via r-universe)

June 12, 2026

Title HTTP Cookies Parser Middleware

Version 1.0.0

Description A cookie is a piece of data sent from a web server to a web client which helps in overcoming the statelessness constraint of the HTTP protocol. This package provides the tools to work with them in the form of a cookie parser middleware function, meant to be attached to a bigger R web application, and utilities to write, sign and unsign a cookie. For more details see the Mozilla Developer Network (MDN) web documentation in <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Cookies>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports secretbase, utils

Suggests curl, httpuv, later, R6, routing, testthat (>= 3.0.0), yyjsonr

Config/testthat/edition 3

NeedsCompilation no

Author Julio Collazos [aut, cre] (ORCID: <https://orcid.org/0009-0006-5503-0997>), cookie contributors [ctb, cph] (cookie contributors; authors listed in <https://github.com/jshttp/cookie/graphs/contributors>), cookie-parser contributors [ctb, cph] (cookie-parser contributors; authors listed in <https://github.com/expressjs/cookie-parser/graphs/contributors>), cookie-signature contributors [ctb, cph] (cookie-signature contributors; authors listed in <https://github.com/tj/node-cookie-signature/graphs/contributors>), httpuv authors [ctb, cph] (Development of the http_date_string function)

Maintainer Julio Collazos <amarullazo626@gmail.com>

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

Date/Publication 2026-06-12 10:10:09 UTC

RemoteUrl https://github.com/cran/cookie

RemoteRef HEAD

RemoteSha 85a2e30bf24a0997939b7f33d3e840558cbe3331

Contents

`cookieParser` *Cookie Parser Middleware*

Description

Creates a middleware function that parses the `Cookie` HTTP header and populates `req$cookies`. When a `secret` is provided, signed cookies are verified and exposed on `req$signedCookies`.

Usage

```
cookieParser(secret = NULL, options = NULL)
```

Arguments

`secret`	A character vector or list of strings used to sign and verify cookies. Optional.
`options`	A function used to decode cookie values. Defaults to `utils::URLdecode()`.

Value

A middleware function that sets `req$cookies`, `req$signedCookies`, and `req$secret`, then calls `forward()`.

`serialise` *Serialise a Set-Cookie Header*

Description

Serialises a cookie name-value pair into a `Set-Cookie` header string.

Usage

```
serialise(name, val = NULL, ...)
```

Arguments

`name`	A string with the cookie name, or a list with `$name` and `$value` elements (in which case `val` is ignored).
`val`	A string. The cookie value.
`...`	Additional cookie attributes: `encode` A function to encode the cookie value. Defaults to `utils::URLencode()`. `maxAge` An integer. Number of seconds until the cookie expires. `domain` A string. The cookie domain. `path` A string. The cookie path. `expires` A `Date`, `POSIXct`, or `POSIXt`. The expiry date. `httpOnly` Logical. Adds the `HttpOnly` attribute. `secure` Logical. Adds the `Secure` attribute. `partitioned` Logical. Adds the `Partitioned` attribute. `priority` A string: `"low"`, `"medium"`, or `"high"`. `sameSite` A string (`"strict"`, `"lax"`, `"none"`) or logical (`TRUE` maps to `"Strict"`).

Value

A `Set-Cookie` header string.

Examples

```
serialise("session", "abc123")
serialise("id", "42", httpOnly = TRUE, secure = TRUE, sameSite = "lax")
```

sign *Sign a Cookie Value*

Description

Sign the given `val` with `secret`.

Usage

```
sign(val, secret)
```

Arguments

`val`	A string. The cookie value to sign.
`secret`	The secret key used to generate the signature.

Value

A string of the form `"<val>.<signature>"`.

Examples

```
sign("hello", "tobiiscool")
```

unsign *Unsign a Cookie Value*

Description

Verifies the signature of a signed cookie value and returns the original value if valid, or `FALSE` if the signature does not match.

Usage

```
unsign(input, secret)
```

Arguments

`input`	A string. A signed cookie value produced by `sign()`.
`secret`	The secret key to verify against.

Value

The original unsigned value if verification succeeds, `FALSE` otherwise.

Examples

```
input <- sign("hello", "tobiiscool")
unsign(input, "tobiiscool")
unsign(input, "luna")
```

Index