Package 'punycoder'

June 16, 2026

Type Package

Title Unicode and Punycode Domain Name Processing

Version 1.1.0

Description High-performance Unicode and Punycode encoding/decoding for internationalized domain names. Provides RFC 3492 compliant conversion functions with a focus on URL processing and data analysis workflows. Addresses limitations in existing R packages for handling international domain names in web scraping and URL parsing applications.

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0)

LinkingTo Rcpp

SystemRequirements GNU libidn2 (optional, for native punycode backend)

License MIT + file LICENSE

URL https://github.com/bart-turczynski/punycoder, https://CRAN.R-project.org/package=punycoder

BugReports https://github.com/bart-turczynski/punycoder/issues

Encoding UTF-8

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Bart Turczynski [aut, cre]

Maintainer Bart Turczynski <bartek+punycoder@turczynski.pl>

Repository CRAN

Date/Publication 2026-06-16 07:00:02 UTC

Contents

`punycoder-package` *Unicode and Punycode Domain Name Processing*

Description

Provides high-performance functions for encoding and decoding internationalized domain names according to RFC 3492 (Punycode) and IDNA standards.

Details

The punycoder package fills a critical gap in R's ecosystem for handling international domain names. It provides reliable, fast conversion between Unicode and ASCII representations of domain names.

Author(s)

Maintainer: Bart Turczynski <bartek+punycoder@turczynski.pl>

Authors:

- Bart Turczynski <bartek+punycoder@turczynski.pl>

See Also

Useful links:

- https://github.com/bart-turczynski/punycoder
- https://CRAN.R-project.org/package=punycoder
- Report bugs at https://github.com/bart-turczynski/punycoder/issues

host_normalize *Normalize hosts to canonical comparison form*

Description

Converts DNS hostnames to their canonical comparison form following the ratified canonical-host normalization contract: Unicode NFC, case mapping, UTS-46 label mapping and validation (non-transitional, with 'UseSTD3ASCIIRules', 'CheckHyphens', 'CheckBidi', and 'CheckJoiners'), conversion to lowercase ASCII A-labels, and DNS length verification, while preserving whether the input carried a single terminal root dot.

Usage

```
host_normalize(x, strict = TRUE)
```

Arguments

x	Character vector of hostnames. 'NA' elements pass through as 'NA' (missing, not invalid). Names are preserved.
strict	Logical scalar. 'TRUE' (the default) applies the full profile. 'strict = FALSE' is reserved for a future documented relaxed variant and currently behaves identically to 'TRUE'. This function never reads the 'punycoder.strict' option.

Details

Unlike [puny_encode()], invalid input is reported by returning 'NA_character_' (never by aborting), so a caller can layer its own policy. The profile is fixed at one pinned Unicode version per release; see [normalization_profile_info()] for the machine-readable identity.

Value

A character vector the same length as 'x'. Each element is the canonical lowercase ASCII A-label host, or 'NA_character_' when the input is 'NA' or invalid under the profile.

See Also

[normalization_profile_info()] for the profile identity, [puny_encode()] for the lower-level RFC 3492 transform.

Examples

```
host_normalize(c("Example.COM", "münchen.de", "example.com."))
host_normalize("a_b.com") # NA: STD3 rejects "_"
```

is_idn *Test if domain contains internationalized characters*

Description

Determines whether a domain name contains Unicode characters that would require punycode encoding for ASCII compatibility.

Usage

```
is_idn(x)
```

Arguments

x	Character vector of domain names to test

Value

A logical vector the same length as x, where TRUE indicates the element contains non-ASCII Unicode characters.

See Also

is_punycode for detecting punycode domains, puny_encode for encoding Unicode domains.

Examples

```
is_idn("caf\u00E9.com") # TRUE
is_idn("example.com")    # FALSE
is_idn(c(
  "caf\u00E9.com",
  "\u043C\u043E\u0441\u043A\u0432\u0430.\u0440\u0444",
  "test.com"
))  # c(TRUE, TRUE, FALSE)
```

is_punycode *Test if string is punycode encoded*

Description

Determines whether a given string or domain name is already encoded in punycode format (starts with xn– prefix).

Usage

```
is_punycode(x)
```

Arguments

x	Character vector to test

Value

A logical vector the same length as x, where TRUE indicates the element contains a punycode-encoded label (xn– prefix).

See Also

is_idn for detecting Unicode domains, puny_decode for decoding punycode domains.

Examples

```
is_punycode("xn--example") # TRUE
is_punycode("example.com") # FALSE
is_punycode(c("xn--caf-dma.com", "regular.com"))  # c(TRUE, FALSE)
```

---

normalization_profile_info

Canonical-host normalization profile identity

Description

Returns the stable, machine-readable identity of the normalization profile applied by [host_normalize()]. Downstream packages read ‘profile‘ and ‘unicode_version‘ to key reproducibility on the exact normalization behavior; the ‘backend‘ column is diagnostic only and must never enter a reproducibility or cache key.

Usage

```
normalization_profile_info()
```

Value

A one-row ‘data.frame‘ with columns ‘profile‘, ‘unicode_version‘, ‘idna‘, ‘transitional‘, ‘use_std3‘, ‘check_hyphens‘, ‘check_bidi‘, ‘check_joiners‘, and ‘backend‘.

See Also

[host_normalize()].

Examples

```
normalization_profile_info()
```

parse_url *Parse URLs with internationalized domain name handling*

Description

Parses URLs and returns a structured list with proper handling of internationalized domain names. This function provides both Unicode and ASCII representations of domain components.

Usage

```
parse_url(url, encode_domains = FALSE)
```

Arguments

url	Character vector of URLs to parse
encode_domains	Logical flag; encode parsed host names to ASCII.

Value

An object of class "punycoder_parsed_url" (a named list) with components:

scheme Character vector of URL schemes (e.g., "https").

domain Character vector of domain names.

port Integer vector of port numbers.

path Character vector of URL paths.

query Character vector of query strings.

fragment Character vector of fragment identifiers.

Each component has one element per input URL. Invalid URLs yield NA components. For valid URLs without an explicit path, path is returned as "".

See Also

url_encode, url_decode for URL transformation with IDN handling.

Examples

```
# Parse URL with Unicode domain
parse_url(
  "https://caf\u00E9.example.com:8080/path?query=value#fragment"
)

# Parse multiple URLs
urls <- c(
  "https://caf\u00E9.com/menu",
  "https://\u043C\u043E\u0441\u043A\u0432\u0430.\u0440\u0444/info"
)
parse_url(urls)
```

print.punycoder_parsed_url

Print method for punycoder parsed URL results

Description

Print method for punycoder parsed URL results

Usage

```
## S3 method for class 'punycoder_parsed_url'
print(x, ...)
```

Arguments

x	A punycoder_parsed_url object
...	Additional arguments (ignored)

Value

Invisibly returns x.

print.punycoder_validation

Print method for punycoder validation results

Description

Print method for punycoder validation results

Usage

```
## S3 method for class 'punycoder_validation'
print(x, ...)
```

Arguments

x	A punycoder_validation object
...	Additional arguments (ignored)

Value

Invisibly returns x.

puny_decode *Decode ASCII punycode to Unicode domains*

Description

Converts ASCII punycode domain names back to their Unicode representation. This is the reverse operation of puny_encode and is useful for displaying human-readable domain names.

Usage

```
puny_decode(x, strict = getOption("punycoder.strict", TRUE))
```

Arguments

x	Character vector of ASCII punycode domains to decode
strict	Logical; whether to apply strict validation. Defaults to ‘getOption("punycoder.strict", TRUE)‘.

Value

A character vector the same length as x, with each element containing the Unicode-decoded domain name. Elements corresponding to NA inputs are NA_character_. In non-strict mode, domains that fail decoding are also returned as NA_character_.

See Also

puny_encode for the reverse operation, url_decode for full URL decoding.

Examples

```
# Basic decoding
puny_decode("xn--caf-dma.com")
puny_decode("xn--80adxhks.xn--p1ai")

# Vectorized decoding
ascii_domains <- c("xn--caf-dma.com", "xn--80adxhks.xn--p1ai")
puny_decode(ascii_domains)
```

puny_encode *Encode Unicode domains to ASCII punycode*

Description

Converts Unicode domain names to their ASCII punycode representation following RFC 3492 standards. This function is essential for processing internationalized domain names (IDNs) in web scraping and URL analysis.

Usage

```
puny_encode(x, strict = getOption("punycoder.strict", TRUE))
```

Arguments

x	Character vector of Unicode domain names to encode
strict	Logical; whether to apply strict validation. Defaults to 'getOption("punycoder.strict", TRUE)'.

Value

A character vector the same length as x, with each element containing the ASCII punycode-encoded domain name. Elements corresponding to NA inputs are NA_character_. In non-strict mode, domains that fail encoding are also returned as NA_character_.

See Also

puny_decode for the reverse operation, url_encode for full URL encoding.

Examples

```
# Basic encoding
puny_encode("caf\u00E9.com")
puny_encode("\u043C\u043E\u0441\u043A\u0432\u0430.\u0440\u0444")

# Vectorized encoding
domains <- c(
  "caf\u00E9.com",
  "\u043C\u043E\u0441\u043A\u0432\u0430.\u0440\u0444",
  "\u5317\u4EAC.\u4E2D\u56FD"
)
puny_encode(domains)
```

`url_decode` *Decode URLs with ASCII punycode domains to Unicode*

Description

Converts URLs containing ASCII punycode domain names back to their Unicode representation for display purposes. This function makes internationalized URLs human-readable.

Usage

```
url_decode(url, strict = getOption("punycoder.strict", TRUE))
```

Arguments

`url`	Character vector of URLs with ASCII punycode domains
`strict`	Logical; whether to apply strict validation. Defaults to 'getOption("punycoder.strict", TRUE)'.

Value

A character vector the same length as `url`, with each element containing the URL with its host portion decoded to Unicode. Only the domain component is transformed; scheme, path, query, and fragment are preserved. Elements corresponding to `NA` inputs are `NA_character_`.

See Also

`url_encode` for the reverse operation, `puny_decode` for domain-only decoding, `parse_url` for URL component extraction.

Examples

```
# Basic URL decoding
url_decode("https://xn--caf-dma.example.com/path")
url_decode("https://xn--80adxhks.xn--p1ai/page")

# Vectorized URL decoding
ascii_urls <- c(
  "https://xn--caf-dma.com/menu",
  "https://xn--1qqw23a.xn--55qx5d/info"
)
url_decode(ascii_urls)
```

`url_encode` *Encode URLs with Unicode domains to ASCII*

Description

Converts URLs containing Unicode domain names to their ASCII representation while preserving the rest of the URL structure. This function is essential for preparing URLs for systems that require ASCII-only domain names.

Usage

```
url_encode(url, strict = getOption("punycoder.strict", TRUE))
```

Arguments

`url`	Character vector of URLs with potential Unicode domains
`strict`	Logical; whether to apply strict validation. Defaults to ‘getOption("punycoder.strict", TRUE)‘.

Value

A character vector the same length as `url`, with each element containing the URL with its host portion ASCII-encoded. Only the domain component is transformed; scheme, path, query, and fragment are preserved. Elements corresponding to `NA` inputs are `NA_character_`.

See Also

`url_decode` for the reverse operation, `puny_encode` for domain-only encoding, `parse_url` for URL component extraction.

Examples

```
# Basic URL encoding
url_encode("https://caf\u00E9.example.com/path?query=value")
url_encode(
  "https://\u043C\u043E\u0441\u043A\u0432\u0430.\u0440\u0444/page"
)

# Vectorized URL encoding
urls <- c(
  "https://caf\u00E9.com/menu",
  "https://\u5317\u4EAC.\u4E2D\u56FD/info"
)
url_encode(urls)
```

validate_domain *Comprehensive domain name validation*

Description

Validates domain names according to RFC standards, checking for proper format, length restrictions, and character requirements. Supports both Unicode and ASCII domain names.

Usage

```
validate_domain(x, strict = getOption("punycoder.strict", TRUE))
```

Arguments

x	Character vector of domain names to validate
strict	Logical; whether to apply strict validation. Defaults to ‘getOption("punycoder.strict", TRUE)‘.

Value

An object of class "punycoder_validation" (a named list) with components:

domains Character vector of the input domain names.

valid Logical vector indicating whether each domain is valid.

errors List of character vectors, each containing error messages for the corresponding domain (empty for valid domains).

See Also

puny_encode for encoding validated domains.

Examples

```
validate_domain("example.com")
validate_domain("caf\u00E9.example.com")
long_label <- paste(rep("x", 250), collapse = "")
validate_domain(c("valid.com", "invalid..com", long_label))
```

Index