Package 'punycoder'

July 19, 2026

Type Package

Title Unicode and Punycode Domain Name Processing

Version 1.2.1

Description High-performance Unicode and Punycode processing for internationalized domain names. The 'puny_encode()' / 'puny_decode()' helpers are a low-level, RFC 3492 compliant Punycode codec for domain labels (the 'xn--' ASCII-Compatible Encoding of RFC 5890/5891); they perform the raw transform plus letter-digit-hyphen checks and do not apply Unicode IDNA normalization. 'host_normalize()' is the Unicode Technical Standard #46 host-normalization entry point, mapping a host name to a canonical lowercase ASCII comparison form (non-transitional profile, pinned Unicode version). The 'url_encode()' / 'url_decode()' / 'parse_url()' helpers do best-effort host extraction and rewriting in URL-shaped strings and are deliberately not RFC 3986 / WHATWG URL parsers or canonicalizers; they are deprecated in favor of dedicated URL packages. Aimed at host normalization and data analysis workflows. Used as the Punycode and IDNA engine by the 'pslr' and 'rurl' packages.

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://bart-turczynski.github.io/punycoder/, https://github.com/bart-turczynski/punycoder, https://bart-turczynski.r-universe.dev/punycoder, https://CRAN.R-project.org/package=punycoder

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

Encoding UTF-8

Language en-US

Suggests testthat (>= 3.0.0), knitr, rmarkdown, oysteR, rosv

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

X-schema.org-keywords punycode, punycode encode, punycode decode, IDNA, IDN, internationalized domain names, UTS-46, RFC 3492, Unicode, host normalization, hostname, domain names, TLD

NeedsCompilation yes

Author Bart Turczynski [aut, cre] (ORCID: <https://orcid.org/0000-0002-8788-7980>)

Maintainer Bart Turczynski <bartek@turczynski.pl>

Repository CRAN

Date/Publication 2026-07-19 18:10:02 UTC

Contents

punycoder-package 2
host_normalize 3
is_idn 5
is_punycode 5
normalization_profile_info 6
print.punycoder_validation 7
puny_decode 8
puny_encode 9
validate_domain 10

Index **11**

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

Description

Provides high-performance functions for processing internationalized domain names, split across two tiers.

Details

The package exposes two distinct surfaces, deliberately kept separate:

- A **low-level Punycode codec** (`puny_encode()` / `puny_decode()`): the raw RFC 3492 transform with `xn--` A-label framing (RFC 5890/5891) and letter-digit-hyphen checks. It performs no Unicode normalization.
- An **IDNA/UTS-46 host-normalization surface** (`host_normalize()`): Unicode NFC, UTS #46 mapping and validation, and conversion to a canonical lowercase ASCII comparison form under a pinned profile.

Use the codec when you need the literal ASCII-Compatible Encoding of a label; use `host_normalize()` when you need a standards-profiled comparison form for a host name.

Author(s)

Maintainer: Bart Turczynski <bartek@turczynski.pl> (ORCID)

Authors:

- Bart Turczynski <bartek@turczynski.pl> (ORCID)

See Also

Useful links:

- https://bart-turczynski.github.io/punycoder/
- https://github.com/bart-turczynski/punycoder
- https://bart-turczynski.r-universe.dev/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,
  check_hyphens = TRUE,
  use_std3 = TRUE,
  verify_dns_length = TRUE
)
```

Arguments

x	Character vector of hostnames. NA elements pass through as NA (missing, not invalid). Names are preserved.
check_hyphens	Logical scalar. When TRUE (the default) the UTS #46 CheckHyphens rule rejects "--" in the 3rd/4th positions and leading or trailing hyphens. FALSE drops that check.
use_std3	Logical scalar. When TRUE (the default) UseSTD3ASCIIRules restricts ASCII to letters, digits, and hyphen. FALSE admits other ASCII (e.g. "_") that the pinned table marks STD3-disallowed-but-valid.
verify_dns_length	Logical scalar. When TRUE (the default) each A-label must be 1-63 octets and the whole host <= 253. FALSE drops the length limits (empty labels are still rejected as structural errors).

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.

This is a **UTS #46 profile, not IDNA2008 / RFC 5891 conformance.** UTS #46 is compatibility processing and deliberately differs from IDNA2008 — it accepts labels IDNA2008 would reject (e.g. a label whose first character is the symbol U+2615 HOT BEVERAGE becomes "xn--53h.example"). The pipeline draws on RFC 3492 (the Punycode transform), NFC per UAX #15, the RFC 5892 ContextJ rules via CheckJoiners (ZWJ/ZWNJ only — full RFC 5892 CONTEXTO is **not** checked), the RFC 5893 Bidi rule via CheckBidi, and STD 3 (RFC 952 + RFC 1123) host-name rules via UseSTD3ASCIIRules. IDNA2003 / Nameprep (RFC 3490/3491/3454) is not used.

The default applies the full strict UTS #46 profile (uts46-nontransitional-std3-v1). The check_hyphens, use_std3, and verify_dns_length arguments are UTS #46 processing flags that can each be relaxed independently; pass the *same* values to normalization_profile_info() to obtain the identity of the resulting profile. These are standard UTS #46 parameters, **not** a browser mode: CheckBidi and CheckJoiners always apply and are never knobs, and full WHATWG host policy (where beStrict = false flips exactly these three) lives upstack in rurl, not here.

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 "_"
host_normalize("a_b.com", use_std3 = FALSE) # "a_b.com"
```

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 a normalization profile. Called with no arguments it reports the default (fully strict) profile host_normalize() applies; the check_hyphens, use_std3, and verify_dns_length arguments report the identity of a specific flag set so a caller can describe the exact profile a given normalization used. Downstream packages key reproducibility on the full per-parameter column set; profile is a coarse cache token (distinct per flag set, but no longer load-bearing alone) and the backend column is diagnostic only and must never enter a reproducibility or cache key.

Usage

```
normalization_profile_info(
  check_hyphens = TRUE,
  use_std3 = TRUE,
  verify_dns_length = TRUE
)
```

Arguments

check_hyphens, use_std3, verify_dns_length	Logical scalars selecting the flag set to report. Each defaults to TRUE (the strict profile).

Details

`check_bidi`, `check_joiners`, and `transitional` are fixed by the profile (UTS #46 non-transitional, both bidi and joiner checks always on) and are reported as constant columns rather than arguments.

Value

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

See Also

`host_normalize()`.

Examples

```
normalization_profile_info()
normalization_profile_info(use_std3 = FALSE)
```

`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`.

Examples

```
result <- validate_domain(c("example.com", "xn--bad-label-"))
print(result)
```

puny_decode *Decode ASCII Punycode to Unicode domain labels (low-level)*

Description

Converts ASCII Punycode (`xn--`) domain names back to their Unicode representation. This is the inverse of `puny_encode()` and is the raw RFC 3492 transform with A-label framing checks. DNS host length limits are intentionally not applied by this raw codec; use `validate_domain()` or `host_normalize()` when you need DNS host validation.

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)`. In strict mode the raw codec enforces structural checks but not DNS host length limits.

Details

Like `puny_encode()`, this is a **low-level ASCII-Compatible Encoding helper, not an IDNA normalization API**: it does not apply UTS #46 mapping or NFC. For IDNA/UTS-46 host normalization, see `host_normalize()`.

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, `host_normalize` for IDNA/UTS-46 host normalization, `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 domain labels to ASCII Punycode (low-level)*

Description

Converts Unicode domain names to their ASCII Punycode (`xn--`) representation: the raw RFC 3492 Bootstring transform wrapped in the RFC 5890/5891 A-label framing, plus letter-digit-hyphen and leading/trailing hyphen checks per label. DNS host length limits are intentionally not applied by this raw codec; use `validate_domain()` or `host_normalize()` when you need DNS host validation.

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)`. In strict mode the raw codec enforces structural checks but not DNS host length limits.

Details

This is a **low-level ASCII-Compatible Encoding helper, not an IDNA normalization API.** It does *not* apply Unicode NFC, UTS #46 mapping, case folding, or Bidi/Joiner validation. To map a host name to its canonical comparison form under a UTS #46 profile (the IDNA surface of this package), use `host_normalize()`.

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, `host_normalize` for IDNA/UTS-46 host normalization, `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)
```

`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).

error_codes List of character vectors, each containing stable machine-readable error codes for the corresponding domain (empty for valid domains). Missing input uses `"domain_na"`.

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

host_normalize, 3, *8, 9*
host_normalize(), *2, 3*, *6–9*

is_idn, 5, *6*
is_punycode, *5*, 5

normalization_profile_info, 6
normalization_profile_info(), *4*

print.punycoder_validation, 7
puny_decode, *6*, 8, *9*
puny_decode(), *2*
puny_encode, *5*, *8*, 9, *10*
puny_encode(), *2*, *4*, *8*
punycoder *(*punycoder-package*)*, 2
punycoder-package, 2

url_decode, *8*
url_encode, *9*

validate_domain, 10
validate_domain(), *8, 9*