Package 'charport'

November 6, 2026

Title ALTREP String Interoperability

Version 0.2.0

Date 2026-11-06

Description Provides infrastructure for interoperable ALTREP character vectors. Producers of ALTREP string classes can register access methods, allowing consumers to read supported character vectors through a common interface without materializing them as ordinary R strings. Also provides 'charvec', a reference ALTREP string implementation backed by stable memory slices, with support for efficient and multithreaded construction.

License MIT + file LICENSE

Copyright Benchmark measurements use the enwik8 corpus, the first 10^8 bytes of the English Wikipedia XML dump of 2006-03-03, whose text carries Wikipedia's own terms: the Creative Commons Attribution-ShareAlike License and the GNU Free Documentation License. No Wikipedia data is included in this package.

Encoding UTF-8

Suggests cpp11, quarto, Rcpp

VignetteBuilder quarto

Depends R (>= 3.6.0)

URL https://github.com/charbase/charport, https://charbase.github.io/charport/

BugReports https://github.com/charbase/charport/issues

Config/roxygen2/version 8.1.0

Config/Needs/website pkgdown, xml2

NeedsCompilation yes

Author Travers Ching [aut, cre, cph], R Consortium [fnd] (Infrastructure Steering Committee grant: Universal ALTREP Interoperability for Strings)

Maintainer Travers Ching <traversc@gmail.com>

Repository CRAN
Date/Publication 2026-11-06 10:40:02 UTC

Contents

as_charpack 2
as_charvec 3
charport_classes 3
charport_class_of 4
charport_info 5
charport_materialize 5
charvec 6
is_charpack 7
is_charvec 7

Index **8**

`as_charpack` *Convert to a charpack*

Description

Converts a character vector (or anything `as.character()` accepts) to a `charpack`, copying its strings. If `x` is already a `charpack` it is returned unchanged. Names are preserved; other attributes are dropped.

Usage

```
as_charpack(x)
```

Arguments

`x`	object to convert.

Details

A `charpack` stays compact while it is read through charport, and vector subsets such as `x[i]` stay packed. Reading individual strings from R, such as `x[[1]]` or `print(x)`, extracting matrix columns, or modifying elements materializes the whole vector. This makes it mainly useful for testing code that reads strings through charport.

Value

A `charpack` (an ALTREP character vector).

Examples

```
x <- as_charpack(c("a", NA, "b"))
is_charpack(x)
```

as_charvec *Convert to a charvec*

Description

Converts a character vector (or anything `as.character()` accepts) to a `charvec`. If `x` is already a `charvec` it is returned unchanged. Names are preserved; other attributes are dropped.

Usage

```
as_charvec(x)
```

Arguments

`x`	object to convert.

Value

A `charvec` (an ALTREP character vector).

Examples

```
as_charvec(letters)
```

charport_classes *Registered charport ALTREP classes*

Description

Reports on the broker's ALTREP class registry. Registered classes are ALTREP character vector classes whose authors registered a reader with charport (via the `charport_register_altrep_v1` C entry point, fetched with `R_GetCCallable`). The built-in `charvec` and `charpack` classes are registered when `charport` loads, so a freshly loaded session normally reports at least those two classes.

Usage

```
charport_classes()
```

Details

ALTREP class names require an instance to query (R's `R_altrep_class_name` takes a vector, not a class descriptor), so this registry view reports the count and capability flags; use `charport_class_of()` on a vector to get its registered class name.

Value

A list with elements `n` (integer: number of registered classes), `persistent_views` (logical vector: whether returned byte views remain valid until the reader borrow ends), `concurrent_access` (logical vector: whether reader access calls may run concurrently), and `reentrant` (logical vector: whether both capabilities are true).

Examples

```
charport_classes()
```

---

charport_class_of *Registered ALTREP class serving a character vector*

Description

Identifies whether a registered ALTREP class claims `x`. This is a class-membership question answered without touching the vector's data: it never materializes `x` and reports a match even when the class reader would decline to serve this particular instance (for example, a materialized `charvec`).

Usage

```
charport_class_of(x)
```

Arguments

`x`	a character vector.

Value

`"package::class"` for a registered class match when class metadata is available; otherwise `NA_character_`.

Examples

```
charport_class_of(charvec("a"))
charport_class_of(letters)
```

charport_info *Character vector diagnostics*

Description

Reports non-forcing diagnostics for a possible character vector. This is a preflight/development helper: it does not call `STRING_ELT()`, `STRING_PTR_RO()`, `DATAPTR()`, or any registered reader callback.

Usage

```
charport_info(x)
```

Arguments

`x`	object to inspect.

Details

`is_materialized` means ordinary string pointer storage is available without forcing (`DATAPTR_OR_NULL(x) != NULL`). For base R deferred strings, some elements may have been cached by `STRING_ELT()` while this still reports `FALSE`; the field is a full-materialization/direct-pointer diagnostic.

Value

A named list containing `is_strsxp`, `length`, `is_altrep`, `is_materialized`, `is_registered`, reader capability flags, `stateful_reader`, `reentrant`, ALTREP class name/package fields, and `altrep_class` as `"package::class"` when class metadata is available.

Examples

```
charport_info(charvec("a"))
```

charport_materialize *Force materialization of a character vector*

Description

Forces a `charvec` or `charpack` to materialize its R-level strings (`CHARSXP`s), caching them on the object; the native store or raw payload is released. Ordinary character vectors are returned unchanged. This is a diagnostic/escape hatch: code that needs guaranteed-plain string storage (for example, before handing a vector to C code that bypasses ALTREP accessors) can call it explicitly.

Usage

```
charport_materialize(x)
```

Arguments

x	a character vector (plain, charvec or charpack).

Value

x, invisibly, after forcing materialization.

Examples

```
x <- charvec("a", "b")
charport_materialize(x)
```

---

charvec *Construct a charvec*

Description

Builds a charvec, charport's reference ALTREP character vector class, from the given values. A charvec is an ordinary character vector to R code (typeof(x) is "character"); its strings live as byte views in stable native memory blocks and are only converted to R's interned CHARSXP strings when something forces materialization.

Usage

```
charvec(...)
```

Arguments

...	values to combine, as in c(); non-character values are coerced with as.character().

Details

Element bytes and encoding marks are preserved verbatim. charvec is a storage/reference class, not an encoding-normalization layer; translation policy belongs in consumers built above charport. NA_character_ is preserved.

Value

A charvec (an ALTREP character vector).

Examples

```
x <- charvec("hello", "world", NA)
is_charvec(x)
x[1]
```

is_charpack *Test for a charpack*

Description

A charpack is a compact ALTREP character vector whose strings live in one raw vector. Packages create them through the C API.

Usage

```
is_charpack(x)
```

Arguments

x	object to test.

Value

TRUE if x is a charpack ALTREP vector, FALSE otherwise. A materialized charpack is still a charpack.

Examples

```
is_charpack(letters)
```

is_charvec *Test for a charvec*

Description

Test for a charvec

Usage

```
is_charvec(x)
```

Arguments

x	object to test.

Value

TRUE if x is a charvec ALTREP vector, FALSE otherwise. Note a materialized charvec is still a charvec; serialization of a materialized charvec round-trips to a plain character vector.

Examples

```
is_charvec(charvec("a"))
is_charvec(letters)
```

Index

as.character(), *2, 3*, *6*
as_charpack, 2
as_charvec, 3

c(), *6*
charport_class_of, 4
charport_class_of(), *3*
charport_classes, 3
charport_info, 5
charport_materialize, 5
charvec, 6

is_charpack, 7
is_charvec, 7