

Package: rzig (via r-universe)

October 9, 2026

Type Package

Title Safe Zig Bindings

Version 0.3.0

Description Scaffolds packages backed by Zig and generates a safe native boundary for plain Zig functions. The generated integration converts objects at the language boundary, registers native routines, propagates errors, and compiles across platforms without requiring C++. Each step of the workflow is a function: locating a supported Zig compiler, scaffolding the package, scanning the Zig source for exported functions, rendering and writing the bindings, checking the package before compilation, and building it with the compiler errors reported by file and line. The scan, the checks and the generated bindings are pure R; a Zig compiler is needed only to build a package created with the scaffold.

License GPL-3

URL <https://github.com/MarekDejaUJ/rzig>

BugReports <https://github.com/MarekDejaUJ/rzig/issues>

Encoding UTF-8

SystemRequirements Zig 0.16.x or 0.17.x
(<https://ziglang.org/download/>), needed to install packages
created with rzig

NeedsCompilation no

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Repository <https://marekdejauj.r-universe.dev>

Date/Publication 2026-10-09 09:41:39 UTC

RemoteUrl <https://github.com/marekdejauj/rzig>

RemoteRef HEAD

RemoteSha 0a83b3925a78f7ea967f4118e65aaedf1c8326a

Contents

rzig-package	2
build_zig	2
document	4
find_zig	5
render_bindings	5
rzig_status	6
scan_exports	7
use_rzig	8

Index	10
--------------	-----------

rzig-package	<i>Safe Zig Bindings</i>
--------------	--------------------------

Description

RZig scaffolds packages backed by Zig and generates safe native bindings for plain Zig functions. Each step of the workflow is a function: `find_zig` locates a supported Zig compiler, `use_rzig` adds the build files and the framework sources to a package, `scan_exports` lists the exported Zig functions with their R arguments, `render_bindings` shows the generated manifest, R wrappers and NAMESPACE block, `document` writes them, `rzig_status` reports which steps are complete and which signatures the boundary rejects, and `build_zig` compiles and installs the package, reporting compiler errors by file and line. The scan and the generated bindings are pure R; a Zig compiler of the 0.16 or 0.17 release series is needed only to install a package created with the scaffold.

Author(s)

Marek Deja

build_zig	<i>Compile and install a Zig-backed package, and report the first errors</i>
-----------	--

Description

Runs the checks of `[rzig_status()]`, then installs a copy of the package with ‘R CMD INSTALL’ into a library, using the Zig compiler reported by `[find_zig()]`. The complete installation log is kept, and the errors of the Zig compiler, the C compiler, the linker and the load test are extracted with their files and lines, so a failed build names its cause instead of the bare "non-zero exit status" of `[utils::install.packages()]`.

Usage

`build_zig(path, lib = NULL, preflight = TRUE)`

Arguments

<code>path</code>	Path to the package root.
<code>lib</code>	The library to install into. By default a new library in the session temporary directory; <code>[library()]</code> then needs <code>'lib.loc = build\$library'</code> .
<code>preflight</code>	If <code>'TRUE'</code> , run the checks of <code>[rzig_status()]</code> first and stop before compiling when one of them fails.

Details

The package is built from a copy in the session temporary directory, so the package source receives no object files or compiler caches. The installation needs a supported Zig compiler and the C toolchain of R. A package that is already loaded in the R session keeps its old native code until R restarts; the printed result says so.

Value

An object of class `'rzig_build'`: a list with `'ok'` (logical), `'package'`, `'step'` (`"preflight"`, `"configure"`, `"compile"`, `"link"`, `"load"`, `"install"` or `'NA'` on success), `'errors'` (a data frame with the columns `'file'`, `'line'`, `'column'`, `'message'` and `'function_name'`), `'problems'` (failed checks before compilation), `'library'`, `'zig'` (the `[rzig_compiler][find_zig()]` used), `'seconds'` and `'log'` (the complete output of `'R CMD INSTALL'`). A failed build also signals a warning.

Examples

```
package_path <- tempfile("rzig-build-", tmpdir = tempdir())
dir.create(package_path)
writeLines(c("Package: examplepkg", "Version: 0.0.1"),
  file.path(package_path, "DESCRIPTION"))
use_rzig(package_path)

# A change to main.zig without document() is caught before compiling.
main <- file.path(package_path, "src", "rzig", "src", "main.zig")
cat("/// @export\npub fn twice(x: f64) f64 {\n    return 2 * x;\n}\n",
  file = main, append = TRUE)
result <- suppressWarnings(build_zig(package_path))
result

# With a supported Zig compiler, the regenerated package compiles; the
# compilation takes several seconds.

if (isTRUE(find_zig(required = FALSE)$supported)) {
  document(package_path)
  result <- build_zig(package_path)
  result
}
```

```
unlink(package_path, recursive = TRUE)
```

document*Generate R bindings from Zig exports*

Description

Scans the Zig source of the package with `[scan_exports()]`, renders the bindings with `[render_bindings()]`, and writes them: the Zig manifest `'src/rzig/framework/generated/manifest.zig'`, the R wrappers `'R/rzig-wrappers.R'`, and a managed block in `'NAMESPACE'`. Lines of `'NAMESPACE'` outside the block, for example directives written by `roxygen2`, are kept. The function is pure R; no Zig compiler is needed.

Usage

```
document(path)
```

Arguments

path	Path to a package previously initialized with <code>[use_rzig()]</code> .
------	---

Details

Run `'document()'` after every change to an exported Zig signature or its documentation comment, and before installing the package.

Value

The `[rzig_bindings][render_bindings()]` object that was written, with the element `'files'` naming the files written, invisibly.

Examples

```
package_path <- tempfile("rzig-document-", tmpdir = tempdir())
dir.create(package_path)
writeLines(c("Package: examplepkg", "Version: 0.0.1"),
  file.path(package_path, "DESCRIPTION"))
use_rzig(package_path)
bindings <- document(package_path)
bindings$files
cat(readLines(file.path(package_path, "NAMESPACE")), sep = "\n")
unlink(package_path, recursive = TRUE)
```

find_zig	<i>Locate a supported Zig compiler</i>
----------	--

Description

Searches the environment variable ‘ZIG’, the ‘PATH’, ‘~/local/share/zig/*/zig’ and ‘~/zig/zig’, in that order, and reports the first Zig executable found, its version and whether rzig supports that version. rzig itself runs without Zig: [use_rzig()], [scan_exports()] and [document()] are pure R. A Zig compiler of the supported release series is needed to install a package created with [use_rzig()].

Usage

```
find_zig(required = TRUE)
```

Arguments

required	If ‘TRUE’, a missing or unsupported compiler is an error. If ‘FALSE’, the returned object records the problem instead.
----------	--

Details

rzig 0.3.0 supports the Zig 0.16 and 0.17 release series (versions ‘0.16.x’ and ‘0.17.x’). Zig changes its language and standard library between release series, so a newer or older compiler is reported as unsupported.

Value

An object of class ‘rzig_compiler’: a list with ‘path’ (character or ‘NULL’), ‘version’ (character or ‘NULL’), ‘supported’ (logical) and ‘series’ (the supported release series, “0.16.x or 0.17.x”).

Examples

```
find_zig(required = FALSE)
```

render_bindings	<i>Render the generated bindings for a set of exported functions</i>
-----------------	--

Description

Produces the three texts that [document()] writes into a package, without writing anything: the Zig manifest that registers the exported functions, the R wrappers that call them, and the ‘NAMESPACE’ block that resolves the native symbols. Use the function to inspect what [document()] will generate.

Usage

```
render_bindings(exports, package = attr(exports, "package", exact = TRUE))
```

Arguments

exports	An object returned by [scan_exports()].
package	The package name; by default taken from ‘exports’.

Value

An object of class ‘rzig_bindings’: a list with the character strings ‘manifest’ (‘src/rzig/framework/generated/manifest.zig’), ‘wrappers’ (‘R/rzig-wrappers.R’) and ‘namespace’ (the generated ‘NAMESPACE’ block), and the ‘exports’ the texts were rendered from.

Examples

```
package_path <- tempfile("rzig-render-", tmpdir = tempdir())
dir.create(package_path)
writeLines(c("Package: examplepkg", "Version: 0.0.1"),
  file.path(package_path, "DESCRIPTION"))
use_rzig(package_path)
bindings <- render_bindings(scan_exports(package_path))
cat(bindings$wrappers)
cat(bindings$namespace)
unlink(package_path, recursive = TRUE)
```

rzig_status

Report the state of a Zig-backed package

Description

Checks each step of the RZig workflow in a package and reports what is present, what is missing and what would make the installation fail:

Usage

```
rzig_status(path, build = FALSE)
```

Arguments

path	Path to the package root.
build	If ‘TRUE’, also compile and install the package with [build_zig()] when every other check passes.

Details

* **DESCRIPTION**: the package name and a valid version. * **Scaffold**: the files written by [use_rzig()], framework sources that match the installed rzig, and executable ‘configure’ and ‘cleanup’ scripts. * **Zig source**: ‘src/rzig/src/main.zig’ with the registration block and the panic handler of rzig. * **Exports**: the exported functions found by [scan_exports()], with every argument and return type that the boundary rejects, more than 32 arguments, misuse of ‘rzig.Mut([]f64)’,

error sets other than ‘`rzig.Error`’, and R functions of the same name elsewhere in ‘`R/`’. * **Bindings**: whether the files written by `[document()]` match the exports. * **Zig compiler**: a compiler of a supported release series (`[find_zig()]`). * **C toolchain**: the C compiler of R, ‘`make`’ and the R headers. * **Build** (only with ‘`build = TRUE`’): the result of `[build_zig()]`.

Every check except the build runs in R without compiling anything.

Value

An object of class ‘`rzig_status`’: a data frame with the columns ‘`step`’, ‘`ok`’ and ‘`detail`’, one row per check, with the attributes ‘`path`’, ‘`package`’ and, with ‘`build = TRUE`’, ‘`build`’ (the `[rzig_build][build_zig()]` object).

Examples

```
package_path <- tempfile("rzig-status-", tmpdir = tempdir())
dir.create(package_path)
writeLines(c("Package: examplepkg", "Version: 0.0.1"),
  file.path(package_path, "DESCRIPTION"))
use_rzig(package_path)
rzig_status(package_path)
main <- file.path(package_path, "src", "rzig", "src", "main.zig")
cat("/// @export\npub fn half(x: f32) f32 {\n    return x / 2;\n}\n",
  file = main, append = TRUE)
rzig_status(package_path)
unlink(package_path, recursive = TRUE)
```

scan_exports

Scan the Zig source of a package for exported functions

Description

Reads ‘`src/rzig/src/main.zig`’ and lists every public top-level Zig function whose documentation comment contains a line ‘`/// @export`’. The scan is pure R and needs no Zig compiler; the Zig compiler checks the parameter and return types when the package is installed.

Usage

```
scan_exports(path)
```

Arguments

`path` Path to a package previously initialized with `[use_rzig()]`.

Details

A function is found when its declaration starts at the beginning of a line with ‘`pub fn`’ and the lines directly above it are ‘`///`’ documentation comments (blank lines and ordinary ‘`///`’ comments between the comment block and the declaration are ignored). The first parameter is omitted from the R signature when its type is ‘`*rzig.Ctx`’.

Value

An object of class ‘rzig_exports’: a list with one element per exported function, each a list with ‘name’ (the R function name), ‘identifier’ (the Zig identifier as written), ‘parameters’ (the R argument names), ‘parameter_types’ (the Zig types of the R arguments), ‘return_type’ (the Zig return type as written), ‘doc’ (the documentation comment without the ‘@export’ line) and ‘line’ (the line of the declaration). The attributes ‘package’, ‘path’ and ‘source’ record the package name, its root and the scanned file. [as.data.frame()] turns the object into a table with one row per function.

Examples

```
package_path <- tempfile("rzig-scan-", tmpdir = tempdir())
dir.create(package_path)
writeLines(c("Package: examplepkg", "Version: 0.0.1"),
  file.path(package_path, "DESCRIPTION"))
use_rzig(package_path)
exports <- scan_exports(package_path)
exports
as.data.frame(exports)
unlink(package_path, recursive = TRUE)
```

use_rzig

Add RZig to a package

Description

Creates the Zig source tree, the portable build files, the native entry stub and a working ‘hello_zig()’ example in an existing package, then generates the bindings with [document()]. The function is pure R; a Zig compiler of the supported release series (see [find_zig()]) is needed only when the package is installed.

Usage

```
use_rzig(path, overwrite = FALSE)
```

Arguments

path	Path to the package root; the directory must contain a ‘DESCRIPTION’ file.
overwrite	Replace files previously managed by RZig. The package author’s Zig sources in ‘src/rzig/src’ are kept, so ‘use_rzig(path, overwrite = TRUE)’ updates the framework sources and build files of a package to the installed version of rzig.

Details

The files written below the package root are: ‘configure’, ‘configure.win’, ‘cleanup’, ‘cleanup.win’, ‘src/entry.c’, ‘src/Makevars.in’, ‘src/Makevars.win.in’, the framework sources under ‘src/rzig/framework’, the package author’s Zig source ‘src/rzig/src/main.zig’, the generated ‘R/rzig-wrappers.R’, and a managed block in ‘NAMESPACE’. Entries for the Zig build caches are added to ‘.Rbuildignore’.

Value

An object of class 'rzig_scaffold': a list with 'path' (the normalized package root), 'package', 'files' (the paths written, relative to the root) and 'exports' (the [rzig_exports][scan_exports()] found in 'main.zig'), invisibly.

Examples

```
package_path <- tempfile("rzig-example-", tmpdir = tempdir())
dir.create(package_path)
writeLines(c("Package: examplepkg", "Version: 0.0.1"),
  file.path(package_path, "DESCRIPTION"))
scaffold <- use_rzig(package_path)
scaffold
file.exists(file.path(package_path, "src", "rzig", "src", "main.zig"))
unlink(package_path, recursive = TRUE)
```

Index

`build_zig`, [2](#), [2](#)

`document`, [2](#), [4](#)

`find_zig`, [2](#), [5](#)

`render_bindings`, [2](#), [5](#)

`rzig (rzig-package)`, [2](#)

`rzig-package`, [2](#)

`rzig_status`, [2](#), [6](#)

`scan_exports`, [2](#), [7](#)

`use_rzig`, [2](#), [8](#)