

# Package ‘cleancall’

May 8, 2026

**Title** C Resource Cleanup via Exit Handlers

**Version** 0.1.4

**Description** Wrapper of .Call() that runs exit handlers to clean up C resources. Helps managing C (non-R) resources while using the R API.

**License** MIT + file LICENSE

**URL** <https://r-lib.github.io/cleancall/>,  
<https://github.com/r-lib/cleancall>

**BugReports** <https://github.com/r-lib/cleancall/issues>

**Depends** R (>= 3.6)

**Suggests** covr, testthat (>= 3.0.0)

**Config/Needs/website** tidyverse/tidytemplate

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

**Author** Lionel Henry [aut],  
Gábor Csárdi [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-7098-9676>>),  
Posit Software, PBC [cph, fnd]

**Maintainer** Gábor Csárdi <[csardi.gabor@gmail.com](mailto:csardi.gabor@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-04-08 07:50:02 UTC

## Contents

cleancall-package . . . . .	2
call_with_cleanup . . . . .	2
<b>Index</b>	<b>4</b>

---

cleancall-package      *cleancall: C Resource Cleanup via Exit Handlers*

---

### Description

Wrapper of `.Call()` that runs exit handlers to clean up C resources. Helps managing C (non-R) resources while using the R API.

### Author(s)

**Maintainer:** Gábor Csárdi <csardi.gabor@gmail.com> ([ORCID](#))

Authors:

- Lionel Henry <lionel@posit.co>

Other contributors:

- Posit Software, PBC [copyright holder, funder]

### See Also

Useful links:

- <https://github.com/r-lib/cleancall#readme>
- <https://r-lib.github.io/cleancall/>
- Report bugs at <https://github.com/r-lib/cleancall/issues>

---

call\_with\_cleanup      *Call a native routine within an exit context*

---

### Description

C functions called this way can call the `r_call_on_exit()` and/or `r_call_on_early_exit()` functions to establish exit handlers.

### Usage

```
call_with_cleanup(ptr, ...)
```

### Arguments

<code>ptr</code>	A native pointer object.
<code>...</code>	Arguments for the native routine. Handlers installed via <code>r_call_on_exit()</code> are always executed on exit. Handlers installed via <code>r_call_on_early_exit()</code> are only executed on early exit, i.e. <i>not</i> on normal termination.

**C API**

- `void r_call_on_exit(void (*fn)(void* data), void *data)`

Push an exit handler to the stack. This exit handler is always executed, i.e. both on normal and early exits.

Exit handlers are executed right after the function called from `call_with_cleanup()` exits. (Or the function used in `r_with_cleanup_context()`, if the cleanup context was established from C.)

Exit handlers are executed in reverse order (last in is first out, LIFO). Exit handlers pushed with `r_call_on_exit()` and `r_call_on_early_exit()` share the same stack.

Best practice is to use this function immediately after acquiring a resource, with the appropriate cleanup function for that resource.
- `void r_call_on_early_exit(void (*fn)(void* data), void *data)`

Push an exit handler to the stack. This exit handler is only executed on early exits, *not* on normal termination.

Exit handlers are executed right after the function called from `call_with_cleanup()` exits. (Or the function used in `r_with_cleanup_context()`, if the cleanup context was established from C.)

Exit handlers are executed in reverse order (last in is first out, LIFO). Exit handlers pushed with `r_call_on_exit()` and `r_call_on_early_exit()` share the same stack.

Best practice is to use this function immediately after acquiring a resource, with the appropriate cleanup function for that resource.
- `SEXP r_with_cleanup_context(SEXP (*fn)(void* data), void* data)`

Establish a cleanup stack and call `fn` with `data`. This function can be used to establish a cleanup stack from C code.

**See Also**

The package README file.

# Index

[call\\_with\\_cleanup, 2](#)  
[cleancall \(cleancall-package\), 2](#)  
[cleancall-package, 2](#)