Package 'RDesk'

April 22, 2026

Title Native Desktop App Framework for 'R'

Version 1.0.5

Description Build native Windows desktop applications using 'R' and 'WebView2'. Provides a robust 'R6'-based event loop, asynchronous background task management via 'mirai' and 'callr', and a native 'Win32' message bridge for seamless 'R'-to-user-interface communication without listening ports or network overhead. Allows 'R' developers to create professional, standalone desktop tools with modern web-based user interfaces while maintaining a pure 'R' backend.

OS_type windows

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL https://github.com/Janakiraman-311/RDesk, https://janakiraman-311.github.io/RDesk/

BugReports https://github.com/Janakiraman-311/RDesk/issues

SystemRequirements Windows 10 or later, 'Rtools44' or later (https://cran.r-project.org/bin/windows/Rtools/), Microsoft 'WebView2' Runtime (https://developer.microsoft.com/microsoft-edge/webview2/)

RoxygenNote 7.3.3

Imports R6, jsonlite, digest, processx, callr, mirai (>= 1.0.0), base64enc, zip, stats, utils, parallel

Suggests testthat (>= 3.0.0), withr, fs, knitr, rmarkdown, devtools, renv, broom, ggplot2, dplyr, rstudioapi, pkgdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Janakiraman G [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6050-2318>),
Serge Zaitsev [cph] (Author of webview library (src/webview/webview.h)),
Niels Lohmann [cph] (Author of nlohmann/json
(inst/include/nlohmann/json.hpp)),
Microsoft Corporation [cph] (WebView2 SDK (src/webview2_sdk/))

Maintainer Janakiraman G <janakiraman.bt@gmail.com>

Repository CRAN

Date/Publication 2026-04-22 13:00:02 UTC

Contents

App 2
async 10
build_app 11
rdesk_async 13
rdesk_auto_update 14
rdesk_cancel_job 15
rdesk_create_app 15
rdesk_df_to_list 16
rdesk_error_plot 17
rdesk_is_bundle 17
rdesk_jobs_list 18
rdesk_jobs_pending 18
rdesk_message 18
rdesk_parse_message 19
rdesk_plot_to_base64 19
rdesk_service 20

Index **21**

App *Create and launch a native desktop application window from R.*

Description

Provides bidirectional native pipe communication between R and the UI.

Methods

Public methods:

- `App$new()`
- `App$on_ready()`
- `App$on_close()`
- `App$check_update()`

- App$register_hotkey()
- App$set_tray_menu()
- App$clipboard_write()
- App$clipboard_read()
- App$on_message()
- App$send()
- App$load_ui()
- App$set_size()
- App$set_position()
- App$set_title()
- App$minimize()
- App$maximize()
- App$restore()
- App$fullscreen()
- App$always_on_top()
- App$set_menu()
- App$dialog_open()
- App$dialog_save()
- App$dialog_folder()
- App$message_box()
- App$dialog_color()
- App$notify()
- App$loading_start()
- App$loading_progress()
- App$loading_done()
- App$toast()
- App$set_tray()
- App$remove_tray()
- App$service()
- App$quit()
- App$get_dir()
- App$run()
- App$clone()

Method `new()`**:** Create a new RDesk application

Usage:

```
App$new(title, width = 1200L, height = 800L, www = NULL, icon = NULL)
```

Arguments:

`title` Window title string

`width` Window width in pixels (default 1200)

`height` Window height in pixels (default 800)

`www` Directory containing HTML/CSS/JS assets (default: built-in template)

`icon` Path to window icon file

Returns: A new App instance

Method `on_ready()`**:** Register a callback to fire when the window is ready

Usage:

```
App$on_ready(fn)
```

Arguments:

`fn` A zero-argument function called after the server starts and window opens

Returns: The App instance (invisible)

Method `on_close()`**:** Register a callback to fire when the user attempts to close the window

Usage:

```
App$on_close(fn)
```

Arguments:

`fn` A zero-argument function. Should return TRUE to allow closing, FALSE to cancel.

Returns: The App instance (invisible)

Method `check_update()`**:** Check for application updates from a remote URL

Usage:

```
App$check_update(version_url, current_version = NULL)
```

Arguments:

`version_url` URL to a JSON metadata file (e.g. `{"version": "1.1.0", "url": "http://..."}`)

`current_version` Optional version string to compare against. Defaults to app description version.

Returns: A list with update status and metadata

Method `register_hotkey()`**:** Register a global keyboard shortcut (hotkey)

Usage:

```
App$register_hotkey(keys, fn)
```

Arguments:

`keys` Character string representing the key combination (e.g., "Ctrl+Shift+A")

`fn` A zero-argument function to be called when the hotkey is pressed

Returns: The App instance (invisible)

Method `set_tray_menu()`**:** Set the native system tray menu

Usage:

```
App$set_tray_menu(items)
```

Arguments:

`items` A named list of lists defining the menu structure

Returns: The App instance (invisible)

Method `clipboard_write()`**:** Write text to the system clipboard

Usage:

```
App$clipboard_write(text)
```

Arguments:

`text` Character string to copy

Returns: The App instance (invisible)

Method `clipboard_read()`**:** Read text from the system clipboard

Usage:

```
App$clipboard_read()
```

Returns: Character string from clipboard or NULL

Method `on_message()`**:** Register a handler for a UI -> R message type

Usage:

```
App$on_message(type, fn)
```

Arguments:

`type` Unique message identifier string
`fn` A function(payload) called when this message type arrives

Returns: The App instance (invisible)

Method `send()`**:** Send a message from R to the UI

Usage:

```
App$send(type, payload = list())
```

Arguments:

`type` Character string message type (received by rdesk.on() in JS)
`payload` A list or data.frame to serialise as JSON payload

Returns: The App instance (invisible)

Method `load_ui()`**:** Load an HTML file into the window

Usage:

```
App$load_ui(path = "index.html")
```

Arguments:

`path` Path relative to the www directory (e.g. "index.html")

Returns: The App instance (invisible)

Method `set_size()`**:** Set the window size dynamically

Usage:

```
App$set_size(width, height)
```

Arguments:

`width` New width (pixels)
`height` New height (pixels)

Returns: The App instance (invisible)

Method `set_position()`**:** Set the window position dynamically

Usage:

```
App$set_position(x, y)
```

Arguments:

`x` Horizontal position from left (pixels)

`y` Vertical position from top (pixels)

Returns: The App instance (invisible)

Method `set_title()`**:** Set the window title dynamically

Usage:

```
App$set_title(title)
```

Arguments:

`title` New title

Returns: The App instance (invisible)

Method `minimize()`**:** Minimize the window to the taskbar

Usage:

```
App$minimize()
```

Returns: The App instance (invisible)

Method `maximize()`**:** Maximize the window to fill the screen

Usage:

```
App$maximize()
```

Returns: The App instance (invisible)

Method `restore()`**:** Restore the window from minimize/maximize

Usage:

```
App$restore()
```

Returns: The App instance (invisible)

Method `fullscreen()`**:** Toggle fullscreen mode

Usage:

```
App$fullscreen(enabled = TRUE)
```

Arguments:

`enabled` If TRUE, enters fullscreen. If FALSE, exits.

Returns: The App instance (invisible)

Method `always_on_top()`**:** Set the window to stay always on top of others

Usage:

```
App$always_on_top(enabled = TRUE)
```

Arguments:

`enabled` If TRUE, always on top.

Returns: The App instance (invisible)

Method `set_menu()`**:** Set the native window menu

Usage:

```
App$set_menu(items)
```

Arguments:

`items` A named list of lists defining the menu structure

Returns: The App instance (invisible)

Method `dialog_open()`**:** Open a native file-open dialog

Usage:

```
App$dialog_open(title = "Open File", filters = NULL)
```

Arguments:

`title` Dialog title

`filters` List of file filters, e.g. list("CSV files" = "*.csv")

Returns: Selected file path (character) or NULL if cancelled

Method `dialog_save()`**:** Open a native file-save dialog

Usage:

```
App$dialog_save(title = "Save File", default_name = "", filters = NULL)
```

Arguments:

`title` Dialog title

`default_name` Initial filename

`filters` List of file filters

Returns: Selected file path (character) or NULL if cancelled

Method `dialog_folder()`**:** Open a native folder selection dialog

Usage:

```
App$dialog_folder(title = "Select Folder")
```

Arguments:

`title` Dialog title

Returns: Selected directory path (character) or NULL if cancelled

Method `message_box()`**:** Show a native message box / alert

Usage:

```
App$message_box(message, title = "RDesk", type = "ok", icon = "info")
```

Arguments:

`message` The message text

`title` The dialog title

`type` One of "ok", "okcancel", "yesno", "yesnocancel"

`icon` One of "info", "warning", "error", "question"

Returns: The button pressed (character: "ok", "cancel", "yes", "no")

Method `dialog_color()`**:** Open a native color selection dialog

Usage:

```
App$dialog_color(initial_color = "#FFFFFF")
```

Arguments:

`initial_color` Optional hex color to start with (e.g. "#FF0000")

Returns: Selected hex color code or NULL if cancelled

Method `notify()`**:** Send a native desktop notification

Usage:

```
App$notify(title, body = "")
```

Arguments:

`title` Notification title

`body` Notification body text

Returns: The App instance (invisible)

Method `loading_start()`**:** Show a loading state in the UI

Usage:

```
App$loading_start(
  message = "Loading...",
  progress = NULL,
  cancellable = FALSE,
  job_id = NULL
)
```

Arguments:

`message` Text shown under the spinner

`progress` Optional numeric 0-100 for a progress bar

`cancellable` If TRUE, shows a cancel button in the UI

`job_id` Optional job_id from rdesk_async() to wire cancel button

Method `loading_progress()`**:** Update progress on an active loading state

Usage:

```
App$loading_progress(value, message = NULL)
```

Arguments:

`value` Numeric 0-100

`message` Optional updated message

Method `loading_done()`**:** Hide the loading state in the UI

Usage:

```
App$loading_done()
```

Method `toast()`**:** Show a non-blocking toast notification in the UI

Usage:

```
App$toast(message, type = "info", duration_ms = 3000L)
```

Arguments:

`message` Text to show

`type` One of "info", "success", "warning", "error"

`duration_ms` How long to show it (default 3000ms)

Method `set_tray()`**:** Set or update the system tray icon

Usage:

```
App$set_tray(label = "RDesk App", icon = NULL, on_click = NULL)
```

Arguments:

`label` Tooltip text for the tray icon

`icon` Path to .ico file (optional)

`on_click` Character "left" or "right" or callback function(button)

Returns: The App instance (invisible)

Method `remove_tray()`**:** Remove the system tray icon

Usage:

```
App$remove_tray()
```

Returns: The App instance (invisible)

Method `service()`**:** Service this app's pending native events

Usage:

```
App$service()
```

Returns: The App instance (invisible)

Method `quit()`**:** Close the window and stop the app's event loop.

Usage:

```
App$quit()
```

Returns: The App instance (invisible)

Method `get_dir()`**:** Get the application root directory (where www/ and R/ are located).

Usage:

```
App$get_dir()
```

Returns: Character string path.

Method `run()`**:** Start the application - opens the window

Usage:

```
App$run(block = TRUE)
```

Arguments:

`block` If TRUE (default), blocks with an event loop until the window is closed.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
App$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Safe logical check (unwrapped)
app_dir <- system.file("templates/hello", package = "RDesk")
if (nzchar(app_dir)) {
  message("Built-in app directory: ", app_dir)
}

if (interactive()) {
  app <- App$new(title = "Car Visualizer", width = 1200, height = 800)

  app$on_ready(function() {
    message("App is ready!")
  })

  # Handle messages from UI
  app$on_message("get_data", function(payload) {
    list(cars = mtcars[1:5, ])
  })

  # Start the app
  app$run()
}
```

---

async *Wrap a message handler to run asynchronously with zero configuration*

Description

`async()` is the simplest way to make an RDesk message handler non-blocking. Wrap any handler function with `async()` and RDesk automatically handles background execution, loading states, error toasts, and result routing.

`async()` transforms a standard RDesk message handler into a background task. The UI remains responsive while the task runs. When finished, a result message (e.g., `get_data_result`) is automatically sent back to the UI.

Usage

```
async(
  fn,
  app = NULL,
  loading_message = "Working...",
  cancellable = TRUE,
  error_message = "Error: "
)
```

Arguments

fn	The handler function, taking a payload argument.
app	The RDesk App instance. If NULL, tries to resolve from the global registry.
loading_message	Message to display in the UI overlay while working.
cancellable	Whether the UI should show a 'Cancel' button.
error_message	Prefix for toast notifications if the task fails.

Details

To ensure the background worker has access to all application logic, RDesk automatically sources every .R file in the application's R/ directory before executing the task. It also snapshots currently loaded packages (excluding system packages) to recreate the environment.

Value

A wrapped handler function suitable for app$on_message().

Examples

```
if (interactive()) {
  app$on_message("filter_cars", async(function(payload) {
    mtcars[mtcars$cyl == payload$cylinders, ]
  }, app = app))
}
```

---

build_app *Build a self-contained distributable from an RDesk application*

Description

Build a self-contained distributable from an RDesk application

Usage

```
build_app(
  app_dir = ".",
  out_dir = file.path(tempdir(), "dist"),
  app_name = NULL,
  version = NULL,
  r_version = NULL,
  include_packages = character(0),
  portable_r_method = c("extract_only", "installer"),
  runtime_dir = NULL,
  overwrite = FALSE,
  build_installer = FALSE,
```

```
  publisher = "RDesk User",
  website = "https://github.com/Janakiraman-311/RDesk",
  license_file = NULL,
  icon_file = NULL,
  prune_runtime = TRUE,
  dry_run = FALSE
)
```

Arguments

app_dir	Path to the app directory (must contain app.R and www/)
out_dir	Output directory for the zip file (created if not exists)
app_name	Name of the application. Defaults to name in DESCRIPTION or "MyRDeskApp".
version	Version string. Defaults to version in DESCRIPTION or "1.0.0".
r_version	R version to bundle e.g. "4.4.2". Defaults to current R version.
include_packages	Character vector of extra CRAN packages to bundle. RDesk's own dependencies are always included automatically.
portable_r_method	How to provision the bundled R runtime when `runtime_dir = "download"` is used explicitly. `"extract_only"` requires standalone 7-Zip and never launches the R installer. `"installer"` allows the legacy silent installer path explicitly.
runtime_dir	Controls which R runtime is bundled with the app. • `NULL` (default) – copies the developer's currently running R installation. Guarantees the bundled packages and the runtime are the same R version, eliminating renv version-mismatch crashes. • A filesystem path – copies that R installation root directly. • `"download"` – downloads a portable R installer from CRAN (legacy behaviour; requires network; risks version mismatch with renv).
overwrite	If TRUE, overwrite existing output. Default FALSE.
build_installer	If TRUE, also build a Windows installer (.exe) using InnoSetup.
publisher	Documentation for the application publisher (used in installer).
website	URL for the application website (used in installer).
license_file	Path to a license file (.txt or .rtf) to include in the installer.
icon_file	Path to an .ico file for the installer and application shortcut.
prune_runtime	If TRUE, remove unnecessary files (Tcl/Tk, docs, tests) from the bundled R runtime to reduce size (~15-20MB saving). Default TRUE.
dry_run	If TRUE, performs a quick validation of the app structure and environment without performing the full build. Default FALSE.

Value

Path to the created zip file, invisibly.

Examples

```
# Prepare an app directory (following scaffold example)
app_path <- file.path(tempdir(), "MyApp")
rdesk_create_app("MyApp", path = tempdir())

# Perform a dry-run build (fast, no external binaries downloaded)
build_app(app_path, out_dir = tempdir(), dry_run = TRUE)
```

`rdesk_async` *Run a task in the background*

Description

Automatically switches between 'mirai' (persistent daemons) and 'callr' (on-demand processes).

Usage

```
rdesk_async(
  task,
  args = list(),
  on_done = NULL,
  on_error = NULL,
  timeout_sec = NULL,
  app_id = NULL
)
```

Arguments

`task`	A function to run in the background.
`args`	A list of arguments to pass to the task.
`on_done`	Callback function(result) called when the task finishes successfully.
`on_error`	Callback function(error) called if the task fails.
`timeout_sec`	Optional timeout in seconds. If exceeded, the job is cancelled and `on_error()` receives a timeout error.
`app_id`	Optional App ID used to associate a job with a specific app.

Value

Invisible job ID.

Examples

```
# Fast, non-interactive task check (safe to unwrap)
rdesk_jobs_pending()

if (interactive()) {
  # Run a long-running computation in the background
  rdesk_async(
    task = function(n) { Sys.sleep(2); sum(runif(n)) },
    args = list(n = 1e6),
    on_done = function(res) message("Task finished: ", res),
    on_error = function(err) message("Task failed: ", err$message)
  )
}
```

`rdesk_auto_update` *Automatically check for and install app updates*

Description

`rdesk_auto_update` is a high-level function designed for bundled (standalone) applications. It checks a remote version string, compares it with the current version, and if a newer version is found, it downloads and executes the installer silently before quitting the current application.

Usage

```
rdesk_auto_update(
  version_url,
  download_url,
  current_version,
  silent = FALSE,
  app = NULL
)
```

Arguments

`version_url`	URL to a plain text file containing the latest version string (e.g., "1.1.0")
`download_url`	URL to the latest installer .exe
`current_version`	Current app version string e.g. "1.0.0"
`silent`	If TRUE, downloads and installs without prompting. Default FALSE.
`app`	Optional App instance for showing toast notifications.

Value

Invisible TRUE if update was applied, FALSE otherwise.

rdesk_cancel_job *Cancel a running background job*

Description

Cancel a running background job

Usage

```
rdesk_cancel_job(job_id)
```

Arguments

job_id	The ID of the job to cancel.

Value

Invisible TRUE if cancelled, FALSE if not found.

rdesk_create_app *Create a new RDesk application*

Description

Scaffolds a professional RDesk application with a modern dashboard layout. The app includes a sidebar for filters, KPI cards, and an asynchronous ggplot2 charting engine fueled by mtcars (default).

Usage

```
rdesk_create_app(
  name,
  path = tempdir(),
  data_source = NULL,
  viz_type = NULL,
  use_async = NULL,
  theme = "light",
  open = TRUE
)
```

Arguments

name	App name. Must be a valid directory name.
path	Directory to create the app in. Default is current directory.
data_source	Internal use. Defaults to "builtin".
viz_type	Internal use. Defaults to "mixed".
use_async	Internal use. Defaults to TRUE.
theme	One of "light", "dark", "system". Default "system".
open	Logical. If TRUE and in RStudio, opens the new project in a new session.

Value

Path to the created app directory, invisibly.

Examples

```
if (interactive()) {
  # Create the Professional Hero Dashboard in a temporary directory
  rdesk_create_app("MyDashboard", path = tempdir())
}

# The following demonstrates just the return value without opening a window
# (Fast and safe - no \dontrun needed for this specific logical check)
path <- file.path(tempdir(), "TestLogic")
if (!dir.exists(path)) {
  # This is just a placeholder example of how to call the function safely
  message("Scaffold path will be: ", path)
}
```

---

rdesk_df_to_list *Convert a data frame to a list suitable for JSON serialization*

Description

Convert a data frame to a list suitable for JSON serialization

Usage

```
rdesk_df_to_list(df)
```

Arguments

df	Data frame to convert

Value

A list with 'rows' (list of lists) and 'cols' (character vector)

rdesk_error_plot *Generate a base64-encoded error plot*

Description

Generate a base64-encoded error plot

Usage

```
rdesk_error_plot(message = "Error generating plot")
```

Arguments

message	Error message to display (optional)

Value

A base64-encoded PNG string

rdesk_is_bundle *Check if the app is running in a bundled (standalone) environment*

Description

Check if the app is running in a bundled (standalone) environment

Usage

```
rdesk_is_bundle()
```

Value

TRUE if running in a bundle, FALSE otherwise

rdesk_jobs_list *List currently pending background jobs*

Description

List currently pending background jobs

Usage

```
rdesk_jobs_list()
```

Value

A data.frame with job ID, started time, backend, and app ID.

rdesk_jobs_pending *Check if any background jobs are pending*

Description

Check if any background jobs are pending

Usage

```
rdesk_jobs_pending()
```

Value

Number of pending jobs.

rdesk_message *Construct a standard RDesk IPC message envelope*

Description

Construct a standard RDesk IPC message envelope

Usage

```
rdesk_message(
  type,
  payload = list(),
  version = getOption("rdesk.ipc_version", "1.0")
)
```

Arguments

type	The message type/action name
payload	A list representing the message data
version	The contract version (default "1.0")

Value

A list representing the standard JSON envelope

---

rdesk_parse_message *Parse and validate an incoming RDesk IPC message*

Description

Parse and validate an incoming RDesk IPC message

Usage

```
rdesk_parse_message(raw_json)
```

Arguments

raw_json	The raw JSON string from the frontend

Value

A list containing the validated message components, or NULL if invalid

---

rdesk_plot_to_base64 *Convert a ggplot2 object to a base64-encoded PNG string*

Description

Convert a ggplot2 object to a base64-encoded PNG string

Usage

```
rdesk_plot_to_base64(plot, width = 6, height = 4, dpi = 96)
```

Arguments

plot	A ggplot2 object
width	Width in inches (default 6)
height	Height in inches (default 4)
dpi	DPI resolution (default 96)

Value

A base64-encoded PNG string or a fallback error plot

`rdesk_service` *Service all active RDesk applications*

Description

Processes native OS events for all open windows. Call this periodically if you are running apps with `block = FALSE`.

Usage

```
rdesk_service()
```

Index

App, 2
async, 10

build_app, 11

rdesk_async, 13
rdesk_auto_update, 14
rdesk_cancel_job, 15
rdesk_create_app, 15
rdesk_df_to_list, 16
rdesk_error_plot, 17
rdesk_is_bundle, 17
rdesk_jobs_list, 18
rdesk_jobs_pending, 18
rdesk_message, 18
rdesk_parse_message, 19
rdesk_plot_to_base64, 19
rdesk_service, 20