

Package: biotrace (via r-universe)

September 27, 2026

Type Package

Title R Integration for the BioTrace GitHub Action

Version 0.1.0

Date 2026-08-20

Description Tools for configuring, validating, running and inspecting integrations with the BioTrace GitHub Action, which traces biological results from data and code to figures and scientific claims. The package provides an R API, a command-line interface, GitHub workflow scaffolding, configuration helpers and report readers. It does not ship or execute the upstream TypeScript engine; BioTrace runs only through its published GitHub Action (uses: rotsl/biotrace@v1).

URL <https://rotsl.r-universe.dev/biotrace>,
<https://github.com/rotsl/biotrace-r>,
<https://github.com/rotsl/biotrace>

BugReports <https://github.com/rotsl/biotrace-r/issues>

License Apache License (>= 2) | file LICENSE

Encoding UTF-8

Depends R (>= 3.6)

Imports fs, jsonlite, rlang, yaml

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

VignetteBuilder knitr

SystemRequirements Rscript on PATH for the installed CLI; no Node.js or TypeScript required to install or use this package.

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libuv1-dev

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

Date/Publication 2026-08-28 07:26:45 UTC
RemoteUrl https://github.com/rotsl/biotrace-r
RemoteRef HEAD
RemoteSha ab494e1759193e3522b4585f47ccc4bc7fbb65f5

Contents

biotrace_config	2
biotrace_report	3
biotrace_version	4
biotrace_workflow	5
run_biotrace_github	7

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

biotrace_config	<i>BioTrace configuration helpers</i>
-----------------	---------------------------------------

Description

write_biotrace_config() writes a BioTrace YAML configuration file. read_biotrace_config() reads one back as an R list. validate_biotrace_config() validates a file against the upstream JSON Schema.

Usage

```
default_biotrace_config()

default_biotrace_config_text()

write_biotrace_config(
  path,
  config = default_biotrace_config(),
  overwrite = FALSE
)

read_biotrace_config(file)

validate_biotrace_config(file, return_errors = FALSE)
```

Arguments

path [character(1)] Path to write or read.
config [list] Configuration object. Must be a list whose version field equals 1L. Use [default_biotrace_config\(\)](#) to get a sensible starting point.

overwrite	[logical(1)] If FALSE (the default), refuse to overwrite an existing file. Set to TRUE to overwrite.
file	[character(1)] Path to a BioTrace configuration file (alias for path, for read_biotrace_config()).
return_errors	[logical(1)] If TRUE, return a list of validation errors instead of throwing. If FALSE (the default), throw on the first validation failure.

Value

- `write_biotrace_config()`: the absolute path to the written file, invisibly.
- `read_biotrace_config()`: a list representing the parsed YAML.
- `validate_biotrace_config()`: TRUE (invisible) when valid and `return_errors = FALSE`; otherwise a list of error objects (possibly empty).

A list representing the smallest valid BioTrace config.

The default BioTrace configuration as a YAML string.

Stability

Configuration follows version: 1 of the BioTrace configuration schema. The schema reference copy shipped in this package is a verbatim copy of the JSON Schema published by [rotsl/biotrace](#).

Examples

```
tmp <- tempfile()
write_biotrace_config(tmp, default_biotrace_config())
validate_biotrace_config(tmp)
read_biotrace_config(tmp)
```

biotrace_report	<i>BioTrace report readers</i>
-----------------	--------------------------------

Description

`read_biotrace_report()` reads the machine-readable JSON report produced by the BioTrace GitHub Action. The result is an S3 object of class `biotrace_report` with `print()` and `summary()` methods.

Usage

```
read_biotrace_report(file)

validate_biotrace_report(file, return_errors = FALSE)

## S3 method for class 'biotrace_report'
summary(object, ...)

## S3 method for class 'biotrace_report'
print(x, ...)
```

Arguments

file	[character(1)] Path to a BioTrace report JSON file.
return_errors	[logical(1)] Return a list of validation errors instead of throwing? Used by <code>validate_biotrace_report()</code> only.
object	[biotrace_report] A report object.
...	Unused. Present so the S3 methods match the generic signature.
x	[biotrace_report] A report object.

Details

`validate_biotrace_report()` validates a report file against the upstream BioTrace report JSON Schema.

`summary.biotrace_report()` returns a compact summary of the report: status, score, finding counts.

Value

- `read_biotrace_report()`: an object of class `biotrace_report` (a list with extra attributes holding the source path).
- `validate_biotrace_report()`: TRUE (invisible) or a list of error objects.
- `summary.biotrace_report()`: a list with elements `status`, `score`, `findings_count`, `blocking_findings_count`, `files_checked`, `claims_checked`, `figures_checked`, `generated_at`, `source`.

Report shape

The report JSON follows the upstream `biotrace-report.schema.json` (shipped under `inst/extdata/schemas/`). Required fields are `schema_version`, `generated_at`, `status`, `score`, `summary`, `findings`.

Examples

```
report_path <- system.file("extdata", "example-report.json",
                           package = "biotrace")
report <- read_biotrace_report(report_path)
print(report)
summary(report)
```

biotrace_version

Report the package version and the upstream BioTrace reference

Description

`biotrace_version()` returns a list with the package version and the upstream BioTrace Action tag this package was designed against. The upstream tag is informational: this package works with the upstream Action at the uses: `rots1/biotrace@v1` ref, which is the tag this package targets.

Usage

```
biotrace_version()
```

Value

A list with elements `package` (character version string), `upstream_repo` (URL), `upstream_ref` (Action tag), and `upstream_action_version` (the package.json version of upstream BioTrace at the time this package was authored).

Examples

```
biotrace_version()
```

<code>biotrace_workflow</code>	<i>BioTrace workflow scaffolding</i>
--------------------------------	--------------------------------------

Description

`use_biotrace()` initialises BioTrace integration in a project by writing `.github/biotrace.yml` and `.github/workflows/biotrace.yml`.

Usage

```
biotrace_workflow_text(  
  path = NULL,  
  config_path = ".github/biotrace.yml",  
  fail_on = "error",  
  ai_enabled = "false",  
  include_workflow_dispatch = FALSE,  
  upload_report = TRUE,  
  extra_inputs = list()  
)  
  
write_biotrace_workflow(  
  path,  
  config_path = ".github/biotrace.yml",  
  fail_on = "error",  
  ai_enabled = "false",  
  include_workflow_dispatch = FALSE,  
  upload_report = TRUE,  
  extra_inputs = list(),  
  overwrite = FALSE  
)  
  
use_biotrace(  
  project = NULL,  
  config_path = ".github/biotrace.yml",
```

```

workflow_path = ".github/workflows/biotrace.yml",
fail_on = "error",
ai_enabled = "false",
include_workflow_dispatch = FALSE,
upload_report = TRUE,
extra_inputs = list(),
overwrite = FALSE
)

```

Arguments

path	[character(1)] For <code>write_biotrace_workflow()</code> , the path to write. For <code>biotrace_workflow_text()</code> , ignored.
config_path	[character(1)] Path (relative to project) of the BioTrace YAML configuration file. Defaults to <code>.github/biotrace.yml</code> .
fail_on	[character(1)] Minimum severity that causes the Action to fail. One of "info", "warning", "error". Defaults to "error".
ai_enabled	[character(1)] Whether AI observations are allowed. One of "auto", "true", "false". Defaults to "false" so the workflow is fully deterministic by default.
include_workflow_dispatch	[logical(1)] Add a <code>workflow_dispatch</code> : trigger so the workflow can be run on-demand from the Actions tab. Defaults to FALSE.
upload_report	[logical(1)] Upload the BioTrace JSON report as a workflow artifact. Defaults to TRUE.
extra_inputs	[named list] Extra with: inputs to pass to <code>rotsl/biotrace@v1</code> . Each name becomes a YAML key. Use this to set <code>create-labels</code> , <code>comment-mode</code> , <code>report-path</code> , <code>ai-provider</code> , etc.
overwrite	[logical(1)] Overwrite existing files. Defaults to FALSE.
project	[character(1)] Project root. Defaults to the current working directory, or the nearest directory containing <code>.git</code> or <code>DESCRIPTION</code> .
workflow_path	[character(1)] Path (relative to project) of the GitHub workflow YAML file. Defaults to <code>.github/workflows/biotrace.yml</code> .

Details

`write_biotrace_workflow()` writes the workflow file alone.

`biotrace_workflow_text()` returns the workflow file content as a character vector (so callers can inspect or post it without writing to disk).

The generated workflow references the upstream Action directly:

```
uses: rotsl/biotrace@v1
```

It never references this package itself as an Action, because `biotrace` is an R package, not a GitHub Action repository.

Value

- `use_biotrace()`: a named list of absolute paths (config, workflow) invisibly.
- `write_biotrace_workflow()`: the absolute path, invisibly.
- `biotrace_workflow_text()`: a character vector (one element per line, no trailing newline) representing the workflow YAML.

Inputs echoed into the workflow

The generated workflow uses these inputs (matching the upstream `action.yml` of BioTrace v1.x): `config`, `github-token`, `fail-on`, `ai-enabled`. Other inputs retain their upstream defaults. Pass `extra_inputs` to add or override inputs.

Examples

```
tmp <- tempfile()
write_biotrace_workflow(tmp)
cat(biotrace_workflow_text(), sep = "\n")
```

<code>run_biotrace_github</code>	<i>Trigger or inspect BioTrace via GitHub</i>
----------------------------------	---

Description

`run_biotrace_github()` provides an R-facing way to interact with a GitHub repository that already uses the BioTrace workflow. It does not embed the upstream BioTrace engine, and it does not modify `rotsl/biotrace`.

Usage

```
run_biotrace_github(
  repo,
  action = c("list-workflows", "list-runs", "trigger"),
  workflow_name = "biotrace.yml",
  ref = NULL,
  token = NULL,
  token_env = "GITHUB_TOKEN",
  api_url = "https://api.github.com",
  transport = NULL,
  dry_run = FALSE
)
```

Arguments

repo	[character(1)] Repository slug, in owner/name form (e.g. "rotsl/biotrace-r").
action	[character(1)] What to do. One of: • "list-workflows": list workflow files in the repo. • "list-runs": list recent runs of the BioTrace workflow. • "trigger": dispatch the BioTrace workflow (workflow_dispatch). Requires repo scope.
workflow_name	[character(1)] Workflow file name (e.g. "biotrace.yml"). Used for list-runs and trigger.
ref	[character(1)] Git ref to dispatch the workflow on. Defaults to the repository's default branch when NULL.
token	[character(1)] GitHub token. Leave NULL to read from environment variables (see "Authentication").
token_env	[character(1)] Name of the environment variable holding the token. Defaults to "GITHUB_TOKEN".
api_url	[character(1)] GitHub API root. Defaults to https://api.github.com. Override for GitHub Enterprise.
transport	[function] Custom HTTP transport for testing. Takes (url, method, headers, body) and returns a list with status, headers, content (raw). The default uses utils::download.file for GETs and httr::POST-equivalent logic for POSTs, but we do not require httr as a dependency.
dry_run	[logical(1)] If TRUE, return the request that <i>would</i> be issued without sending it. Useful for the CLI's --dry-run flag and for tests.

Details

The function uses the GitHub REST API via `utils::download.file` for the read-only parts (workflow listing, workflow runs, run artifacts). For triggering (`workflow_dispatch`) it requires a token with repo scope (or `public_repo` for public repositories).

This helper is intentionally narrow: it does not attempt to re-implement GitHub Actions, and it does not claim to execute BioTrace locally. It is a convenience wrapper around the GitHub REST API for the same workflow that `use_biotrace()` writes.

Value

A list (the parsed JSON response, or the request that would have been issued when `dry_run = TRUE`). The list always contains a request element describing what was sent.

Authentication

Tokens are read from environment variables only. They are never written to disk, never embedded into generated workflows, and never printed. The lookup order is:

1. The token argument.
2. The environment variable named by the `token_env` argument (default: `GITHUB_TOKEN`).
3. The environment variable `GH_TOKEN` (a GitHub CLI convention).

If none of those is set, the function aborts with a clear error message. It does not silently degrade to unauthenticated access.

Network behaviour

All network access is opt-in. The function never touches the network during package load, during R CMD check, or during any of the local helpers. The unit tests use a mock transport (the `transport` argument) so they run completely offline.

Skip-on-CRAN

Tests that exercise the live network path are tagged with `skip_on_cran()`. The default behaviour when `transport` is NULL and no token is present is to abort cleanly.

Examples

```
## Not run:
# Requires GITHUB_TOKEN set.
run_biotrace_github("rotsl/biotrace-r", action = "list-runs")

## End(Not run)

# Safe, offline: see the request that would be issued.
run_biotrace_github(
  "rotsl/biotrace-r",
  action = "trigger",
  dry_run = TRUE
)
```

Index

biotrace_config, [2](#)
biotrace_report, [3](#)
biotrace_version, [4](#)
biotrace_workflow, [5](#)
biotrace_workflow_text
 (biotrace_workflow), [5](#)
biotrace_workflow_text(), [6](#)

default_biotrace_config
 (biotrace_config), [2](#)
default_biotrace_config(), [2](#)
default_biotrace_config_text
 (biotrace_config), [2](#)

print.biotrace_report
 (biotrace_report), [3](#)

read_biotrace_config (biotrace_config),
 [2](#)
read_biotrace_config(), [3](#)
read_biotrace_report (biotrace_report),
 [3](#)
run_biotrace_github, [7](#)

summary.biotrace_report
 (biotrace_report), [3](#)

use_biotrace (biotrace_workflow), [5](#)

validate_biotrace_config
 (biotrace_config), [2](#)
validate_biotrace_report
 (biotrace_report), [3](#)
validate_biotrace_report(), [4](#)

write_biotrace_config
 (biotrace_config), [2](#)
write_biotrace_workflow
 (biotrace_workflow), [5](#)
write_biotrace_workflow(), [6](#)