

Package ‘sentryR’

July 23, 2025

Type Package

Title Send Errors and Messages to Sentry

Version 1.1.2

Description Unofficial client for 'Sentry' <<https://sentry.io>>, a self-hosted or cloud-based error-monitoring service. It will inform about errors in real-time, and includes integration with the 'Plumber' package.

License MIT + file LICENSE

Encoding UTF-8

ByteCompile true

URL <https://github.com/jcpsantiago/sentryR>

BugReports <https://github.com/jcpsantiago/sentryR/issues>

RoxygenNote 7.2.3

Imports http, jsonlite, stringr, stats, tibble, uuid

Suggests httptest, mockery, testthat (>= 2.1.0)

NeedsCompilation no

Author Joao Santiago [aut, cre],
Daniel Kirsch [aut]

Maintainer Joao Santiago <me@jcpsantiago.xyz>

Repository CRAN

Date/Publication 2023-12-18 07:50:03 UTC

Contents

calls_to_stacktrace	2
capture	2
capture_exception	3
capture_function_calls	4
capture_message	4
configure_sentry	5
default_error_handler	6

is_sentry_configured	6
parse_dsn	7
prepare_payload	7
sentryR	8
sentry_error_handler	8
sentry_headers	9
sentry_url	9
with_captured_calls	9
wrap_error_handler_with_sentry	10

Index	11
--------------	-----------

calls_to_stacktrace	<i>Convert function call to a stack trace</i>
---------------------	---

Description

Convert function call to a stack trace

Usage

calls_to_stacktrace(calls)

Arguments

calls function calls, e.g. from sys.calls()

Value

a data.frame

capture	<i>Send a message to a Sentry server</i>
---------	--

Description

Send a message to a Sentry server

Usage

capture(...)

Arguments

... named parameters

Value

sends message to Sentry

Examples

```
## Not run:
capture(message = "oh hai there!") # send message to sentry

## End(Not run)
```

capture_exception	<i>Report an error or exception object</i>
-------------------	--

Description

Report an error or exception object

Usage

```
capture_exception(error, ..., level = "error")
```

Arguments

error	an error object
...	optional additional named parameters
level	the level of the message. Default: "error"

Value

nothing; sends error to Sentry

Examples

```
## Not run:
capture_exception(simpleError("foo"), tags = list(version = "1.0"))

## End(Not run)
```

capture_function_calls	<i>Capture function calls</i>
------------------------	-------------------------------

Description	
Capture function calls	
Usage	
capture_function_calls(error)	
Arguments	
error	error object

capture_message	<i>Report a message to Sentry</i>
-----------------	-----------------------------------

Description

Report a message to Sentry

Usage

capture_message(message, ..., level = "info")

Arguments

- message message text
- ... optional additional named parameters
- level the level of the message. Default: "info"

Value

nothing; sends message to Sentry

Examples

```
## Not run:
capture_message("this is an important message", logger = "my.logger")

## End(Not run)
```

configure_sentry	<i>Configure Sentry</i>
------------------	-------------------------

Description

Configure Sentry

Usage

```
configure_sentry(  
  dsn,  
  app_name = NULL,  
  app_version = NULL,  
  environment = NULL,  
  ...  
)
```

Arguments

dsn	the DSN of a Sentry project.
app_name	name of your application (optional). Default: NULL
app_version	version of your application (optional). Default: NULL
environment	the environment name, such as production or staging (optional). Default: NULL
...	named lists as extra parameters for the Sentry payload

Value

populates the .sentry_env environment with character strings

Examples

```
## Not run:  
configure_sentry("https://12345abcddbc45e49773bb1ca8d9c533@sentry.io/1234567")  
sentry_env$host # sentry.io  
  
## End(Not run)
```

default_error_handler *Default error handler for Plumber*

Description

Default error handler for Plumber

Usage

```
default_error_handler(req, res, error)
```

Arguments

req	a Plumber request object
res	a Plumber response object
error	an error object

Value

a list

is_sentry_configured *Check if Sentry is configured*

Description

Check if Sentry is configured

Usage

```
is_sentry_configured()
```

Value

boolean

Examples

```
## Not run:
configure_sentry("https://12345abcddbc45e49773bb1ca8d9c533@sentry.io/1234567")
is_sentry_configured() # TRUE

## End(Not run)
```

parse_dsn	<i>Parse a Sentry DSN into its components</i>
-----------	---

Description

Parse a Sentry DSN into its components

Usage

```
parse_dsn(dsn)
```

Arguments

dsn the DSN of a Sentry project.

Value

a named list with parsed elements of the DSN

Examples

```
parse_dsn("https://1234@sentry.io/1")
```

prepare_payload	<i>Prepare JSON payload for Sentry</i>
-----------------	--

Description

Prepare JSON payload for Sentry

Usage

```
prepare_payload(...)
```

Arguments

... named parameters

Value

a JSON character string

Examples

```
## Not run:
prepare_payload() # return only the core parameters
prepare_payload(tags = list(foo = 123, bar = "meh")) # add tags

## End(Not run)
```

`sentryR``sentryR` *package*

Description

SDK for 'sentry.io', a cross-platform application monitoring service

`sentry_error_handler` *Error handler with Sentry reporting*

Description

Error handler with Sentry reporting

Usage

```
sentry_error_handler(req, res, error, ...)
```

Arguments

<code>req</code>	a plumber request object
<code>res</code>	a plumber response object
<code>error</code>	an error object
<code>...</code>	extra named variables for Sentry

Value

a list with response payload

Examples

```
## Not run:
sentryR::configure_sentry(Sys.getenv("SENTRY_DSN"))
pr <- plumber::plumb("example_plumber.R")
pr$setErrorHandler(sentryR::sentry_error_handler)
pr$run()

## End(Not run)
```

sentry_headers	<i>Set the sentry.io call header</i>
----------------	--------------------------------------

Description

Set the sentry.io call header

Usage

```
sentry_headers()
```

Value

a character vector

sentry_url	<i>Build the sentry.io call URL</i>
------------	-------------------------------------

Description

Build the sentry.io call URL

Usage

```
sentry_url()
```

Value

a character string

with_captured_calls	<i>Create safe function</i>
---------------------	-----------------------------

Description

Create safe function

Usage

```
with_captured_calls(z)
```

Arguments

z	the function whose errors we want to track
---	--

Value

a function

`wrap_error_handler_with_sentry`*Wrap a plumber error handler such that it reports errors to Sentry*

Description

Wrap a plumber error handler such that it reports errors to Sentry

Usage

```
wrap_error_handler_with_sentry(error_handler = default_error_handler)
```

Arguments

`error_handler` a function to handle plumber errors

Value

a function

Index

`calls_to_stacktrace`, [2](#)
`capture`, [2](#)
`capture_exception`, [3](#)
`capture_function_calls`, [4](#)
`capture_message`, [4](#)
`configure_sentry`, [5](#)

`default_error_handler`, [6](#)

`is_sentry_configured`, [6](#)

`parse_dsn`, [7](#)
`prepare_payload`, [7](#)

`sentry_error_handler`, [8](#)
`sentry_headers`, [9](#)
`sentry_url`, [9](#)
`sentryR`, [8](#)

`with_captured_calls`, [9](#)
`wrap_error_handler_with_sentry`, [10](#)