Package ‘azlogr’

March 21, 2024

Title Logging in 'R' and Post to 'Azure Log Analytics' Workspace

Version 0.0.6

Description It extends the functionality of 'logger' package. Additional
logging metadata can be configured to be collected. Logging messages are
displayed on console and optionally they are sent to 'Azure Log Analytics'
workspace in real-time.

License MIT + file LICENSE

URL https://atalv.github.io/azlogr/ https://github.com/atalv/azlogr/

BugReports https://github.com/atalv/azlogr/issues/

Suggests covr, knitr, mockery, pkgdown, rmarkdown, testthat (>= 3.1.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

Imports caTools, digest, httr (>= 1.0.0), jsonlite, logger (>= 0.2.0)

VignetteBuilder knitr

NeedsCompilation no

Author Vivek Atal [aut, cre] (<https://orcid.org/0000-0002-9948-7458>)

Maintainer Vivek Atal <atalvivek@yahoo.co.in>

Repository CRAN

Date/Publication 2024-03-21 04:20:05 UTC

R topics documented:

.add_meta_variables *Add additional meta variable*

Description

Add additional meta variables in the logging context on top of the ones that are readily collected in `get_logger_meta_variables` function. It might be needed to add some other metadata specific to the logging usage - that goal is served by this function.

Usage

```
.add_meta_variables(
  additional_fields = NULL,
  log_level = NULL,
  namespace = NA_character_,
  .logcall = sys.call(),
  .topcall = sys.call(-1),
  .topenv = parent.frame()
)
```

Arguments

`additional_fields`	A named vector of type list with key-value pairs of additional meta data which needs to be added in logging context on top of `log_fields`. The respective value of each key is expected to be of length 1. It is `NULL` by default.
`log_level`	log level as per `log_levels`
`namespace`	string referring to the `logger` environment / config to be used to override the target of the message record to be used instead of the default namespace, which is defined by the R package name from which the logger was called, and falls back to a common, global namespace.
`.logcall`	the logging call being evaluated (useful in formatters and layouts when you want to have access to the raw, unevaluated R expression)
`.topcall`	R expression from which the logging function was called (useful in formatters and layouts to extract the calling function's name or arguments)
`.topenv`	original frame of the `.topcall` calling function where the formatter function will be evaluated and that is used to look up the `namespace` as well via `logger:::top_env_name`

Value

Returns a vector of collected meta-data. It is used in defining the `log_layout` function.

`.build_signature` *Build API signature for logging to 'Azure Log Analytics'*

Description

'Azure Log Analytics' HTTP REST API documentation for 'Python' is followed to create the 'R' version of it. 'Python' version of this function is described at `https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-collector-api?tabs=python#sample-requests/`

Usage

```
.build_signature(
  customer_id,
  shared_key,
  date,
  content_length,
  method,
  content_type,
  resource
)
```

Arguments

`customer_id`	`customer_id` of the 'Azure Log Analytics' workspace
`shared_key`	`shared_key` of the 'Azure Log Analytics' workspace
`date`	date-time of logging event
`content_length`	Content length of the body
`method`	Only one value is expected - `POST`
`content_type`	Only one value is expected - `application/json`
`resource`	Only one value is expected - `/api/logs`

Value

Returns part of the header of HTTP POST request to be sent to 'Azure Log Analytics' workspace

`.layout_json_custom` *Customized logging layout*

Description

This is an extended function of `layout_json` function from `'logger'` package. Objective is to add additional component in the logging layout in JSON format so that they can also be reported while logging along with the components collected by `.add_meta_variables`.

Usage

```
.layout_json_custom(
  log_fields = c("time", "level", "ns", "ans", "topenv", "fn", "node", "arch", "os_name",
    "os_release", "os_version", "pid", "user", "msg"),
  additional_fields = NULL,
  enforce_ascii_msg = TRUE,
  enforce_tz_utc = TRUE
)
```

Arguments

`log_fields`	Vector of components which are collected in `get_logger_meta_variables` function. Converting `time` component to UTC additionally.
`additional_fields`	A named vector of type list with key-value pairs of additional meta data which needs to be added in logging context on top of `log_fields`. The respective value of each key is expected to be of length 1. It is `NULL` by default.
`enforce_ascii_msg`	If `TRUE` (default), the logging message is guaranteed to have all non-ASCII characters escaped. If `FALSE`, the characters will be logged as-is. Please note, it is better to ensure ASCII, otherwise there might be error while sending the HTTP POST request to 'Azure Log Analytics' workspace.
`enforce_tz_utc`	If `TRUE` (default), the logging time field is converted to UTC timezone while sending the logging dump to 'Azure Log Analytics' workspace. If `FALSE`, then the local time captured by `Sys.time` is recorded in the time field.

Value

Returns a generator function typically to be used by `log_layout` function.

`.post_data` *Build and send a request to the POST API of 'Azure Log Analytics'*

Description

Build and send a request to the POST API of 'Azure Log Analytics'

Usage

```
.post_data(customer_id, shared_key, body, log_type)
```

Arguments

`customer_id`	`customer_id` of the 'Azure Log Analytics' workspace
`shared_key`	`shared_key` of the 'Azure Log Analytics' workspace
`body`	Content or message to be logged in JSON format
`log_type`	Log-Type as defined in 'Azure Log Analytics' document, for custom logging

Value

Returns the HTTP response object

`get_log_config` *Get configuration value*

Description

Get the configuration value of a specific key which was set (or not set) using `set_log_config` function. If nothing was set, then it reuses the default value defined in the function signature of `set_log_config` function.

Usage

```
get_log_config(key = NULL)
```

Arguments

`key`	Specify the key whose value needs to be extracted. `NULL` implies no specific key, rather all of them to be extracted at once.

Value

Returns the respective configuration value of the given `key`. If `key` is `NULL`, then all the configuration values will be returned together as a list.

Examples

```
# Get configuration value without setting anything
get_log_config("log_to_azure")
# Set some configuration first and then get the respective values
set_log_config(enforce_tz_utc = FALSE, log_to_azure = FALSE)
get_log_config("enforce_tz_utc")
get_log_config("log_to_azure")
# Reset the values
set_log_config()
get_log_config("log_to_azure")

# Extract list of all configurations
get_log_config()
```

logger_level *Logging related functions*

Description

Logger function defined which are created on top of log_level and layout_json - these are part of another package 'logger'. Additional capabilities have been added to those functions which enables this function to be able to send logs directly to the 'Azure Log Analytics' workspace, and also have control to post log outputs into the console - as per user input. Note that, logging threshold can be directly set (if needed) using log_threshold function from 'logger' package.

Usage

```
logger_level(
  ...,
  log_fields = get_log_config("log_fields"),
  additional_fields = get_log_config("additional_fields"),
  enforce_ascii_msg = get_log_config("enforce_ascii_msg"),
  enforce_tz_utc = get_log_config("enforce_tz_utc"),
  log_to_azure = get_log_config("log_to_azure"),
  log_type = get_log_config("log_type"),
  log_customer_id = Sys.getenv(get_log_config("customer_id_env"), "abcd"),
  log_shared_key = Sys.getenv(get_log_config("shared_key_env"), "abcd")
)

logger_info(...)

logger_error(...)

logger_warn(...)

logger_debug(...)

logger_fatal(...)

logger_success(...)

logger_trace(...)
```

Arguments

`...`	Content(s) of this argument is directly passed on to log_level function of the 'logger' package.
`log_fields`	Character vector of field names to be included in the JSON. These field names are automatically collected by get_logger_meta_variables function, please refer to that function's documentation to see which fields are collected.

`additional_fields`	A named vector of type list with key-value pairs of additional meta data which needs to be added in logging context on top of `log_fields`. The respective value of each key is expected to be of length 1. It is `NULL` by default.
`enforce_ascii_msg`	If `TRUE` (default), the logging message is guaranteed to have all non-ASCII characters escaped. If `FALSE`, the characters will be logged as-is. Please note, it is better to ensure ASCII, otherwise there might be error while sending the HTTP POST request to 'Azure Log Analytics' workspace.
`enforce_tz_utc`	If `TRUE` (default), the logging time field is converted to UTC timezone while sending the logging dump to 'Azure Log Analytics' workspace. If `FALSE`, then the local time captured by `Sys.time` is recorded in the time field.
`log_to_azure`	If `TRUE` (default), then logs will be sent to 'Azure Log Analytics' workspace and console. Else if `FALSE` then logs will not be sent to 'Azure Log Analytics' workspace, it will only be displayed on console, which is the default layout of `'logger'` package.
`log_type`	Single element character vector is expected. Logs will be posted to this event on 'Azure Log Analytics'. For details, check this: `https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-collector-api?tabs=python/`. Default value is `"log_from_r"`.
`log_customer_id`	Workspace ID of 'Azure Log Analytics' workspace. By default it fetches from the environment variable named `AZ_LOG_ID`. If the environment variable is not set, then a dummy value `"abcd"` is used. The environment variable's name can be modified by `set_log_config`
`log_shared_key`	Shared key of 'Azure Log Analytics' workspace. By default it fetches from the environment variable named `AZ_LOG_KEY`. If the environment variable is not set, then a dummy value `"abcd"` is used. The environment variable's name can be modified by `set_log_config`

Details

- Most of the arguments of this function have a default value which is read from the output of `get_log_config`. The idea is to run the `set_log_config` function once to define the default arguments; and use them automatically while logging anything without the need of specifying them every time it is triggered.
- 'Azure Log Analytics' workspace id and shared key are intentionally fetched from environment variables for security purpose. It is not a good practice to specify them explicitly. Using environment variable is one easy approach to potentially hide it from unintentional user.
- It may take ~5–10 minutes to see the logging messages on the 'Azure Log Analytics' portal after the first time a message is posted to a new custom log table.

Value

If `log_to_azure` is `FALSE` then log output is shown on console. Else, if `TRUE`, then log output is shown on console, as well as posted to 'Azure Log Analytics' workspace under the custom table name as specified by `log_type` argument. If POST request is unsuccessful, then additional warning message is thrown with POST request response. If POST request is successful, then it invisibly returns the `POST` object.

Note

Logging layout is set in JSON format, required to send to ’Azure Log Analytics’. Note that this layout modifies the global `namespace` of `'logger'` package by default - that is not important for this use case.

`logger_info` is a wrapper function around `logger_level` - logging level is set as `INFO` by default.

`logger_error` is a wrapper function around `logger_level` - logging level is set as `ERROR` by default.

`logger_warn` is a wrapper function around `logger_level` - logging level is set as `WARN` by default.

`logger_debug` is a wrapper function around `logger_level` - logging level is set as `DEBUG` by default.

`logger_fatal` is a wrapper function around `logger_level` - logging level is set as `FATAL` by default.

`logger_success` is a wrapper function around `logger_level` - logging level is set as `SUCCESS` by default.

`logger_trace` is a wrapper function around `logger_level` - logging level is set as `TRACE` by default.

Examples

```
# Define logging config and then use logger_* functions to log
set_log_config(log_to_azure = FALSE)
logger_level(logger::INFO, "logging message")
# Specify other arguments explicitly inside the logger_level function
logger_level(logger::INFO, "logging message", log_to_azure = FALSE)

# For ease, use wrapper functions instead of using `logger_level` function as
# below
logger_info("logging message info", log_to_azure = FALSE)

# Also, instead of writing `log_to_azure = FALSE` every time, set the
# configuration in one step using `set_log_config`, and continue to use
# wrapper functions as usual.
set_log_config(log_to_azure = FALSE)
logger_info("logging message info")

# Wrapper function for log level 'error'
logger_error("logging message error")

# Wrapper function for log level 'warn'
logger_warn("logging message warn")

# Change log threshold to debug
logger::log_threshold(logger::DEBUG)
# Wrapper function for log level 'debug'
logger_debug("logging message debug")
```

```
# Wrapper function for log level 'fatal'
logger_fatal("logging message fatal")

# Wrapper function for log level 'success'
logger_success("logging message success")

# Change logging threshold
logger::log_threshold(logger::TRACE)
# Wrapper function for log level 'trace'
logger_trace("logging message trace")
```

`set_log_config` *Set logging configuration*

Description

Set the logging configuration once by executing this function. There won't be any need to set them every time while logging something via `logger_level` or any wrapper of that, e.g. `logger_info` function(s).

Usage

```
set_log_config(
  log_fields = c("level", "time", "msg"),
  additional_fields = NULL,
  enforce_ascii_msg = TRUE,
  enforce_tz_utc = TRUE,
  log_to_azure = TRUE,
  log_type = "log_from_r",
  customer_id_env = "AZ_LOG_ID",
  shared_key_env = "AZ_LOG_KEY"
)
```

Arguments

`log_fields`	Character vector of field names to be included in the JSON. These field names are automatically collected by `get_logger_meta_variables` function, please refer to that function's documentation to see which fields are collected.
`additional_fields`	A named vector of type list with key-value pairs of additional meta data which needs to be added in logging context on top of `log_fields`. The respective value of each key is expected to be of length 1. It is `NULL` by default.
`enforce_ascii_msg`	If `TRUE` (default), the logging message is guaranteed to have all non-ASCII characters escaped. If `FALSE`, the characters will be logged as-is. Please note, it is better to ensure ASCII, otherwise there might be error while sending the HTTP POST request to 'Azure Log Analytics' workspace.

`enforce_tz_utc`	If `TRUE` (default), the logging time field is converted to UTC timezone while sending the logging dump to 'Azure Log Analytics' workspace. If `FALSE`, then the local time captured by `Sys.time` is recorded in the time field.
`log_to_azure`	If `TRUE` (default), then logs will be sent to 'Azure Log Analytics' workspace and console. Else if `FALSE` then logs will not be sent to 'Azure Log Analytics' workspace, it will only be displayed on console, which is the default layout of `'logger'` package.
`log_type`	Single element character vector is expected. Logs will be posted to this event on 'Azure Log Analytics'. For details, check this: `https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-collector-api?tabs=python/`. Default value is `"log_from_r"`.
`customer_id_env`	The name of the environment variable (default is `AZ_LOG_ID`) which stores the workspace ID of the 'Azure Log Analytics' workspace. Please refer `https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-collector-api?tabs=powershell#sample-requests/` to see how you may get the required workspace ID.
`shared_key_env`	The name of the environment variable (default is `AZ_LOG_KEY`) which stores the shared key of the 'Azure Log Analytics' workspace. Please refer `https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-collector-api?tabs=powershell#sample-requests/` to see how you may get the required shared key.

Value

It saves the configuration in an environment enclosed within this package. Returns nothing explicitly.

Examples

```
set_log_config(log_fields = c("level", "time", "msg", "user", "pid"))
set_log_config(enforce_tz_utc = FALSE, log_to_azure = FALSE)
```

Index