Package 'uptimeRobot'

May 8, 2026

Type Package

Version 1.0.0

Title Access the UptimeRobot Ping API

Description Provide a set of wrappers to call all the endpoints of UptimeRobot API which includes various kind of ping, keep-alive and speed tests. See <https://uptimerobot.com/> for more information.

Date 2015-10-21

URL https://gabrielebaldassarre.com/r/uptimerobot

BugReports https://github.com/theclue/uptimeRobot/issues

Depends R (>= 3.0)

Imports rjson, RCurl, plyr

Suggests testthat

License MIT + file LICENSE

NeedsCompilation no

BuildKeepEmpty TRUE

Author Gabriele Baldassarre [aut, cre]

Maintainer Gabriele Baldassarre <gabriele@gabrielebaldassarre.com>

Repository CRAN

Date/Publication 2015-10-22 15:23:18

Contents

`uptimerobot.account.details`
Get the account details for who is linked to the given API key

---

Description

`uptimerobot.account.details` returns a list or a vector with the account details connected to the given api key.

Usage

```
uptimerobot.account.details(api.key, unlist = FALSE)
```

Arguments

`api.key`	string with a valid key for connecting to Uptimerobot API.
`unlist`	logical. Set to `TRUE` to unlist the output to a named vector, `FALSE` to get a named list.

Value

A list or a vector with the account details.

Author(s)

Gabriele Baldassarre

Examples

```
## Not run:
# Let's assume the api.key is available into the environment variable KEY
api.key <- Sys.getenv("KEY", "")

# Returns details as a list
details.list <- uptimerobot.account.details(api.key)

# Returns details as a vector
details.num <- uptimerobot.account.details(api.key, unlist = TRUE)

## End(Not run)
```

uptimerobot.contact.delete *Delete an alert contact*

Description

uptimerobot.contact.delete removes an alert contanct, unlinking from all the registered monitors.

Usage

```
uptimerobot.contact.delete(api.key, id)
```

Arguments

api.key	string with a valid key for connecting to Uptimerobot API.
id	numeric or integer with the ID of the contact to delete.

Value

The function returns TRUE in case success. An error is thrown otherwise.

Author(s)

Gabriele Baldassarre

Examples

```
## Not run:
 # Let's assume the api.key is available into the environment variable KEY
 api.key <- Sys.getenv("KEY", "")

 # Delete the contact with id=12345678
 if(uptimerobot.contact.delete(api.key, 12345678){
   message("Alert contact successfully deleted!")
 }

## End(Not run)
```

`uptimerobot.contact.new` *Add a new alert contact*

Description

`uptimerobot.contact.new` creates a new alert contact with the given properties.

Usage

```
uptimerobot.contact.new(api.key, type, value, friendly.name)
```

Arguments

`api.key`	string with a valid key for connecting to Uptimerobot API.
`type`	string or numeric with the type of the contact. You can use both the value (string) or the index (numeric) here.
`value`	string with the value of the contact (ie. the email address).
`friendly.name`	string the friendly (screen) name of the contact.

Details

The alert contacts are whom to be notified when the monitor goes up/down.

The index lookup keys and values are available on the Uptimerobot API page on https://uptimerobot.com/api

Value

The function returns the ID of the newly created contact in case success. An error is thrown otherwise.

Author(s)

Gabriele Baldassarre

See Also

`uptimerobot.contacts`, `uptimerobot.contact.delete`

Examples

```
## Not run:
 # Let's assume the api.key is available into the environment variable KEY
 api.key <- Sys.getenv("KEY", "")
 # Create a new contact and get the ID
 contact.new <- uptimerobot.contact.new(api.key, type = "email", value = "foo@bar.com", "John Doe")
```

```
 # Get informations about this new contact
 contact.detail <- uptimerobot.contacts(api.key, contacts = contact.new)

## End(Not run)
```

`uptimerobot.contacts` *Get general informations about the alert contacts*

Description

`uptimerobot.contacts` extracts a dataset with general informations for a set of contacts used to be alert in case of up/down of the given monitors.

Usage

```
uptimerobot.contacts(api.key, contacts = NULL, limit = 50, offset = 0,
  fields = uptimerobot.fields("contact")$typical)
```

Arguments

`api.key`	A valid key for connecting to UptimeRobors public API.
`contacts`	vector or comma-delimited string with the IDs of the contacts to get. If the argument is NULL or missing, all the available contacts will be returned.
`limit`	An integer value used for pagination. Defines the max number of records to return in each page. Default and max. is 50.
`offset`	An integer value to set the index of the first monitor to get (used for pagination).
`fields`	vector or comma-delimited string with the general informations to include in the output dataset. You may use the helper function `uptimerobot.fields` if you don't want to manually compile the list of fields.

Details

The alert contacts are whom to be notified when the monitor goes up/down.

If a vector of contact IDs is not given, the function will return data for all the available contacts.

The API uses pagination and returns no more than 50 contacts on each page. Use `limit` and `offset` to set a different number of monitors to get on each page and to move between pages. Leave default values to get all the data.

Value

A dataset with general informations about the contacts.

Author(s)

Gabriele Baldassarre

See Also

uptimerobot.monitors

Examples

```
## Not run:
# Let's assume the api.key is available into the environment variable KEY
api.key <- Sys.getenv("KEY", "")

# Returns all the contacts with a default set of attributes
contacts.df <- uptimerobot.contacts(api.key)

# Returns all the contacts and all the attributes
contacts.full.df <- uptimerobot.contacts(api.key, fields=uptimerobot.fields("contact")$full))

# Returns only the two contacts with ID: 1234, 5678
contacts.df <- uptimerobot.contacts(api.key, c("1234", "5678"))

## End(Not run)
```

---

uptimerobot.fields *Get a list of the available fields for various endpoints*

Description

uptimerobots.fields returns a list of vectors of available fields for commodity uses. Use it to avoid manually typing long list of fields in vectors or comma-delimited strings when used in various endpoints.

Usage

```
uptimerobot.fields(type)
```

Arguments

type	string with the type of fields to be reported. Only monitor and contact are currently supported.

Details

Use the type parameter to choose which set of fields to return in a list of vectors. These endpoints are currently supported: monitor and contact.

Value

The function returns a list of 3 elements which in turn contains a vector of available fields for a given `set` each. The returned elements are:

1. `typical` returns a typical set of fields, used in most situations;
2. `full` returns the full set of available fields, including passwords and other potentially confidential data;
3. `compact` return a minimal set of fields.

Author(s)

Gabriele Baldassarre

See Also

`uptimerobot.monitors`, `uptimerobot.contacts`

`uptimerobot.monitor.contacts`

Get contacts informations for one or more monitors

Description

`uptimerobot.monitor.contacts` return a dataset with all the alert contacts that will be triggered when a log is collected for the given monitors IDs.

Usage

```
uptimerobot.monitor.contacts(api.key, monitors, limit = 50, offset = 0)
```

Arguments

`api.key`	A valid key for connecting to UptimeRobors public API.
`monitors`	vector or comma-delimited string with the IDs of the monitors to get.
`limit`	An integer value used for pagination. Defines the max number of records to return in each page. Default and max. is 50.
`offset`	An integer value to set the index of the first monitor to get (used for pagination).

Details

The API uses pagination and returns no more than 50 monitors on each page. Use `limit` and `offset` to set a different number of monitors to get on each page and to move between pages. Leave default values to get all the data.

Value

A dataset with the alert contacts.

Author(s)

Gabriele Baldassarre

See Also

uptimerobot.monitors, uptimerobot.monitor.logs, uptimerobot.monitor.responses

Examples

```
## Not run:
# Let's assume the api.key is available into the environment variable KEY
api.key <- Sys.getenv("KEY", "")

# Returns all the monitors IDs. Since the function always return a data.frame
# (even if you ask only for a column), you have to reference the column to get a character vector.
monitors.id <- uptimerobot.monitors(api.key, fields="id")$id

# Returns all the contacts registered for the given monitors
logs.df <- uptimerobot.monitor.contacts(api.key, monitors=monitors.id)

## End(Not run)
```

---

uptimerobot.monitor.delete

Delete a monitor

Description

`uptimerobot.monitor.delete` remove a monitor and all existing statistics of it.

Usage

```
uptimerobot.monitor.delete(api.key, id)
```

Arguments

`api.key`	string with a valid key for connecting to Uptimerobot API.
`id`	numeric or integer with the ID of the monitor to delete.

Value

The function returns `TRUE` in case success. An error is thrown otherwise.

Author(s)

Gabriele Baldassarre

Examples

```
## Not run:
 # Let's assume the api.key is available into the environment variable KEY
 api.key <- Sys.getenv("KEY", "")

 # Create a monitor and get its monitor.id
 monitor.id <- uptimerobot.monitor.new(api.key,
   friendly.name="Open Analytics",
   url="https://gabrielebaldassarre.com", type="http"
 )

 # Change the friendly name of the monitor
  if(uptimerobot.monitor.edit(api.key,
     monitor.id,
     friendly.name="Open Analytics - gabrielebaldassarre.com"
  ){
   message("Monitor has been successfully edited!")
 }

 # Delete the just-made monitor
 if(uptimerobot.monitor.delete(api.key, monitor.id){
   message("Monitor has been successfully deleted!")
 }

## End(Not run)
```

`uptimerobot.monitor.edit` *Edit a monitor*

Description

`uptimerobot.monitor.edit` edits the properties for an existing monitor.

Usage

```
uptimerobot.monitor.edit(api.key, id, friendly.name = NULL, url = NULL,
  activate = TRUE, subtype = NULL, port = NULL, interval = NULL,
  keyword.type = NULL, keyword.value = NULL, HTTP.username = NULL,
  HTTP.password = NULL, alert.contacts = NULL)
```

Arguments

`api.key`	string with a valid key for connecting to Uptimerobot API.
`id`	numeric or integer with the ID of the monitor to edit.
`friendly.name`	string the friendly (screen) name of the monitor.
`url`	string with the URL/IP of the monitor.

activate	logical to set the status of the monitor. Set to TRUE to start the monitor or FALSE to put it in paused state.
subtype	string used only for "Port monitoring" to set which pre-defined port/service is monitored or if a custom port is monitored. You can use both the friendly name (string) or the index (integer) here.
port	string used only for "Port monitoring" to set the port monitored.
interval	integer with the interval for the monitoring check (in minutes).
keyword.type	required string in Keyword monitoring".
keyword.value	string with the value of the keyword (required for keyword monitoring).
HTTP.username	string used for password-protected web pages (HTTP Basic Auth). Set to empty string to erase the current username. Available for HTTP and keyword monitoring.
HTTP.password	string used for password-protected web pages (HTTP Basic Auth). Set to empty string to erase the current password.Available for HTTP and keyword monitoring.
alert.contacts	character vector or data frame with the IDs to alert each with their threshold and recurrence values.

Details

If a property has not to be updated, just omit it from the parameters or set to NA. To erase the value of a property, set it to an empty string, ie "", instead (not NA or NULL!).

The type of a monitor can not be edited (like changing a HTTP monitor into a Port monitor).

The alert contacts are whom to be notified when the monitor goes up/down.

Multiple alert contact IDs can be sent in a character vector or in a data frame. If you pass alert contact IDs in a vector, each element must be formatted in the form <id>_<threshold>_<recurrence> (note the underscores). If you prefer to format it as a data.frame, it must have these three columns: id, threshold, recurrence, numeric or integer. Order of the columns doesn't matter.

Please note that thresholds and recurrences can be omitted (default to zero) and, as they are only available in the Pro Plan, they are always 0 in the Free Plan.

Value

The function returns TRUE in case success. An error is thrown otherwise.

Author(s)

Gabriele Baldassarre

Examples

```
## Not run:
 # Let's assume the api.key is available into the environment variable KEY
 api.key <- Sys.getenv("KEY", "")

 # Create a monitor and get its monitor.id
```

```
monitor.id <- uptimerobot.monitor.new(api.key,
  friendly.name="Open Analytics",
  url="https://gabrielebaldassarre.com", type="http"
)

# Change the friendly name of the monitor
 if(uptimerobot.monitor.edit(api.key,
    monitor.id,
    friendly.name="Open Analytics - gabrielebaldassarre.com"
 ){
  message("Monitor has been successfully edited!")
}

# Delete the just-made monitor
if(uptimerobot.monitor.delete(api.key, monitor.id){
  message("Monitor has been successfully deleted!")
}

## End(Not run)
```

`uptimerobot.monitor.logs` *Get log records for one or more monitors*

Description

`uptimerobot.monitor.logs` return a dataset with all logged messages for the given monitors IDs.

Usage

```
uptimerobot.monitor.logs(api.key, monitors, limit = 50, offset = 0)
```

Arguments

`api.key`	A valid key for connecting to UptimeRobors public API.
`monitors`	vector or comma-delimited string with the IDs of the monitors to get.
`limit`	An integer value used for pagination. Defines the max number of records to return in each page. Default and max. is 50.
`offset`	An integer value to set the index of the first monitor to get (used for pagination).

Details

The API uses pagination and returns no more than 50 monitors on each page. Use `limit` and `offset` to set a different number of monitors to get on each page and to move between pages. Leave default values to get all the data.

Value

A dataset with the log events for the given monitors.

Author(s)

Gabriele Baldassarre

See Also

uptimerobot.monitors, uptimerobot.monitor.responses, uptimerobot.monitor.contacts

Examples

```
## Not run:
# Let's assume the api.key is available into the environment variable KEY
api.key <- Sys.getenv("KEY", "")

# Returns all the monitors IDs. Since the function always return a data.frame
# (even if you ask only for a column), you have to reference the column to get a character vector.
monitors.id <- uptimerobot.monitors(api.key, fields="id")$id

# Returns all the log events for the given monitors
logs.df <- uptimerobot.monitor.logs(api.key, monitors=monitors.id)

## End(Not run)
```

---

uptimerobot.monitor.new

Add a new monitor

Description

uptimerobot.monitor.new creates a new monitor with the given properties.

Usage

```
uptimerobot.monitor.new(api.key, friendly.name, url, type, subtype = NULL,
  port = NULL, interval = 5, keyword.type = NULL, keyword.value = NULL,
  HTTP.username = NULL, HTTP.password = NULL, alert.contacts = NULL)
```

Arguments

api.key	string with a valid key for connecting to Uptimerobot API.
friendly.name	string the friendly (screen) name of the monitor.
url	string with the URL/IP of the monitor.
type	string or integer with the type of the monitor. You can use both the friendly name (string) or the index (integer) here.
subtype	string used only for "Port monitoring" to set which pre-defined port/service is monitored or if a custom port is monitored. You can use both the friendly name (string) or the index (integer) here.
port	string used only for "Port monitoring" to set the port monitored.

interval	integer with the interval for the monitoring check (in minutes).
keyword.type	required string in Keyword monitoring".
keyword.value	string with the value of the keyword (required for keyword monitoring).
HTTP.username	string used for password-protected web pages (HTTP Basic Auth). Available for HTTP and keyword monitoring.
HTTP.password	string used for password-protected web pages (HTTP Basic Auth). Available for HTTP and keyword monitoring.
alert.contacts	character vector or data frame with the IDs to alert each with their threshold and recurrence values.

Details

The alert contacts are whom to be notified when the monitor goes up/down.

Multiple alert contact IDs can be sent in a character vector or in a data frame. If you pass alert contact IDs in a vector, each element must be formatted in the form `<id>_<threshold>_<recurrence>` (note the underscores). If you prefer to format it as a data.frame, it must have these three columns: `id`, `threshold`, `recurrence`, numeric or integer. Order of the columns doesn't matter.

Please note that thresholds and recurrences can be omitted (default to zero) and, as they are only available in the Pro Plan, they are always 0 in the Free Plan.

Value

A numeric with the ID of the newly created monitor in case of success. An error is thrown otherwise.

Author(s)

Gabriele Baldassarre

Examples

```
## Not run:
 # Let's assume the api.key is available into the environment variable KEY
 api.key <- Sys.getenv("KEY", "")

 # Create a monitor and get its monitor.id
 monitor.id <- uptimerobot.monitor.new(api.key,
   friendly.name="Open Analytics",
   url="https://gabrielebaldassarre.com", type="http"
 )

 # Change the friendly name of the monitor
  if(uptimerobot.monitor.edit(api.key,
     monitor.id,
     friendly.name="Open Analytics - gabrielebaldassarre.com"
  ){
   message("Monitor has been successfully edited!")
 }

 # Delete the just-made monitor
```

```
 if(uptimerobot.monitor.delete(api.key, monitor.id){
   message("Monitor has been successfully deleted!")
 }

## End(Not run)
```

`uptimerobot.monitor.reset` *Reset a monitor*

Description

`uptimerobot.monitor.reset` remove all the statistics and logs associated to a monitor ID.

Usage

```
uptimerobot.monitor.reset(api.key, id)
```

Arguments

`api.key`	string with a valid key for connecting to Uptimerobot API.
`id`	numeric or integer with the ID of the monitor to delete.

Value

The function returns `TRUE` in case success. An error is thrown otherwise.

Author(s)

Gabriele Baldassarre

Examples

```
## Not run:
 # Let's assume the api.key is available into the environment variable KEY
 api.key <- Sys.getenv("KEY", "")

# Get a list of all available monitors, and take the first id
monitors.id <- uptimerobot.monitors(api.key, fields="id")[1,1]

 # Reset the stats for that monitor
 uptimerobot.monitor.reset(api.key, monitor.id)


## End(Not run)
```

uptimerobot.monitor.responses
Get response times for one or more monitors

Description

`uptimerobot.monitor.responses` returns a dataset with all the response times for the given monitors IDs.

Usage

```
uptimerobot.monitor.responses(api.key, monitors, limit = 50, offset = 0)
```

Arguments

`api.key`	A valid key for connecting to UptimeRobors public API.
`monitors`	vector or comma-delimited string with the IDs of the monitors to get.
`limit`	An integer value used for pagination. Defines the max number of records to return in each page. Default and max. is 50.
`offset`	An integer value to set the index of the first monitor to get (used for pagination).

Details

The API uses pagination and returns no more than 50 monitors on each page. Use `limit` and `offset` to set a different number of monitors to get on each page and to move between pages. Leave default values to get all the data.

Value

A dataset with the response times for the given monitors.

Author(s)

Gabriele Baldassarre

See Also

`uptimerobot.monitors`, `uptimerobot.monitor.logs`, `uptimerobot.monitor.contacts`

Examples

```
## Not run:
# Let's assume the api.key is available into the environment variable KEY
api.key <- Sys.getenv("KEY", "")

# Returns all the monitors IDs. Since the function always return a data.frame
# (even if you ask only for a column), you have to reference the column to get a character vector.
monitors.id <- uptimerobot.monitors(api.key, fields="id")$id
```

```
# Returns all the ping events for the given monitors
logs.df <- uptimerobot.monitor.responses(api.key, monitors=monitors.id)

## End(Not run)
```

`uptimerobot.monitors` *Get general informations about monitors*

Description

`uptimerobots.monitors.responses` return a dataset with general informations for a set of monitors.

Usage

```
uptimerobot.monitors(api.key, monitors = NULL, types = NULL,
  statuses = NULL, search = NULL, summary = list(), limit = 50,
  offset = 0, fields = uptimerobot.fields("monitor")$typical)
```

Arguments

`api.key`	A valid key for connecting to UptimeRobors public API.
`monitors`	vector or comma-delimited string with the IDs of the monitors to get. If not used or set to `NULL`, will return all monitors in an account.
`types`	vector or comma-delimited string of monitor types. If not used or set to `NULL`, the function will return all monitors types (HTTP, keyword, ping..) in an account. Else, it is possible to define any number of monitor types. You can use both the friendly name (string) or the index (integer) here.
`statuses`	vector or comma-delimited string of monitor statuses. If not used or set to `NULL`, the function will return all monitors statuses (up, down, paused) in an account. Else, it is possible to define any number of monitor statuses. You can use both the friendly name (string) or the index (integer) here.
`search`	An optional keyword of to search within monitor URL or friendly name to get filtered results.
`summary`	list of logical values to flag summary indicators to add to the output dataset.
`limit`	An integer value used for pagination. Defines the max number of records to return in each page. Default and max. is 50.
`offset`	An integer value to set the index of the first monitor to get (used for pagination).
`fields`	vector or comma-delimited string with the general informations to include in the output dataset. You may want to use the helper function `uptimerobot.fields` if you don't want to manually compile the list of fields.

Details

If a vector of monitor is not given, the function will return data for all the available monitors.

`summary` parameter expect a lists of three named logic values that set which columns of additional statistics for each monitor must be added to output dataset for each available monitor. These are summary values only, as the instances are obtained using a set of dedicated functions.

1. `response.times` set to `TRUE` to add a column with the number of pings with response times available for the monitor to the output. These values can be queried using `uptimerobot.monitor.responses` function.
2. `log.records` set to `TRUE` to add a column with the number of log entries recorded for the monitor to the output. These records can be queried using `uptimerobot.monitor.logs` function.
3. `alert.contacts` set to `TRUE` to add a column with the number of alert contacts binded to the monitor to the output. Detailed informations about these contacts can be queried using `uptimerobot.monitor.contacts` function.

You may just add the elements you want to include into the list, as they default to `FALSE` if missing. Set an empty list to exclude all the summary statistics from the output.

The API uses pagination and returns no more than 50 monitors on each page. Use `limit` and `offset` to set a different number of monitors to get on each page and to move between pages. Leave default values to get all the data.

Value

A dataset with general informations about the given monitors

Author(s)

Gabriele Baldassarre

See Also

`uptimerobot.monitor.responses`, `uptimerobot.monitor.logs`, `uptimerobot.monitor.contacts`

Examples

```
## Not run:
# Let's assume the api.key is available into the environment variable KEY
api.key <- Sys.getenv("KEY", "")

# Returns all the  monitors with a default set of attributes
monitors.df <- uptimerobot.monitors(api.key)

#' # Returns all the monitors of 'keyword' type
monitors.kwd..df <- uptimerobot.monitors(api.key, type="keyword")

# Returns all the monitors and all the attributes
monitors.full.df <- uptimerobot.monitors(api.key, fields=uptimerobot.fields("monitor")$full))

# Returns only the two monitors with ID: 1234, 5678
```

```
monitors.df <- uptimerobot.monitors(api.key, c("1234", "5678"))

## End(Not run)
```

Index