Package ‘pushoverr’

November 16, 2021

Type Package

Title Send Push Notifications using 'Pushover'

Version 1.1.0

Date 2021-11-16

Description Send push notifications to mobile devices or the desktop using 'Pushover' <https://pushover.net>. These notifications can display things such as results, job status, plots, or any other text or numeric data.

License BSD_2_clause + file LICENSE


BugReports https://github.com/briandconnelly/pushoverr/issues

Depends R (>= 3.0.0)

Imports checkmate, cli, glue, httr, rlang

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

ByteCompile no

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.1.2

VignetteBuilder knitr

Author Brian Connelly [aut, cre] (<https://orcid.org/0000-0002-9948-0379>)

Maintainer Brian Connelly <bdc@bconnelly.net>

Repository CRAN

Date/Publication 2021-11-16 05:30:02 UTC
cancel_retries

**Description**

cancel_retries() stops Pushover from sending repeat messages for un-acknowledged emergency priority notifications.

**Usage**

cancel_retries(receipt, app = get_pushover_app())

**Arguments**

- receipt: The receipt from sending an emergency message
- app: application token (see `set_pushover_app()`)

**Value**

an invisible list containing the following fields:

- **status**: request status (1 = success)
- **request**: unique request ID
- **errors**: a list of error messages (only for unsuccessful requests)
- **raw**: the raw `httr::response` object
check_receipt

Examples

```r
## Not run:
msg1 <- pushover_emergency(message = "Test emergency message")
cancel_retries(receipt = msg1$receipt)
## End(Not run)
```

check_receipt  

Check whether an emergency priority message was received

Description

`check_receipt()` checks the status of an emergency priority message, receiving information about whether and by whom it was acknowledged, when the message was last delivered, whether a callback URL was visited, and more.

`is.acknowledged()` returns a logical value indicating whether or not the emergency message was acknowledged.

Usage

```r
check_receipt(receipt, app = get_pushover_app())
```

```r
is.acknowledged(receipt, app = get_pushover_app())
```

Arguments

- `receipt`  
  receipt ID from sending an emergency message
- `app`  
  application token (see `set_pushover_app()`)

Value

a list containing the following fields:

- `status`: request status (1 = success)
- `acknowledged`: number indicating whether (1) or not (0) notification has been acknowledged
- `acknowledged_at`: Unix timestamp indicating when notification was acknowledged, or 0
- `acknowledged_by`: key of the user who first acknowledged the notification, or ""
- `acknowledged_by_device`: name of the device on which the first user acknowledged the notification
- `last_delivered_at`: Unix timestamp of when the notification was last acknowledged, or 0
- `expired`: whether (1) or not (0) the notification has expired
- `expires_at`: Unix timestamp indicating when the notification will no longer be retried
- `called_back`: whether (1) or not (0) the callback URL has been visited
- `called_back_at`: Unix timestamp indicating when the callback URL was visited
- `request`: unique request ID
- `errors`: a list of error messages (only for unsuccessful requests)
- `raw`: the raw `http::response` object
Examples

```r
## Not run:
msg1 <- pushover_emergency(message = "Test emergency message")
check_receipt(receipt = msg1$receipt)
is.acknowledged(receipt = msg1$receipt)

## End(Not run)
```

---

**get_devices**

Get a list of the user's registered devices

Description

`get_devices()` queries the Pushover API for a list of the devices that have been registered by the given user.

`is.registered_device()` determines whether the given device is registered to the given user.

Usage

```r
get_devices(user = get_pushover_user(), app = get_pushover_app())

is.registered_device(
  device,
  user = get_pushover_user(),
  app = get_pushover_app()
)
```

Arguments

- `user`: Pushover user key (see `set_pushover_user()`)
- `app`: Pushover application token (see `set_pushover_app()`)
- `device`: The name of a device

Value

- `get_devices()` returns a list of device names registered by the given user.
- `is.registered_device()` returns a logical value for each of the given devices that indicates whether or not that device is registered to the given user.

Examples

```r
## Not run:
get_devices(
  user = "uQ1Rzpo4DXghDmr9QzzfQu27cmVRsG",
  app = "azGDOREPK8gMaC0QOYAMyEEuzJnyUI"
)
```
get_group_info

## End(Not run)
## Not run:
is.registered_device(device = "phone")

## End(Not run)

---

### get_group_info

*Get information about a Pushover group*

**Description**

Get information about a Pushover group

**Usage**

```r
get_group_info(group, app = get_pushover_app())
```

**Arguments**

- **group**: group key
- **app**: application token (see `set_pushover_app()`)

**Value**

A list containing information for the given group. Fields include:

- **name**: the group's name
- **users**: list containing information about each user in the group
- **status**: request status (1 = success)
- **request**: unique request ID
- **raw**: the raw `http::response` object

**Examples**

```r
## Not run:
get_group_info(group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF")

## End(Not run)
```
get_pushover_limits

Get usage and limit information for Pushover applications

Description

get_pushover_limits() retrieves the message usage and limit information for the given application.

Usage

get_pushover_limits(app = get_pushover_app())

Arguments

app application token (see set_pushover_app())

Value

A list containing messaging usage for the given app. Fields include:

- limit: Number of messages allowed per month
- remaining: Number of remaining messages in current month
- reset: Unix timestamp indicating when message count is reset
- status: request status (1= success)
- request: unique request ID
- raw: the raw httr::response object

Note

This information can alternatively be gotten by examining the headers in the response to previous API calls. Look for headers x-limit-app-limit, x-limit-app-remaining, and x-limit-app-reset. For example, if x stores the response from a pushover() call, httr::headers(x$raw) will return all of the headers included in the response.

Examples

```r
## Not run:
lims <- get_pushover_limits(app = "azGDORePK8gMaCQYOAMMyEzJnyUi")
## End(Not run)
```
get_pushover_sounds

Get a list of sounds available for Pushover notifications

Description
Get a list of sounds available for Pushover notifications

Usage
get_pushover_sounds(app = get_pushover_app())

Arguments
app application token (see set_pushover_app())

Value
A list of available sounds and their descriptions.

Examples
## Not run:
get_pushover_sounds(app = "azGDORePK8gMaC0QOYAMyEEuZJnyUI")
## End(Not run)

group_add_user
Manage group subscriptions

Description
These functions manage a user’s membership in a Pushover delivery group
group_add_user() adds a user to a group. Optionally, a device can be specified on which that user
will receive notifications
group_delete_user() removes a user from a group
group_disable_user() temporarily disables a user from receiving group notifications.
group_enable_user() re-enables a user to receive group notifications for a group
Usage

```r
group_add_user(
  group,
  user,
  app = get_pushover_app(),
  device = NULL,
  memo = NULL
)

group_delete_user(group, user, app = get_pushover_app())

group_disable_user(group, user, app = get_pushover_app())

group_enable_user(group, user, app = get_pushover_app())
```

Arguments

- `group`: group key
- `user`: user key
- `app`: application token (see `set_pushover_app()`)
- `device`: (optional) device name to receive messages (defaults to all devices)
- `memo`: (optional) memo about the user

Value

An invisible list containing the following fields:

- `status`: request status (1 = success)
- `request`: unique request ID
- `raw`: the raw `httr::response` object

Examples

```r
## Not run:
group_add_user(
  group = "gznej3rKEVAvPUux9vvNnqpmZpokzF",
  user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG",
  device = "phone"
)

## End(Not run)

## Not run:
group_delete_user(
  group = "gznej3rKEVAvPUux9vvNnqpmZpokzF",
  user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG"
)

## End(Not run)
```

group_rename

## Not run:
group_disable_user(
  group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF",
  user = "uQiRzpo4DXghDmr9QzsfQu27cmVRsG"
)

## End(Not run)

## Not run:
group_enable_user(
  group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF",
  user = "uQiRzpo4DXghDmr9QzsfQu27cmVRsG"
)

## End(Not run)

---

group_rename  Rename a delivery group

### Description

Rename a delivery group

### Usage

```r
group_rename(group, name, app = get_pushover_app())
```

### Arguments

- `group`: group key
- `name`: new group name
- `app`: application token (see `set_pushover_app()`)

### Value

An invisible list containing the following fields:

- **status**: request status (1 = success)
- **request**: unique request ID
- **raw**: the raw `httr::response` object

### Examples

```r
## Not run:
group_rename(
  group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF",
  name = "Coffee Party"
)
## End(Not run)
```
pushover

Send a message using Pushover

Description

pushover() sends a message (push notification) to a user or group. Messages can be given different priorities, play different sounds, or require acknowledgments. pushover_normal(), pushover_silent, pushover_quiet, pushover_high, and pushover_emergency functions send messages with those priorities.

Usage

pushover(
  message,
  title = NULL,
  priority = 0,
  attachment = NULL,
  user = get_pushover_user(),
  app = get_pushover_app(),
  device = NULL,
  sound = NULL,
  url = NULL,
  url_title = NULL,
  format = c("html", "monospace"),
  retry = 60,
  expire = 3600,
  callback = NULL,
  timestamp = NULL
)

pushover_silent(message, ...)

pushover_quiet(message, ...)

pushover_normal(message, ...)

pushover_high(message, ...)

pushover_emergency(message, ...)

Arguments

message The message to be sent (max. 1024 characters). Messages use glue::glue() for formatting and interpolation.

title (optional) The message’s title. Titles use glue::glue() for formatting and interpolation.
### priority
Message priority (-2: silent, -1: quiet, 0: normal (default), 1: high, 2: emergency)

### attachment
Path of file attachment to include. File must be image format (bmp, jpg, png, tif) and no larger than 2.6 MB.

### user
user/group key (see `set_pushover_user()`)

### app
application token (see `set_pushover_app()`)

### device
(optional) name of the device(s) to send message to. Defaults to all devices.

### sound
(optional) name of the sound to play (see `https://pushover.net/api#sounds`)

### url
(optional) supplementary URL to display with message

### url_title
(optional) title to show for supplementary URL

### format
Message formatting. If html (default), messages can include a limited subset of HTML formatting. If monospace, text is formatted using monospace font.

### retry
(optional) how often (in seconds) to repeat emergency priority messages (min: 30 seconds; default: 60 seconds)

### expire
(optional) how long (in seconds) emergency priority messages will be retried (max: 86400 seconds; default: 3600 seconds)

### callback
(optional) callback URL to be visited (HTTP POST) once an emergency priority message has been acknowledged (details)

### timestamp
(optional) a Unix timestamp containing the date and time to display to the user instead of the time at which the message was received

### ... Additional arguments to pass to `pushover()`

#### Value

an invisible list containing the following fields:

- **status**: request status (1 = success)
- **request**: unique request ID
- **raw**: the raw `http::response` object
- **receipt**: a receipt ID (only for emergency priority messages)
- **errors**: a list of error messages (only for unsuccessful requests)

#### Examples

```r
## Not run:
pushover(message = "Hola Mundo!"

## End(Not run)
```
pushover_api

Description

pushover_api() allows commands to be issued using the Pushover API. This is a generic function that is meant to be used by higher level functions. In most instances, more specific functions should be used (e.g., pushover()).

Usage

pushover_api(verb, url, ...)

Arguments

verb  Name of verb to use.
url   the url of the page to retrieve
...   Arguments passed on to http::VERB
config Additional configuration settings such as http authentication (authenticate()), additional headers (add_headers()), cookies (set_cookies()) etc. See config() for full details and list of helpers.
body  One of the following:
   • FALSE: No body. This is typically not used with POST, PUT, or PATCH, but can be useful if you need to send a bodyless request (like GET) with VERB().
   • NULL: An empty body
   • "": A length 0 body
   • upload_file("path/"): The contents of a file. The mime type will be guessed from the extension, or can be supplied explicitly as the second argument to upload_file()
   • A character or raw vector: sent as is in body. Use content_type() to tell the server what sort of data you are sending.
   • A named list: See details for encode.
encode If the body is a named list, how should it be encoded? Can be one of form (application/x-www-form-urlencoded), multipart, (multipart/form-data), or json (application/json).
   For "multipart", list elements can be strings or objects created by upload_file().
   For "form", elements are coerced to strings and escaped, use I() to prevent double-escaping. For "json", parameters are automatically "unboxed" (i.e. length 1 vectors are converted to scalars). To preserve a length 1 vector as a vector, wrap in I(). For "raw", either a character or raw vector. You’ll need to make sure to set the content_type() yourself.
**set_pushover_app**

The handle to use with this request. If not supplied, will be retrieved and reused from the `handle_pool()` based on the scheme, hostname and port of the url. By default **httr** requests to the same scheme/host/port combo. This substantially reduces connection time, and ensures that cookies are maintained over multiple requests to the same host. See `handle_pool()` for more details.

**Value**

A list containing the following fields and any other fields related to the specific API call:

- status: request status (1 = success)
- request: unique request ID
- raw: the raw `http::response` object
- errors: a list of error messages (only for unsuccessful requests)

**Examples**

```r
## Not run:
pushover_api(
  verb = "GET",
  url = "https://api.pushover.net/1/sounds.json",
  query = list(token = "azGDORePK8gMaC0QYAMYEEuJnyUi")
)
## End(Not run)
```

---

**set_pushover_app**  
*Set, get, and unset the Pushover application token*

**Description**

`set_pushover_app()` sets the Pushover application token to be used in subsequent commands, `get_pushover_app()` gets the application token that is currently set, and `unset_pushover_app()` unsets the token.

**Usage**

```r
set_pushover_app(token = NULL, ask = is_interactive())

get_pushover_app(ask = is_interactive())

unset_pushover_app()
```
set_pushover_user

Arguments

- **token**: The application token to be used. If none is provided, a prompt will request the token (interactive sessions only).
- **ask**: Whether or not to ask for the token if none is provided. Note that this option only works in interactive sessions.

Details

- `set_pushover_app()` only sets the Pushover app token for the current session. If a different value is specified in `.Renviron`, that value will be used in future sessions. Similarly, `unset_pushover_app()` will only unset the app token for the current session.
- To receive an application token, register a new application after logging in to your account at [https://pushover.net/apps](https://pushover.net/apps).

Value

- `get_pushover_app()` returns a string containing the current application token. If the token is not set but `ask` is `TRUE`, the user will be prompted for a token.

Examples

```r
## Not run:
set_pushover_app(token = "azGDORePK8gMaC0QOYAMyEEuzJnyU1")
## End(Not run)
```

Description

- `set_pushover_user()` sets the Pushover user or group key to be used in subsequent commands,
- `get_pushover_user()` gets the user or group key that is currently set, and `unset_pushover_user()` unsets the key.

Usage

```r
set_pushover_user(user = NULL, ask = is_interactive())
get_pushover_user(ask = is_interactive())
unset_pushover_user()
set_pushover_group(user = NULL, ask = is_interactive())
get_pushover_group(ask = is_interactive())
unset_pushover_group()
```
**Arguments**

user
The user or group key to be used. If none is provided, a prompt will request the key.

ask
Whether or not to ask for the key if none is provided. Note that this only works for interactive sessions.

**Details**

set_pushover_group(), get_pushover_group(), and unset_pushover_group() are aliases for these functions.

set_pushover_user() only sets the Pushover user or group for the current session. If a different value is specified in .Renviron, that value will be used in future sessions. Similarly, unset_pushover_user() will only unset the user or group for the current session.

User keys can be found within the settings of the Pushover app or by logging in to https://pushover.net. Group keys can be found after creating a delivery group in your account on https://pushover.net.

**Value**

get_pushover_user() returns a string containing the current user or group key or an empty string if not set. If the user is not set but ask is TRUE, the user will be prompted for a key.

**Examples**

```r
## Not run:
set_pushover_user(user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG")
```

```r
## End(Not run)
```

---

**update_glance**

*Update a Pushover glance data*

**Description**

Glances allow you to push small pieces of data to a frequently-updated screen such as a smartwatch or a lock screen. At least one of the title, text, subtext, count, or percent arguments must be specified.

**Usage**

```r
update_glance(
  title = NULL,
  text = NULL,
  subtext = NULL,
  count = NULL,
  percent = NULL,
)```
user = get_pushover_user()
app = get_pushover_app()
device = NULL

Arguments

- **title** (optional) a description of the data being shown, such as "Widgets Sold" (max. 100 characters)
- **text** (optional) the main line of data, used on most screens (max. 100 characters)
- **subtext** (optional) a second line of data (max. 100 characters)
- **count** (optional) integer value shown on smaller screens; useful for simple counts
- **percent** (optional) integer percent value (0..100) shown on some screens as a progress bar/circle
- **user** user/group key (see `set_pushover_user()`)
- **app** application token (see `set_pushover_app()`)
- **device** (optional) name of the device(s) to send message to. Defaults to all devices.

Value

an invisible list containing the following fields:

- **status**: request status (1 = success)
- **request**: unique request ID
- **raw**: the raw `httr::response` object
- **errors**: a list of error messages (only for unsuccessful requests)

Note

Glances are currently in beta, and features may change.

Examples

```r
## Not run:
update_glance(count = 37)

## End(Not run)
```
Description

verify_user() determines whether or not the given user or group is registered with Pushover, returning information about that user.

verify_group() is an alias for verify_user()

is.registered_user() indicates whether or not a given user ID is registered with Pushover.

Usage

verify_user(user, app = get_pushover_app(), device = NULL)
verify_group(user, app = get_pushover_app(), device = NULL)
is.registered_user(user, app = get_pushover_app(), device = NULL)
is.registered_group(user, app = get_pushover_app(), device = NULL)

Arguments

user user/group key to verify
app application token (see set_pushover_app)
device (optional) device to verify If supplied the device must be registered to the given user's account.

Value

verify_user() and verify_group() return a list containing the following fields:

• status: request status (1 = success)
• devices: a list of the user's devices
• request: unique request ID
• errors: a list of error messages (only for unsuccessful requests)
• raw: the raw http::response object

is.registered_user() and is.registered_group() return a logical value indicating whether or not the given user or group is registered.

Examples

## Not run:
verify_user(user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG")

## End(Not run)
Index

add_headers(), 12
authenticate(), 12

cancel_retries, 2
check_receipt, 3
config(), 12
content_type(), 12

get_devices, 4
get_group_info, 5
get_pushover_app (set_pushover_app), 13
get_pushover_group (set_pushover_user), 14
get_pushover_limits, 6
get_pushover_sounds, 7
get_pushover_user (set_pushover_user), 14

glue::glue(), 10
group_add_user, 7
group_delete_user (group_add_user), 7
group_disable_user (group_add_user), 7
group_enable_user (group_add_user), 7
group_rename, 9

handle_pool(), 13
httr::response, 2, 3, 5, 6, 8, 9, 11, 13, 16, 17
httr::VERB, 12

is.acknowledged (check_receipt), 3
is.registered_device (get_devices), 4
is.registered_group (verify_user), 17
is.registered_user (verify_user), 17

pushover, 10
pushover(), 6, 12
pushover_api, 12
pushover_emergency (pushover), 10
pushover_high (pushover), 10
pushover_normal (pushover), 10
pushover_quiet (pushover), 10
pushover_silent (pushover), 10

set_cookies(), 12
set_pushover_app, 13, 17
set_pushover_app(), 2–9, 11, 16
set_pushover_group (set_pushover_user), 14
set_pushover_user, 14
set_pushover_user(), 4, 11, 16

unset_pushover_app (set_pushover_app), 13
unset_pushover_group (set_pushover_user), 14
unset_pushover_user (set_pushover_user), 14
update_glance, 15
upload_file(), 12

verify_group (verify_user), 17
verify_user, 17