# Package ‘pushoverr’

November 23, 2016

**Type**  Package  
**Title**  Send Push Notifications using Pushover  
**Version**  1.0.0  
**Date**  2016-11-22  
**Description**  Send push notifications to mobile devices or the desktop using Pushover. These notifications can display job status, results, scraped web data, or any other text or numeric data.  
**License**  BSD_2_clause + file LICENSE  
**URL**  https://github.com/briandconnelly/pushoverr  
**BugReports**  https://github.com/briandconnelly/pushoverr/issues  
**ByteCompile**  no  
**NeedsCompilation**  no  
**Depends**  R (>= 3.0.0)  
**Imports**  assertthat, httr  
**RoxygenNote**  5.0.1  
**Author**  Brian Connelly [aut, cre]  
**Maintainer**  Brian Connelly <bdc@bconnelly.net>  
**Repository**  CRAN  
**Date/Publication**  2016-11-23 09:38:52

R topics documented:

cancel_retries  ..........................................................  2  
check_receipt  ..................................................................  3  
get_devices  .....................................................................  4  
get_group_info  .............................................................  5  
get_pushover_limits .......................................................  5  
get_pushover_sounds .......................................................  6  
group_rename  ...............................................................  7  
group_subscription ........................................................  8  
is.pushover_sound ..........................................................  9
cancel_retries

Description

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

cancel_receipt is deprecated in favor of cancel_retries

Usage

cancel_retries(receipt, app = get_pushover_app())

cancel_receipt(receipt, ...)

Arguments

- receipt: The receipt from sending an emergency message
- app: application token (see set_pushover_app)
- ...: Additional arguments (no longer used)

Value

- status: request status (1 = success)
- request: unique request ID
- errors: a list of error messages (only for unsuccessful requests)
- raw: the raw 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 the emergency message was acknowledged (TRUE) or not (FALSE).

Usage

```r
check_receipt(receipt, app = get_pushover_app())
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 response object
get_devices

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.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.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.device**: returns a logical value for each of the given devices that indicates whether (TRUE) or not (FALSE) that device is registered to the given user.

Examples

```r
## Not run:
get_devices(user = "uQiRzpo4DXghDmr9QzsfQu27cVRsG", app = "azGDORePK8gMaC0QYAMyUEuJnyUq")

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

## End(Not run)
```
get_group_info  

Get information about a Pushover group

**Description**

Get information about a Pushover group

**Usage**

```python
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 response object

**Examples**

```python
## Not run:
get_group_info(group = "gznjk3rKEvAvPuux9vvnqpmZpokzF")

## 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**

```python
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 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 = "azGDORePK8gMaC0QOYAMyEEuzJnyu")
## 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.
group_rename

Examples

```bash
## Not run:
group_rename(name = "new group")

## End(Not run)
```

Description

Rename a delivery group

Usage

```bash
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 response object

Examples

```bash
## Not run:
group_rename(group = "gznej3rKEVAvPUxu9v9vNqpm2pokzF",
             name = "Coffee Party")

## End(Not run)
```
group_subscription  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

```ruby
group_subscription(cmd, ...)

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

- **cmd** The group subscription command to execute
- **...** Specific arguments for the given group subscription command
- **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 response object
Examples

```sh
## Not run:
group_add_user(group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF",
                user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG",
                device = "phone")

## End(Not run)
## Not run:
group_delete_user(group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF",
                    user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG")

## End(Not run)
## Not run:
group_disable_user(group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF",
                    user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG")

## End(Not run)
## Not run:
group_enable_user(group = "gznej3rKEVAvPUxu9vvNnqpmZpokzF",
                   user = "uQiRzpo4DXghDmr9QzzfQu27cmVRsG")

## End(Not run)
```

---

**is.pushover_sound**

**Test whether the given sound is supported for Pushover messages**

**Description**

Test whether the given sound is supported for Pushover messages

**Usage**

```sh
is.pushover_sound(x)
```

**Arguments**

```sh
  x
```

A string containing a sound name

**Value**

A logical value indicating whether the given sound is supported by Pushover (TRUE) or not (FALSE)

**Examples**

```sh
is.pushover_sound("cosmic")
```
is.valid_app

Determine whether or not a given application token is valid

Description

is.valid_app determines whether or not a given application token is valid or not according to Pushover's specifications. It does not determine whether or not the given token is associated with an application.

is.valid_token is deprecated in favor of is.valid_app

Usage

is.valid_app(token)

is.valid_token(token)

Arguments

token A application token (e.g., "azGDORePK8gMaC0QOYAMyEEuzJnyUi")

Value

A logical value indicating whether the application token is valid (TRUE) or not (FALSE)

Note

To acquire an application token, create an app at https://pushover.net/apps

Examples

is.valid_app(token = "azGDORePK8gMaC0QOYAMyEEuzJnyUi")

is.valid_device

Determine whether or not a given device name is valid

Description

Valid device names are strings up to 25 characters long and can include letters, numbers, _, and -.

Usage

is.valid_device(device)
is.valid_receipt

Arguments
device one or more device names (e.g., "phone", c("phone", "tablet"))

Value
Logical value(s) indicating whether the corresponding device name is valid (TRUE) or not (FALSE)

Note
is.valid_device only determines whether a device name is valid. It does not determine whether that device is actually registered (see: is.device).

Examples
is.valid_device("my_phone")

is.valid_receipt
Determine whether or not a given message receipt is valid

Description
Receipts are 30-character strings containing letters and numbers ([A-Za-z0-9])

Usage
is.valid_receipt(receipt)

Arguments
receipt A message receipt (e.g., "KAWXTswy4cekx6vZbHBKbCKk1cl1df")

Details
is.valid_receipt determines whether or not a given message receipt is valid or not according to Pushover’s specifications. It does not determine whether or not the given receipt actually exists.

Value
A boolean value for each given message receipt ID indicating whether that receipt ID is valid (TRUE) or not (FALSE)

Examples
## Not run:
is.valid_receipt(receipt = "KAWXTswy4cekx6vZbHBKbCKk1cl1df")

## End(Not run)
is.valid_user

Determine whether a user or group key is valid

Description

is.valid_user and is.valid_group determine whether or not a given user/group key is valid. They do not determine whether or not the key is registered with Pushover.

is.valid_group is an alias for is.valid_user

Usage

is.valid_user(user)

is.valid_group(group)

Arguments

user user/group key to verify (e.g., "uQiRzpo4DXghDmr9QzzfQu27cmVRsG")
group user/group key to verify (e.g., "gznej3rKEVAxPUxu9vvNnqpmZpokzF")

Value

A logical value indicating whether (TRUE) or not (FALSE) the given user/group key is valid.

See Also

verify_user and verify_group determine whether or not the given user key is registered with Pushover

Examples

## Not run:

is.valid_user("uQiRzpo4DXghDmr9QzzfQu27cmVRsG")

## 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. The pushover_normal, pushover_silent, pushover_quiet, pushover_high, and pushover_emergency functions send messages with those priorities.
Usage

pushover(message, title = NULL, priority = 0, user = get_pushover_user(),
app = get_pushover_app(), device = NULL, sound = NULL, url = NULL,
url_title = NULL, 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)
title    (optional) The message’s title
priority Message priority (-2: silent, -1: quiet, 0: normal (default), 1: high, 2: emergency)
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
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
pushover

- raw: the raw 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)

### Description

pushover is a package for sending push notifications to mobile devices (iOS and Android) and the desktop using Pushover. Messages can be quickly sent using the `pushover` function. Apple Watch users can push small bits of data to their watches using `update_glance`.

### Details

To use pushover, you will first need your user key and an application token. The user key can be found within the settings of the app (e.g., uQ1Rzpo4DXghDmr9QzzfQu27cmVRsG) or by logging in to https://pushover.net. To receive an application token, register a new application after logging in to https://pushover.net/apps.

For more information, browse the package index by following the link at the bottom of this help page. For release information, run `news(package = "pushover")`.

### Disclaimer

This package and its author are not affiliated with Superblock, LLC, creators of Pushover.

### Author(s)

Brian Connelly

### See Also

GitHub Repository: https://github.com/briandconnelly/pushover

Bug Reports and Feature Requests: https://github.com/briandconnelly/pushover/issues

Pushover: https://pushover.net

Pushover API: https://pushover.net/api
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, visible = TRUE, ...)

Arguments

verb
The http method to use

url
The URL to visit

visible
Whether or not the result should be visible (default: TRUE)

... Any additional parameters to be passed to VERB

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 response object
- errors: a list of error messages (only for unsuccessful requests)

Examples

## Not run:
pushover_api(verb = "GET",
                   url = "https://api.pushover.net/1/sounds.json",
                   query = list(token = "azGDORePK8gMc8QOYAMyEEuzJnyui"))

## End(Not run)
set_pushover_app

---

**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. `pushover_app.isset` indicates whether or not the application token is set.

**Usage**

```r
set_pushover_app(token = NULL, ask = interactive())
get_pushover_app(ask = interactive())
unset_pushover_app()
pushover_app.isset()
```

**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 on 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.

`pushover_user.isset` returns a logical value indicating whether the application token is set (`TRUE`) or not (`FALSE`).

**Examples**

```r
## Not run:
set_pushover_app(token = "azGDORePK8gMaC0QYAMyEEuzJnyU!")
## End(Not run)
```
set_pushover_user

set_pushover_user: Set, get, and unset the Pushover user/group key

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. pushover_user.isset indicates whether or not the user/group key has been set.

Usage

```r
set_pushover_user(user = NULL, ask = interactive())
get_pushover_user(ask = interactive())
unset_pushover_user()
pushover_user.isset()
set_pushover_group(user = NULL, ask = interactive())
get_pushover_group(ask = 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
pushover_user.isset returns a logical value indicating whether the user/group is set (TRUE) or not (FALSE).

Examples

## Not run:
set_pushover_user(user = "uQ1Rzpo4DXghDmr9QzzfQu27cmVRsG")
## End(Not run)

stop_for_pushover_status

*Raise an Error if Pushover Request Failed*

Description

Examines a response from a Pushover API call for errors. If there were errors, execution is stopped, and an error message is shown.

Usage

stop_for_pushover_status(x)

Arguments

x  

a response object returned by an API call

Value

If request was successful, the response (invisibly).

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

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

**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 `response` object
- **errors**: a list of error messages (only for unsuccessful requests)

**Note**

Glances are currently in beta, and features may change.

**Examples**

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

## End(Not run)
```

---

| verify_user | User and group verification |

**Description**

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

- `is.registered_user` indicates whether or not a given user ID is registered with Pushover
- `verify_group` is an alias for `verify_user`
- `validate_key` is deprecated in favor of `verify_user`
- `is.valid_key` is deprecated in favor of `is.registered_user` or `is.registered_group`
Usage

verify_user(user, app = get_pushover_app(), device = NULL)

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

verify_group(user, app = get_pushover_app(), device = NULL)

validate_key(user, device = NA_character_, ...)

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

is.valid_key(user, device = NA, ...)

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.

... Additional arguments (no longer used)

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 response object

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

See Also

is.valid_user and is.valid_group to determine whether or not a user key has valid formatting, but is not necessarily registered.

Examples

## Not run:
verify_user(user = "uQiRzpo4DXghDmr9QzufQu27cmVRsG")

## End(Not run)
Index

cancel_receipt (cancel_retries), 2
cancel_retries, 2
check_receipt, 3

get_devices, 4
get_group_info, 5
get_pushover_app (set_pushover_app), 16
get_pushover_group (set_pushover_user), 17
get_pushover_limits, 5
get_pushover_sounds, 6
get_pushover_user (set_pushover_user), 17
group_add_user (group_subscription), 8
group_delete_user (group_subscription), 8
group_disable_user (group_subscription), 8
group_enable_user (group_subscription), 8
group_rename, 7
group_subscription, 8

is.acknowledged (check_receipt), 3
is.device, 11
is.device (get_devices), 4
is.pushover_sound, 9
is.registered_group (verify_user), 19
is.registered_user (verify_user), 19
is.valid_app, 10
is.valid_device, 10
is.valid_group, 20
is.valid_group (is.valid_user), 12
is.valid_key (verify_user), 19
is.valid_receipt, 11
is.valid_token (is.valid_app), 10
is.valid_user, 12, 20

pushover, 6, 12, 14, 15

pushover_app.isset (set_pushover_app), 16
pushover_emergency (pushover), 12
pushover_high (pushover), 12
pushover_normal (pushover), 12
pushover_quiet (pushover), 12
pushover_silent (pushover), 12
pushover_user.isset
  (set_pushover_user), 17
pushoverrr, 14
pushoverrr-package (pushoverrr), 14
response, 2, 3, 5–8, 14, 15, 18–20
set_pushover_app, 2–8, 13, 16, 19, 20
set_pushover_group (set_pushover_user), 17
set_pushover_user, 4, 13, 17, 19
stop_for_pushover_status, 18
unset_pushover_app (set_pushover_app), 16
unset_pushover_group
  (set_pushover_user), 17
unset_pushover_user
  (set_pushover_user), 17
update_glance, 14, 18

validate_key (verify_user), 19
VERB, 15

verify_group, 12
verify_group (verify_user), 19
verify_user, 12, 19