Package `osmapiR`

June 28, 2024

**Type** Package

**Title** 'OpenStreetMap' API

**Version** 0.1.0

**Maintainer** Joan Maspons &lt;joanmaspons@gmail.com&gt;

**Description** Interface to 'OpenStreetMap API' for fetching and saving data from/to the 'OpenStreetMap' database (&lt;https://wiki.openstreetmap.org/wiki/API_v0.6&gt;).

**License** GPL (&gt;= 3)


**BugReports** https://github.com/jmaspons/osmapiR/issues

**Imports** curl, httr2, xml2

**Suggests** httrtest2, httpuv, knitr, rmarkdown, testthat (&gt;= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**X-schema.org-keywords** open street map, openstreetmap, OSM, openstreetmap-api, osmapi, API

**NeedsCompilation** no

**Author** Joan Maspons [aut, cre, cph] (&lt;https://orcid.org/0000-0003-2286-8727&gt;), Jon Harmon [rev] (Jon reviewed the package for rOpenSci, see https://github.com/ropensci/software-review/issues/633, &lt;https://orcid.org/0000-0003-4781-4346&gt;), Carlos Cámara [rev] (Carles reviewed the package for rOpenSci, see https://github.com/ropensci/software-review/issues/633, &lt;https://orcid.org/0000-0002-9378-0549&gt;)

**Repository** CRAN

**Date/Publication** 2024-06-28 14:40:04 UTC
Contents

authenticate_osmapi ........................................... 3
osmapi_objects ............................................... 3
osmchange_create ........................................... 4
osmchange_delete .......................................... 5
osmchange_modify ......................................... 6
osm_api_versions ......................................... 8
osm_bbox_objects ......................................... 8
osm_capabilities .......................................... 10
osm_close_note ........................................... 11
osm_comment_changeset_discussion ...................... 12
osm_create_changeset .................................... 13
osm_create_comment_note ................................ 14
osm_create_gpx ........................................... 15
osm_create_note ........................................... 16
osm_create_object ....................................... 17
osm_delete_gpx ........................................... 18
osm_delete_note .......................................... 19
osm_delete_object ....................................... 20
osm_details_logged_user ................................ 21
osm_diff_upload_changeset ............................... 22
osm_download_changeset ................................ 24
osm_feed_notes ........................................... 25
osm_get_changesets ...................................... 26
osm_get_data_gpx ......................................... 28
osm_get_gpx_metadata .................................... 29
osm_get_notes ............................................ 30
osm_get_objects .......................................... 31
osm_get_points_gps ..................................... 33
osm_get_preferences_user ................................. 35
osm_get_user_details ................................... 36
osm_hide_comment_changeset_discussion .............. 38
osm_history_object ..................................... 39
osm_list_gpxs ........................................... 40
osm_permissions .......................................... 41
osm_query_changesets ................................... 42
osm_read_bbox_notes .................................... 45
osm_redaction_object .................................. 47
osm_relations_object ................................... 48
osm_search_notes ........................................ 49
osm_subscribe_changeset_discussion .................. 51
osm_update_gpx ........................................... 52
osm_update_object ...................................... 53
osm_ways_node ........................................... 54
set_osmapi_connection .................................. 55
tags_list2wide ........................................... 56

Index ......................................................... 58
authenticate_osmapi
Authenticate or logout osmapiR

Description
Log in/out osmapiR.

Usage
authenticate_osmapi()
logout_osmapi()

Details
All functions that require authentication will trigger the log in if the session is not yet authenticated, so calling this function is not really needed. Use authenticate_osmapi to sign in before executing scripts that require authentication to avoid interruptions.

Value
For authenticate_osmapi, print the user and permissions of the connection and return invisibly the display name of the logged user. logout_osmapi clear the OAuth2 token and can be useful to change user.

See Also
Other API functions: osm_api_versions(), osm_capabilities(), osm_permissions(), set_osmapi_connection()

Examples
## Not run:
authenticate_osmapi()
logout_osmapi()
## End(Not run)

osmapi_objects
osmapi_objects constructor

Description
osmapi_objects constructor

Usage
osmapi_objects(x, tag_columns, keep_na_tags = FALSE)
Arguments

x  data.frame representing OSM objects as rows. At least it has a type column with node, way or relation.
tag_columns  A vector indicating the name or position of the columns representing tags. If missing, it’s assumed that tags column contain the tags (see details).
keep_na_tags  If TRUE, don’t drop the empty tags specified in tag_columns and add NA as a value. Useful to remove specific tags with osmchange_modify() and specific tag_keys.

Value

An osmapi_objects

See Also

Other get OSM objects' functions: osm_bbox_objects(), osm_get_objects(), osm_history_object(), osm_relations_object(), osm_ways_node()

Examples

x <- data.frame(
  type = c("node", "node", "way"), id = 1:3, name = c(NA, NA, "My way")
)
x$members <- list(NULL, NULL, 1:2)
obj <- osmapi_objects(x, tag_columns = "name")
obj
Details

Objects IDs are unknown and will be allocated by the server. If id column is missing in x, a negative placeholders will be used. Check OsmChange page for details about how to refer to objects still not created to define the members of relations and nodes of ways.

Value

If format = "R", returns a osmapi_OsmChange data frame with one OSM edition per row. If format = "osc" or format = "xml", returns a xml2::xml_document following the OsmChange format that can be saved with xml2::write_xml() and opened in other applications such as JOSM.

The results are ready to send the editions to the servers with osm_diff_upload_changeset().

See Also

Other OsmChange's functions: osm_diff_upload_changeset(), osm_download_changeset(), osmchange_delete(), osmchange_modify()

Examples

d <- data.frame(
  type = c("node", "node", "way", "relation"),
  id = -(1:4),
  lat = c(0, 1, NA, NA),
  lon = c(0, 1, NA, NA),
  name = c(NA, NA, "My way", "Our relation"),
  type.1 = c(NA, NA, NA, "Column clash!")
)
d$members <- list(
  NULL, NULL, -(1:2),
  matrix(
    c("node", "-1", NA, "node", "-2", NA, "way", "-3", "outer"),
    nrow = 3, ncol = 3, byrow = TRUE, dimnames = list(NULL, c("type", "ref", "role")))
)
oobj <- osmapi_objects(d, tag_columns = c(name = "name", type = "type.1"))
osmcha <- osmchange_create(obj)
osmcha

osmchange_delete osmchange to delete existing OSM objects

Description

Prepare data to delete OSM objects.

Usage

osmchange_delete(x, delete_if_unused = FALSE, format = c("R", "osc", "xml"))
Arguments

x  A `osmapi_objects` or `data.frame` with the columns `type` and `id` for the objects to delete. Other columns will be ignored.

delete_if_unused
   If TRUE, the `if-unused` attribute will be added (see details). Can be a vector of length `nrow(x)`.

format
   Format of the output. Can be "R" (default), "osc" ("xml" is a synonym for "osc").

Details

If `if-unused` attribute is present, then the delete operation(s) in this block are conditional and will only be executed if the object to be deleted is not used by another object. Without the `if-unused`, such a situation would lead to an error, and the whole diff upload would fail. Setting the attribute will also cause deletions of already deleted objects to not generate an error.

Value

If `format` = "R", returns a `osmapi_OsmChange` data frame with one OSM edition per row. If `format` = "osc" or `format` = "xml", returns a `xml2::xml_document` following the OsmChange format that can be saved with `xml2::write_xml()` and opened in other applications such as JOSM.

The results are ready to send the editions to the servers with `osm_diff_upload_changeset()`.

See Also

Other OsmChange's functions: `osm_diff_upload_changeset()`, `osm_download_changeset()`, `osmchange_create()`, `osmchange_modify()`

Examples

```r
## Not run:
obj_id <- osmapi_objects(data.frame(
    type = c("way", "way", "relation", "node"),
    id = c("722379703", "629132242", "8387952", "4739010921")
))
osmchange_del <- osmchange_delete(obj_id)

## End(Not run)
```

---

**osmchange_modify**

*osmchange to modify existing OSM objects*

Description

Prepare data to update tags, members and/or latitude and longitude.
Usage

```r
osmchange_modify(
  x,
  tag_keys,
  members = FALSE,
  lat_lon = FALSE,
  format = c("R", "osc", "xml")
)
```

Arguments

- `x`: A `osmapi_objects` with the columns `type` and `id` with unique combinations of values plus columns specifying tags, members or latitude and longitude.
- `tag_keys`: A character vector with the keys of the tags that will be modified. If missing (default), all tags will be updated, removed or created. If `FALSE`, don’t modify tags.
- `members`: If `TRUE` and `x` has a `members` column, update the members of the ways and relations objects.
- `latLon`: If `TRUE` and `x` has a `lat` and `lon` columns, update the coordinates of the node objects.
- `format`: Format of the output. Can be "R" (default), "osc" ("xml" is a synonym for "osc").

Details

`x` should be a `osmapi_objects` or follow the same format. Missing tags or tags with NA in the value will be removed if `tag_keys` is not specified. See `osm_get_objects()` for examples of the format.

Value

If `format = "R"`, returns a `osmapi_OsmChange` data frame with one OSM edition per row. If `format = "osc"` or `format = "xml"`, returns a `xml2::xml_document` following the OsmChange format that can be saved with `xml2::write_xml()` and opened in other applications such as JOSM.

The results are ready to send the editions to the servers with `osm_diff_upload_changeset()`.

See Also

Other OsmChange’s functions: `osm_diff_upload_changeset()`, `osm_download_changeset()`, `osmchange_create()`, `osmchange_delete()`

Examples

```r
## Not run:
obj <- osm_get_objects(
  osm_type = c("node", "way", "way", "relation", "relation", "node"),
  osm_id = c("35308286", "13073736", "235744929", "40581", "341530", "1935675367"),
  version = c(1, 3, 2, 5, 7, 1) # Old versions
)```

osm_bbox_objects

osmch <- osmchange_modify(obj)
osmch

## End(Not run)

\begin{tabular}{ll}
\hline
\textbf{osm_api_versions} & \textit{Available API versions} \\
\hline
\end{tabular}

\textbf{Description}

Available API versions

\textbf{Usage}

\begin{verbatim}
  osm_api_versions()
\end{verbatim}

\textbf{Value}

A character vector with the supported versions

\textbf{See Also}

Other API functions: authenticate_osmapi(), osm_capabilities(), osm_permissions(), set_osmapi_connection()

\textbf{Examples}

\begin{verbatim}
  osm_api_versions()
\end{verbatim}

\begin{tabular}{ll}
\hline
\textbf{osm_bbox_objects} & \textit{Retrieve map data by bounding box} \\
\hline
\end{tabular}

\textbf{Description}

The following command returns:

- All nodes that are inside a given bounding box and any relations that reference them.
- All ways that reference at least one node that is inside a given bounding box, any relations that reference them [the ways], and any nodes outside the bounding box that the ways may reference.
- All relations that reference one of the nodes, ways or relations included due to the above rules.
  (Does "not" apply recursively, see explanation below.)

\textbf{Usage}

\begin{verbatim}
  osm_bbox_objects(bbox, format = c("R", "xml", "json"), tags_in_columns = FALSE)
\end{verbatim}
Arguments

bbox Coordinates for the area to retrieve the map data from (left, bottom, right, top).
format Format of the output. Can be "R" (default), "xml", or "json".
tags_in_columns If FALSE (default), the tags of the objects are saved in a single list column tags containing a data.frame for each OSM object with the keys and values. If TRUE, add a column for each key. Ignored if format != "R".

Details

Note that, while this command returns those relations that reference the aforementioned nodes and ways, the reverse is not true: it does not (necessarily) return all of the nodes and ways that are referenced by these relations. This prevents unreasonably-large result sets. For example, imagine the case where:

- There is a relation named "England" that references every node in England.
- The nodes, ways, and relations are retrieved for a bounding box that covers a small portion of England. While the result would include the nodes, ways, and relations as specified by the rules for the command, including the "England" relation, it would (fortuitously) not include every node and way in England. If desired, the nodes and ways referenced by the "England" relation could be retrieved by their respective IDs.

Also note that ways which intersect the bounding box but have no nodes within the bounding box will not be returned.

Value

If format = "R", returns a data frame with one OSM object per row. If format = "xml", returns a xml2::xml_document following the OSM_XML format. If format = "json", returns a list with a json structure following the OSM_JSON format.

Note

For downloading data for purposes other than editing or exploring the history of the objects, perhaps is better to use the Overpass API. A similar function to download OSM objects using Overpass, is implemented in the osmdata function opq().

See Also

Other get OSM objects’ functions: osm_get_objects(), osm_history_object(), osm_relations_object(), osm_ways_node(), osmapi_objects()

Examples

map_data <- osm_bbox_objects(bbox = c(1.8366775, 41.8336843, 1.8379971, 41.8344537))
## bbox as a character value also works (bbox = "1.8366775,41.8336843,1.8379971,41.8344537").
map_data
Description

Provide information about the capabilities and limitations of the current API.

Usage

```python
osm_capabilities()
```

Details

API:

- `version minimum` and `maximum` are the API call versions that the server will accept.
- `area maximum` is the maximum area in square degrees that can be queried by API calls.
- `tracepoints per_page` is the maximum number of points in a single GPS trace. (Possibly incorrect)
- `waynodes maximum` is the maximum number of nodes that a way may contain.
- `relationmember maximum` is the maximum number of members that a relation may contain.
- `changesets maximum_elements` is the maximum number of combined nodes, ways and relations that can be contained in a changeset.
- `changesets default_query_limit` and `maximum_query_limit` are the default and maximum values of the limit parameter of `osm_query_changesets()`.
- `notes default_query_limit` and `maximum_query_limit` are the default and maximum values of the limit parameter of notes bounding box queries (`osm_read_bbox_notes()`) and search (`osm_search_notes()`).
- The `status` element returns either `online`, `readonly` or `offline` for each of the database, API and GPX API. The `database` field is informational, and the `API`/`GPX-API` fields indicate whether a client should expect read and write requests to work (`online`), only read requests to work (`readonly`) or no requests to work (`offline`).

Policy:

- Imagery blacklist lists all aerial and map sources, which are not permitted for OSM usage due to copyright. Editors must not show these resources as background layer.

Value

A list with the API capabilities and policies.

See Also

Other API functions: `authenticate_osmapi()`, `osm_api_versions()`, `osm_permissions()`, `set_osmapi_connection()`
Examples

osm_capabilities()

**osm_close_note**  
*Close or reopen notes*

**Description**

Requires authentication.

**Usage**

```r
osm_close_note(note_id)

osm_reopen_note(note_id)
```

**Arguments**

- **note_id**: Note ids represented by a numeric or character vector.

**Value**

Returns a data frame with the closed map notes (same format as `osm_get_notes()` with format = "R").

**Functions**

- `osm_close_note()`: Close notes as fixed.
- `osm_reopen_note()`: Reopen closed notes.

**See Also**

Other edit notes’ functions: `osm_create_comment_note()`, `osm_create_note()`, `osm_delete_note()`

**Examples**

```r
## Not run:
set_osmapi_connection("testing") # use the testing server
note <- osm_create_note(lat = 41.38373, lon = 2.18233, text = "Testing osmapiR")
closed_note <- osm_close_note(note$id)
closed_note
reopened_note <- osm_reopen_note(note$id)
reopened_note
closed_note <- osm_close_note(note$id) # leave it closed

## End(Not run)
```
osm_comment_changeset_discussion

*Comment a changeset*

**Description**

Add a comment to a changeset and subscribe to the discussion. The changeset must be closed. Requires authentication.

**Usage**

```r
osm_comment_changeset_discussion(changeset_id, comment)
```

**Arguments**

- `changeset_id` The id of the changeset to comment represented by a numeric or a character value.
- `comment` The text of the comment to post.

**Value**

Returns a data frame with the changeset (same format as `osm_get_changesets()` with `format = "R"`).

**See Also**

Other changeset discussion’s functions: `osm_hide_comment_changeset_discussion()`, `osm_subscribe_changeset_discussion()`.

**Examples**

```r
## Not run:
set_osmapi_connection("testing") # use the testing server
closed_changeset <- osm_get_changesets(300626)
updated_changeset <- osm_comment_changeset_discussion(
  changeset_id = closed_changeset$id,
  comment = "A new comment to test osmapiR"
)
updated_changeset

## End(Not run)
```
osm_create_changeset

Create, update, or close a changeset

Description

Create, update, or close a changeset

Usage

```r
osm_create_changeset(
  comment,
  ..., 
  created_by = paste("osmapiR", getOption("osmapir.osmapir_version"),
                     ,verbose = FALSE
)

osm_update_changeset(
  changeset_id,
  comment,
  ..., 
  created_by = paste("osmapiR", getOption("osmapir.osmapir_version"),
                     ,verbose = FALSE
)

osm_close_changeset(changeset_id)
```

Arguments

- **comment**: Tag comment is mandatory.
- **...**: Arbitrary tags to add to the changeset as named parameters (key = "value").
- **created_by**: Tag with the client data. By default, `osmapiR` x.y.z.
- **verbose**: If TRUE, print the tags of the new changeset.
- **changeset_id**: The id of the changeset to update. The user issuing this API call has to be the same that created the changeset.

Details

See [https://wiki.openstreetmap.org/wiki/Changeset](https://wiki.openstreetmap.org/wiki/Changeset) for details and the most common changeset’s tags.

When updating a changeset, unchanged tags have to be repeated in order to not be deleted.

Value

The ID of the newly created changeset or a `data.frame` inheriting `osmapi_changesets` with the details of the updated changeset.

Nothing is returned upon successful closing of a changeset.
osm_create_comment_note

Description

Add a new comment to an existing note. Requires authentication.

Usage

osm_create_comment_note(note_id, text)

Arguments

- note_id: Note id represented by a numeric or a character value.
- text: The comment as arbitrary text.

Functions

- osm_create_changeset(): Open a new changeset for editing.
- osm_update_changeset(): Update the tags of an open changeset.
- osm_close_changeset(): Close a changeset. A changeset may already have been closed without the owner issuing this API call. In this case an error code is returned.

See Also

Other edit changeset's functions: osm_diff_upload_changeset()

Examples

```r
## Not run:
set_osmapi_connection("testing") # use the testing server

chset_id <- osm_create_changeset(
  comment = "Describe the changeset",
  source = "GPS;survey",
  hashtags = "#testing;#osmapiR"
)

chaset <- osm_read_changeset(changeset_id = chset_id)
chaset

upd_chaset <- osm_update_changeset(
  changeset_id = chset_id,
  comment = "Improved description of the changeset",
  hashtags = "#testing;#osmapiR"
)
upd_chaset

## End(Not run)
```
Value

Returns a data frame with the map note and the new comment (same format as \texttt{osm_get_notes()} with format = "R").

See Also

Other edit notes' functions: \texttt{osm_close_note()}, \texttt{osm_create_note()}, \texttt{osm_delete_note()}

Examples

```r
## Not run:
set_osmapi_connection("testing") # use the testing server
note <- osm_get_notes(53726)
updated_note <- osm_create_comment_note(note$id, text = "A new comment to the note")
updated_note
## End(Not run)
```

---

\textbf{osm_create_gpx} \hspace{1cm} \textit{Create GPS trace}

Description

Use this to upload a GPX file or archive of GPX files. Requires authentication.

Usage

```r
osm_create_gpx(
  file,
  description,
  tags,
  visibility = c("private", "public", "trackable", "identifiable")
)
```

Arguments

- \textit{file} The GPX file path containing the track points.
- \textit{description} The trace description. Cannot be empty.
- \textit{tags} A string containing tags for the trace. Can be empty.
- \textit{visibility} One of the following: private, public, trackable, identifiable. For explanations see OSM trace upload page or Visibility of GPS traces).

Details

Note that for successful processing, the file must contain trackpoints (<trkpt>), not only waypoints, and the trackpoints must have a valid timestamp. Since the file is processed asynchronously, the call will complete successfully even if the file cannot be processed. The file may also be a .tar, .tar.gz or .zip containing multiple gpx files, although it will appear as a single entry in the upload log.
Value

A number representing the ID of the new gpx.

See Also

Other edit GPS traces' functions: osm_delete_gpx(), osm_update_gpx()

Examples

vignette("how_to_edit_gps_traces", package = "osmapiR")

---

osm_create_note  Create a new note

Description

Create a new note

Usage

osm_create_note(lat, lon, text, authenticate = TRUE)

Arguments

lat  Specifies the latitude in decimal degrees of the note.
lon  Specifies the longitude in decimal degrees of the note.
text  A text field with arbitrary text containing the note.
authenticate  If TRUE (default), the note is authored by the logged user. Otherwise, anonymous note.

Details

If the request is made as an authenticated user, the note is associated to that user account. If the OAuth access token used does not have the allow_write_notes permission, it is created as an anonymous note instead.

Value

Returns a data frame with the map note (same format as osm_get_notes() with format = "R").

See Also

Other edit notes' functions: osm_close_note(), osm_create_comment_note(), osm_delete_note()
osm_create_object

Examples

```r
## Not run:
set_osmapi_connection("testing") # use the testing server
new_note <- osm_create_note(lat = 41.38373, lon = 2.18233, text = "Testing osmapiR")
new_note

## End(Not run)
```

osm_create_object **Create an OSM object**

Description

Creates a new element in an open changeset as specified.

Usage

```r
osm_create_object(x, changeset_id)
```

Arguments

- **x** The new object data. Can be the path to an xml file, a `xml2::xml_document` or a data.frame inheriting or following the structure of an `osmapi_objects` object.
- **changeset_id** The ID of an open changeset where to create the object. If missing, `x` should define the changeset ID, otherwise it will be overwritten with `changeset_id`. Ignored if `x` is a path.

Details

If `x` is a data.frame, the columns `type`, `changeset`, `tags` must be present + column `members` for ways and relations + `lat` and `lon` for nodes. For the xml format, see the OSM wiki.

If multiple elements are provided only the first is created. The rest is discarded.

Value

The ID of the newly created OSM object.

Note

- This updates the bounding box of the changeset.
- The `role` attribute for relations is optional. An empty string is the default.
- To avoid performance issues when uploading multiple objects, the use of the `osm_diff_upload_changeset()` is highly recommended.
- The version of the created object will be 1.
osm_delete_gpx

See Also

Other edit OSM objects' functions: osm_delete_object(), osm_update_object()

Examples

vignette("how_to_edit_osm", package = "osmapiR")

osm_delete_gpx

Delete GPS traces

Description

Use this to delete GPX files. Only usable by the owner account. Requires authentication.

Usage

osm_delete_gpx(gpx_id)

Arguments

gpx_id The track ids represented by a numeric or a character vector.

Value

Returns NULL invisibly.

See Also

Other edit GPS traces’ functions: osm_create_gpx(), osm_update_gpx()

Examples

vignette("how_to_edit_gps_traces", package = "osmapiR")
Description

Hide (delete) notes. This request needs to be done as an authenticated user with moderator role.

Usage

`osm_delete_note(note_id, text)`

Arguments

- `note_id`: Note ids represented by a numeric or a character vector.
- `text`: A non-mandatory comment as text.

Details

Use `osm_reopen_note()` to make the note visible again.

Value

Returns a data frame with the hided map notes (same format as `osm_get_notes()` with format = "R").

See Also

Other edit notes' functions: `osm_close_note()`, `osm_create_comment_note()`, `osm_create_note()`

Other functions for moderators: `osm_hide_comment_changeset_discussion()`, `osm_redaction_object()`

Examples

```r
## Not run:
set_osmapi_connection("testing") # use the testing server  
note <- osm_create_note(lat = "40.7327375", lon = "0.1702526", text = "Test note to delete.")

del_note <- osm_delete_note(note_id = note$id, text = "Hide note")
del_note

## End(Not run)
```
osm_delete_object  Delete an OSM object

Description
Expects a valid XML representation of the element to be deleted.

Usage

```r
osm_delete_object(x, changeset_id)
```

Arguments

- `x` The object data. Can be the path of an xml file, a `xml2::xml_document` or a `data.frame` inheriting or following the structure of an `osmapi_objects` object.
- `changeset_id` The ID of an open changeset where to create the object. If missing, `x` should define the changeset ID, otherwise it will be overwritten with `changeset_id`. Ignored if `x` is a path.

Details
The version must match the version of the element you downloaded and the changeset must match the id of an open changeset owned by the current authenticated user. It is allowed, but not necessary, to have tags on the element except for lat/long which are required for nodes, without lat+lon the server gives 400 Bad request.

If `x` is a data.frame, the columns `type`, `id`, `version` and `changeset` must be present + lat and lon for nodes. For the xml format, see the OSM wiki.

If multiple elements are provided only the first is deleted. The rest is discarded.

Value
Returns the new version number of the object.

Note

- This updates the bounding box of the changeset.
- To avoid performance issues when deleting multiple objects, the use of the `osm_diff_upload_changeset()` is highly recommended. This is also the only way to ensure that multiple objects are updated in a single database transaction.

See Also
Other edit OSM objects' functions: `osm_create_object()`, `osm_update_object()`

Examples

```r
vignette("how_to_edit_osm", package = "osmapiR")
```
osm_details_logged_user

Details of the logged-in user

Description

You can get the home location, the display name of the user and other details.

Usage

osm_details_logged_user(format = c("R", "xml", "json"))

Arguments

format Format of the output. Can be "R" (default), "xml", or "json".

Value

If format = "R", returns a list with the user details.

format = "xml":

Returns a xml2::xml_document with the following format:

<osm version="0.6" generator="OpenStreetMap server">
  <user display_name="Max Muster" account_created="2006-07-21T19:28:26Z" id="1234">
    <contributor-terms agreed="true" pd="true"/>
    <img href="https://www.openstreetmap.org/attachments/users/images/000/000/1234/original/someLongURLOrOther.JPG"/>
    <roles></roles>
    <changesets count="4182"/>
    <traces count="513"/>
    <blocks>
      <received count="0" active="0"/>
    </blocks>
    <home lat="49.4733718952806" lon="8.89285988577866" zoom="3"/>
    <description>The description of your profile</description>
    <languages>
      <lang>de-DE</lang>
      <lang>de</lang>
      <lang>en-US</lang>
      <lang>en</lang>
    </languages>
    <messages>
      <received count="1" unread="0"/>
      <sent count="0"/>
    </messages>
  </user>
</osm>
format = "json":

{
  "version": "0.6",
  "generator": "OpenStreetMap server",
  "user": {
    "id": 1234,
    "display_name": "Max Muster",
    "account_created": "2006-07-21T19:28:26Z",
    "description": "The description of your profile",
    "contributor_terms": {"agreed": True, "pd": True},
    "img": {"href": "https://www.openstreetmap.org/attachments/users/images/000/000/1234/original/someLongURLOrOther.JPG"},
    "roles": [],
    "changesets": {"count": 4182},
    "traces": {"count": 513},
    "blocks": {"received": {"count": 0, "active": 0}},
    "home": {"lat": 49.4733718952806, "lon": 8.89285988577866, "zoom": 3},
    "languages": ["de-DE", "de", "en-US", "en"],
    "messages": {"received": {"count": 1, "unread": 0},
                 "sent": {"count": 0}}
  }
}

See Also

Other users’ functions: osm_get_preferences_user(), osm.get_user_details()

Examples

```
## Not run:
usr_details <- osm_details_logged_user()
usr_details

## End(Not run)
```
Arguments

changeset_id  The ID of the changeset this diff belongs to. The user issuing this API call has to be the same that created the changeset.

osmcha  The OsmChange data. Can be the path of an OsmChange file, a xml2::xml_document or an osmapi_OsmChange object (see osmchange_*() functions).

format  Format of the output. Can be "R" (default) or "xml".

Details

To upload an OSC file it has to conform to the OsmChange specification with the following differences:

- each element must carry a changeset and a version attribute (xml) / column (data.frame), except when you are creating an element where the version is not required as the server sets that for you. The changeset must be the same as the changeset ID being uploaded to.
- a <delete> block in the OsmChange document may have an if-unused attribute (the value of which is ignored) (action_type column with delete if-unused for data.frames). If this attribute is present, then the delete operation(s) in this block are conditional and will only be executed if the object to be deleted is not used by another object. Without the if-unused, such a situation would lead to an error, and the whole diff upload would fail. Setting the attribute will also cause deletions of already deleted objects to not generate an error.
- OsmChange documents generally have user and uid attributes on each element. These are not required in the document uploaded to the API.

Value

If a diff is successfully applied and format = "R", it returns a data frame with one row for each edited object. For format = "xml", a xml2::xml_document is returned in the following format:

<diffResult generator="OpenStreetMap Server" version="0.6">  
<node|way|relation old_id="#" new_id="#" new_version="#"/>  
...  
</diffResult>

with one element for every object in the upload.

Note that this can be counter-intuitive when the same element has appeared multiple times in the input then it will appear multiple times in the output.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>create</th>
<th>modify</th>
<th>delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>old_id</td>
<td>same as uploaded element</td>
<td>same as uploaded element</td>
<td>same as uploaded element</td>
</tr>
<tr>
<td>new_id</td>
<td>new ID</td>
<td>new ID &quot;or&quot; same as uploaded</td>
<td>not present</td>
</tr>
<tr>
<td>new_version</td>
<td>new version</td>
<td>new version</td>
<td>not present</td>
</tr>
</tbody>
</table>
Note

- Processing stops at the first error, so if there are multiple conflicts in one diff upload, only the first problem is reported.
- Refer to `osm_capabilities()` -> `changesets$maximum_elements` for the maximum number of changes permitted in a changeset.
- There is currently no limit in the diff size on the Rails port. CGImap limits diff size to 50MB (uncompressed size).
- Forward referencing of placeholder ids is not permitted and will be rejected by the API.

See Also

Other edit changeset’s functions: `osm_create_changeset()`

Other OsmChange’s functions: `osm_download_changeset()`, `osmchange_create()`, `osmchange_delete()`, `osmchange_modify()`

Examples

vignette("how_to_edit_osm", package = "osmapiR")

---

**osm_download_changeset**

*Download a changeset in OsmChange format*

Description

Returns the **OsmChange** document describing all changes associated with the changeset.

Usage

```r
osm_download_changeset(changeset_id, format = c("R", "osc", "xml"))
```

Arguments

- **changeset_id**: The id of the changeset represented by a numeric or a character value for which the OsmChange is requested.
- **format**: Format of the output. Can be "R" (default) or "osc" ("xml" is a synonym for "osc").

Details

- The result of calling this may change as long as the changeset is open.
- The elements in the OsmChange are sorted by timestamp and version number.
- There is `osm_get_changesets()` to get only information about the changeset itself.
Value

If `format = "R"`, returns a data frame with one row for each edit action in the changeset. If `format = "osc"`, returns a `xml2::xml_document` in the `OsmChange` format.

See Also

Other get changesets’ functions: `osm_get_changesets()`, `osm_query_changesets()`
Other `OsmChange`’s functions: `osm_diff_upload_changeset()`, `osmchange_create()`, `osmchange_delete()`, `osmchange_modify()`

Examples

```r
## Not run:
chaset <- osm_download_changeset(changeset_id = 137003062)
chaset
## End(Not run)
```

---

### `osm_feed_notes`

**RSS Feed of notes in a bbox**

Description

RSS Feed of notes in a bbox

Usage

```r
osm_feed_notes(bbox)
```

Arguments

- `bbox`: Coordinates for the area to retrieve the notes from (left, bottom, right, top). Floating point numbers in degrees, expressing a valid bounding box, not larger than the configured size limit, 25 square degrees (see `osm_capabilities()`$note_area and this line in settings for the current value), not overlapping the dateline.

Value

Returns a `xml2::xml_document` in the RSS format.

See Also

Other get notes’ functions: `osm_get_notes()`, `osm_read_bbox_notes()`, `osm_search_notes()`

Examples

```r
feed_notes <- osm_feed_notes(bbox = c(0.8205414, 40.6686604, 0.8857727, 40.7493377))
## bbox as a character value also works (bbox = "0.8205414,40.6686604,0.8857727,40.7493377").
feed_notes
```
osm_get_changesets  Get changesets

Description

Returns the changesets with the given changeset_id.

Usage

osm_get_changesets(
  changeset_id,
  include_discussion = FALSE,
  format = c("R", "xml", "json"),
  tags_in_columns = FALSE
)

Arguments

changeset_id  A vector with ids of changesets to retrieve represented by a numeric or a character values.
include_discussion  Indicates whether the result should contain the changeset discussion or not.
format  Format of the output. Can be "R" (default), "xml", or "json".
tags_in_columns  If FALSE (default), the tags of the changesets are saved in a single list column containing a data.frame for each changeset with the keys and values. If TRUE, add a column for each key. Ignored if format != "R".

Details

- The uid might not be available for changesets auto generated by the API v0.5 to API v0.6 transition
- The bounding box attributes will be missing for an empty changeset.
- The changeset bounding box is a rectangle that contains the bounding boxes of all objects changed in this changeset. It is not necessarily the smallest possible rectangle that does so.
- This API call only returns information about the changeset itself but not the actual changes made to elements in this changeset. To access this information use osm_download_changeset().

Value

If format = "R", returns a data frame with one OSM changeset per row.

format = "xml":
Returns a xml2::xml_document with the following format:
osm_get_changesets

<osm>
<changeset id="10" created_at="2008-11-08T19:07:39+01:00" open="true" user="fred" uid="123" min_lon="7.0191821" min_lat="49.2785426" max_lon="7.0197485" max_lat="49.2793101" comments_count="3" changes_count="10">
<tag k="created_by" v="JOSM 1.61"/>
<tag k="comment" v="Just adding some streetnames"/>
...
<discussion>
<comment date="2015-01-01T18:56:48Z" uid="1841" user="metaodi">
<text>Did you verify those street names?</text>
</comment>
<comment date="2015-01-01T18:58:03Z" uid="123" user="fred">
<text>sure!</text>
</comment>
...
</discussion>
</changeset>
<changeset>
...
</changeset>
</osm>

format = "json";
Returns a list with the following json structure:

{
  "version": "0.6",
  "elements": [
    {
      "type": "changeset",
      "id": 10,
      "created_at": "2005-05-01T16:09:37Z",
      "closed_at": "2005-05-01T17:16:44Z",
      "open": false,
      "user": "Petter Reinholdtsen",
      "uid": 24,
      "minlat": 59.9513092,
      "minlon": 10.7719727,
      "maxlat": 59.9561501,
      "maxlon": 10.7994537,
      "comments_count": 1,
      "changes_count": 10,
      "discussion": [{"date": "2022-03-22T20:58:30Z", "uid": 15079200, "user": "Ethan White of Cheriton", ...}]
    }, ...
  ]
}

See Also
Other get changesets' functions: osm_download_changeset(), osm_query_changesets()

Examples

## Not run:
chaset <- osm_get_changesets(changeset_id = 137595351, include_discussion = TRUE)
chaset
chaset$discussion

## End(Not run)

---

### osm_get_data_gpx

**Download GPS Track Data**

**Description**

Use this to download the full GPX file. Available without authentication if the file is marked public. Otherwise only usable by the owner account and requires authentication.

**Usage**

```r
osm_get_data_gpx(gpx_id, format)
```

**Arguments**

- `gpx_id` The track id represented by a numeric or a character value.
- `format` Format of the output. If missing (default), the response will be the exact file that was uploaded. If "R", a data frame. If "gpx", the response will always be a GPX format file. If "xml", a "xml" file in an undocumented format.

**Value**

If missing `format`, returns a `xml2::xml_document` with the original file data. If `format = "R"`, returns a data frame with one point per row. If `format = "gpx"`, returns a `xml2::xml_document` in the GPX format. If `format = "xml"`, returns a `xml2::xml_document` in an undocumented format.

**Note**

If you request refers to a multi-file archive the response when you force gpx or xml format will consist of a non-standard simple concatenation of the files.

**See Also**

Other get GPS’ functions: `osm_get_gpx_metadata()`, `osm_get_points_gps()`, `osm_list_gpxs()`

**Examples**

```r
## Not run:
trk_data <- osm_get_data_gpx(gpx_id = 3498170, format = "R")
trk_data

## End(Not run)
```
osm_get_gpx_metadata  Download GPS Track Metadata

Description

Use this to access the metadata about GPX files. Available without authentication if the file is marked public. Otherwise only usable by the owner account and requires authentication.

Usage

osm_get_gpx_metadata(gpx_id, format = c("R", "xml"))

Arguments

gpx_id  A vector of track ids represented by a numeric or a character value.
format   Format of the output. Can be "R" (default) or "xml".

Value

If format = "R", returns a data frame with one trace per row. If format = "xml", returns a xml2::xml_document with the following format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="OpenStreetMap server">
<gpx_file id="836619" name="track.gpx" lat="52.0194" lon="8.51807" uid="1234" user="Hartmut Holzgraeffe" visibility="public" pending="false" timestamp="2010-10-09T09:24:19Z">
<description>PHP upload test</description>
<tag>test</tag>
<tag>php</tag>
</gpx_file>
<gpx_file>
...
</gpx_file>
</osm>
```

See Also

Other get GPS' functions: osm_get_data_gpx(), osm_get_points_gps(), osm_list_gpxs()

Examples

```r
## Not run:
trk_meta <- osm_get_gpx_metadata(gpx_id = 3498170)
trk_meta

## End(Not run)
```
osm_get_notes

Get notes

Description

Returns the existing note with the given ID.

Usage

osm_get_notes(note_id, format = c("R", "xml", "rss", "json", "gpx"))

Arguments

- **note_id**: Note id represented by a numeric or a character value.
- **format**: Format of the output. Can be "R" (default), "xml", "rss", "json" or "gpx".

Value

If format = "R", returns a data frame with one map note per row.

- **format = "xml"**:

  Returns a `xml2::xml_document` with the following format:

  ```xml
  <?xml version="1.0" encoding="UTF-8"?>
  <osm version="0.6" generator="OpenStreetMap server" copyright="OpenStreetMap and contributors" attribution="https://www.openstreetmap.org/copyright" license="https://opendatacommons.org/licenses/odbl/1-0/">
  <note lon="0.1000000" lat="51.0000000">
    <id>16659</id>
    <url>https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659</url>
    <comment_url>https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/comment</comment_url>
    <close_url>https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/close</close_url>
    <date_created>2019-06-15 08:26:04 UTC</date_created>
    <status>open</status>
    <comments>
      <comment>
        <date>2019-06-15 08:26:04 UTC</date>
        <uid>1234</uid>
        <user>userName</user>
        <user_url>https://master.apis.dev.openstreetmap.org/user/userName</user_url>
        <action>opened</action>
        <text>ThisIsANote</text>
        <html>&lt;p&gt;ThisIsANote&lt;/p&gt;</html>
      </comment>
      ...
    </comments>
  </note>
  ...
</osm>
osm_get_objects

format = "json":
Returns a list with the following json structure:

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": "{"type": "Point", "coordinates": [0.100000, 51.000000]},
      "properties": {
        "id": 16659,
        "url": "https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659.json",
        "comment_url": "https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/comment.json",
        "close_url": "https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/close.json",
        "date_created": "2019-06-15 08:26:04 UTC",
        "status": "open",
        "comments": [
          {"date": "2019-06-15 08:26:04 UTC", "uid": 1234, "user": "userName", "user_url": "https://master.api.dev.openstreetmap.org/user/userName"}
        ]
      }
    }
  ]
}

format = "rss" & format = "gpx":
For format in "rss", and "gpx", a xml2::xml_document with the corresponding format.

See Also

Other get notes’ functions: osm_feed_notes(), osm_read_bbox_notes(), osm_search_notes()

Examples

## Not run:
```
note <- osm_get_notes(note_id = "2067786")
note
```
## End(Not run)
Usage

```r
osm_get_objects(
  osm_type,
  osm_id,
  version,
  full_objects = FALSE,
  format = c("R", "xml", "json"),
  tags_in_columns = FALSE
)
```

Arguments

- **osm_type**
  A vector with the type of the objects ("node", "way" or "relation"). Recycled if it has a different length than `osm_id`.
- **osm_id**
  Object ids represented by a numeric or a character vector.
- **version**
  An optional vector with the version number for each object. If missing, the last version will be retrieved. Recycled if it has different length than `osm_id`.
- **full_objects**
  If `TRUE`, retrieves all other objects referenced by ways or relations. Not compatible with `version`.
- **format**
  Format of the output. Can be "R" (default), "xml", or "json".
- **tags_in_columns**
  If `FALSE` (default), the tags of the objects are saved in a single list column `tags` containing a data.frame for each OSM object with the keys and values. If `TRUE`, add a column for each key. Ignored if `format != "R"`.

Details

`full_objects = TRUE` does not support specifying `version`. For ways, `full_objects = TRUE` implies that it will return the way specified plus all nodes referenced by the way. For a relation, it will return the following:

- The relation itself
- All nodes, ways, and relations that are members of the relation
- Plus all nodes used by ways from the previous step
- The same recursive logic is not applied to relations. This means: If relation r1 contains way w1 and relation r2, and w1 contains nodes n1 and n2, and r2 contains node n3, then a "full" request for r1 will give you r1, r2, w1, n1, and n2. Not n3.

Value

If `format = "R"`, returns a data frame with one OSM object per row. If `format = "xml"`, returns a `xml2::xml_document` following the `OSM_XML` format. If `format = "json"`, returns a list with a json structure following the `OSM_JSON` format.

Objects are sorted in the same order than `osm_id` except for `full_objects = TRUE`, where the nodes comes first, then ways, and relations at the end as specified by `OSM_XML format`. 
Note

For downloading data for purposes other than editing or exploring the history of the objects, perhaps is better to use the Overpass API. A similar function to download OSM objects by type and id using Overpass, is implemented in the `osmdata` function `opq_osm_id()`.

See Also

Other get OSM objects' functions: `osm_bbox_objects()`, `osm_history_object()`, `osm_relations_object()`, `osm_ways_node()`, `osmapi_objects()`

Examples

```r
## Not run:
obj <- osm_get_objects(
  osm_type = c("node", "way", "way", "relation", "relation", "node"),
  osm_id = c("35308286", "13073736", "235744929", "40581", "341530", "1935675367"),
  version = c(1, 3, 2, 5, 7, 1)
)
obj

## End(Not run)
```

---

`osm_get_points_gps`  
*Get GPS Points*

Description

Use this to retrieve the GPS track points that are inside a given bounding box (formatted in a GPX format).

Usage

```r
osm_get_points_gps(bbox, page_number = 0, format = c("R", "gpx"))
```

Arguments

- `bbox` Coordinates for the area to retrieve the notes from (left, bottom, right, top). Floating point numbers in degrees, expressing a valid bounding box. The maximal width (right - left) and height (top - bottom) of the bounding box is 0.25 degree.
- `page_number` Specifies which groups of 5,000 points, or page, to return. The API call does not return more than 5,000 points at a time. In order to retrieve all of the points for a bounding box, set page_number = -1. When this parameter is 0 (zero), the command returns the first 5,000 points; when it is 1, the command returns points 5,001–10,000, etc. A vector is also valid (e.g. `0:2` to get the first 3 pages).
- `format` Format of the output. Can be "R" (default) or "gpx".
**Value**

If `format = "R"`, returns a list of data frames with the points for each trace. For public traces, the data frame include the attributes `name`, `desc` and `url`.

`format = "gpx"`:

Returns a `xml2::xml_document` with the following format:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.0" creator="OpenStreetMap.org" xmlns="http://www.topografix.com/GPX/1/0">
  <trk>
    <name>20190626.gpx</name>
    <desc>Footpaths near Blackweir Pond, Epping Forest</desc>
    <url>https://api.openstreetmap.org/user/John%20Leeming/traces/3031013</url>
    <trkseg>
      <trkpt lat="51.6616100" lon="-0.0534560">
        <time>2019-06-26T14:27:58Z</time>
      </trkpt>
      ...
    </trkseg>
    ...
  </trk>
  ...
</gpx>
```

- This response is NOT wrapped in an OSM xml parent element.
- The file format is GPX Version 1.0 which is not the current version. Verify that your tools support it.

**Note**

In violation of the [GPX standard](http://www.topografix.com/GPX/1/0) when downloading public GPX traces through the API, all way-points of non-trackable traces are randomized (or rather sorted by lat/lon) and delivered as one trackSegment for privacy reasons. Trackable traces are delivered, sorted by descending upload time, before the waypoints of non-trackable traces.

Private traces without `name`, `desc` and `url` can be separated in different items in the result if they get split due to server pagination. Public traces are united using matching URL.

**See Also**

Other get GPS' functions: `osm_get_data_gpx()`, `osm_get_gpx_metadata()`, `osm_list_gpxs()`

**Examples**

```r
pts_gps <- osm_get_points_gps(bbox = c(-0.3667545, 40.2153246, -0.3354263, 40.2364915))
## bbox as a character value also works(bbox = "-0.3667545,40.2153246,-0.3354263,40.2364915").
pts_gps

## get attributes
lapply(pts_gps, function(x) attributes(x)[c("name", "desc", "url")])
```
osm_get_preferences_user

Get or set preferences for the logged-in user

Description

Get or set preferences for the logged-in user

Usage

osm_get_preferences_user(key, format = c("R", "xml", "json"))

osm_set_preferences_user(key, value, all_prefs)

Arguments

key
Returns a string with this preference’s value. If missing, return all preferences.

format
Format of the output. Can be "R" (default), "xml", or "json". Only relevant when key is missing.

value
A string with the preference value to set for key. If NULL, deletes the key preference.

all_prefs
A data.frame, xml_document or a json list following the format returned by osm_get_preferences_user(). Also, a path to an xml file describing the user preferences. All existing preferences are replaced by the newly uploaded set.

Details

The sizes of the key and value are limited to 255 characters.

The OSM server supports storing arbitrary user preferences. This can be used by editors, for example, to offer the same configuration wherever the user logs in, instead of a locally-stored configuration. For an overview of applications using the preferences-API and which key-schemes they use, see this wiki page.

Value

If format = "R", returns a data frame with key and value columns of the user preferences.

format = "xml":
Returns a xml2::xml_document with the following format:

<osm version="0.6" generator="OpenStreetMap server">
<preferences>
<preference k="somekey" v="somevalue" />
...
</preferences>
</osm>
Returns a list with the following json structure:

```json
{
    "version": "0.6",
    "generator": "OpenStreetMap server",
    "preferences": {"somekey": "somevalue, ...}
}
```

Set preferences:
Nothing is returned upon successful setting of user preferences.

See Also
Other users' functions: `osm_details_logged_user()`, `osm_get_user_details()`

Examples
```r
## Not run:
prefs_ori <- osm_get_preferences_user()
prefs_ori

osm_set_preferences_user(key = "osmapiR-test", value = "good!")
osm_get_preferences_user(key = "osmapiR-test")

osm_set_preferences_user(key = "osmapiR-test", value = NULL) # Delete pref

## Restore all preferences
osm_set_preferences_user(all_prefs = prefs_ori)

## End(Not run)
```

Description
Details of users

Usage
```
osm_get_user_details(user_id, format = c("R", "xml", "json"))
```

Arguments
- **user_id**: The ids of the users to retrieve the details for, represented by a numeric or a character value (not the display names).
- **format**: Format of the output. Can be "R" (default), "xml", or "json".
**osm_get_user_details**

**Value**

For users not found, the result is empty. If **format = "R"**, returns a data frame with one user per row.

**format = "xml"**:

Returns a `xml2::xml_document` with the following format:

```xml
<osm version="0.6" generator="OpenStreetMap server">
  <user id="12023" display_name="jbpbis" account_created="2007-08-16T01:35:56Z">
    <description/>
    <contributor-terms agreed="false"/>
    <roles/>
    <changesets count="1"/>
    <traces count="0"/>
    <blocks>
      <received count="0" active="0"/>
    </blocks>
  </user>
  <user id="210447" display_name="siebh" account_created="2009-12-20T10:11:42Z">
    <description/>
    <contributor-terms agreed="true"/>
    <roles/>
    <changesets count="267"/>
    <traces count="1"/>
    <blocks>
      <received count="0" active="0"/>
    </blocks>
  </user>
</osm>
```

**format = "json"**:

Returns a list with the following json structure:

```json
{
  "version": "0.6",
  "generator": "OpenStreetMap server",
  "users": [
    {"id": 12023, "display_name": "jbpbis", "account_created": "2007-08-16T01:35:56Z", "description": "", "roles": [], "changesets": {"count": 1}, "traces": {"count": 0}, "blocks": {"received": {"count": 0, "active": 0}}},
    {"id": 210447, "display_name": "siebh", "account_created": "2009-12-20T10:11:42Z", "description": "", "roles": [], "changesets": {"count": 267}, "traces": {"count": 1}, "blocks": {"received": {"count": 0, "active": 0}}}
  ]
}
```

**See Also**

Other users' functions: `osm_details_logged_user()`, `osm_get_preferences_user()`
Examples

```r
## Not run:
usrs <- osm_get_user_details(user_id = c(1, 24, 44, 45, 46, 48, 49, 50))
usrs
## End(Not run)
```

### osm_hide_comment_changeset_discussion

Hide or unhide a changeset comment

#### Description

This request needs to be done as an authenticated user with moderator role.

#### Usage

```r
osm_hide_comment_changeset_discussion(comment_id)
osm_unhide_comment_changeset_discussion(comment_id)
```

#### Arguments

- `comment_id`
  
  Note that the changeset comment id differs from the changeset id.

#### Value

Returns a data frame with the changeset (same format as `osm_get_changesets()` with format = "R").

#### Functions

- `osm_hide_comment_changeset_discussion()`: Sets visible flag on changeset comment to false.
- `osm_unhide_comment_changeset_discussion()`: Sets visible flag on changeset comment to true.

#### See Also

Other changeset discussion's functions: `osm_comment_changeset_discussion()`, `osm_subscribe_changeset_discussion()`

Other functions for moderators: `osm_delete_note()`, `osm_redaction_object()`
osm_history_object

Examples

```r
## Not run:
chdis <- osm_get_changesets("265646", include_discussion = TRUE)
hide_com <- osm_hide_comment_changeset_discussion(comment_id = chdis$discussion[[1]]$id[[1]])
unhide_com <- osm_unhide_comment_changeset_discussion(comment_id = chdis$discussion[[1]]$id[[1]])
## End(Not run)
```

---

**osm_history_object**  
*Get the history of an object*

**Description**

Retrieves all old versions of an object from OSM.

**Usage**

```r
osm_history_object(
  osm_type = c("node", "way", "relation"),
  osm_id,
  format = c("R", "xml", "json"),
  tags_in_columns = FALSE
)
```

**Arguments**

- **osm_type**  
  Object type ("node", "way" or "relation").

- **osm_id**  
  Object id represented by a numeric or a character value.

- **format**  
  Format of the output. Can be "R" (default), "xml", or "json".

- **tags_in_columns**  
  If FALSE (default), the tags of the objects are saved in a single list column tags containing a data.frame for each OSM object with the keys and values. If TRUE, add a column for each key. Ignored if format != "R".

**Value**

If format = "R", returns a data frame with a version of the OSM object per row. If format = "xml", returns a xml2::xml_document following the OSM_XML format. If format = "json", returns a list with a json structure following the OSM_JSON format.

**See Also**

Other get OSM objects’ functions: osm_bbox_objects(), osm_get_objects(), osm_relations_object(), osm_ways_node(), osmapi_objects()
osm_list_gpxs

## Not run:
```r
node <- osm_history_object(osm_type = "node", osm_id = 35308286)
node

way <- osm_history_object(osm_type = "way", osm_id = 13073736L)
way

rel <- osm_history_object(osm_type = "relation", osm_id = "40581")
rel

## End(Not run)
```

---

**osm_list_gpxs**  
*List user's GPX traces*

### Description

Use this to get a list of GPX traces owned by the authenticated user. Requires authentication.

### Usage

```r
osm_list_gpxs(format = c("R", "xml"))
```

### Arguments

- **format**  
  Format of the output. Can be "R" (default) or "xml".

### Value

If `format = "R"`, returns a data frame with one trace per row. If `format = "xml"`, returns a `xml2::xml_document` similar to `osm_get_gpx_metadata()`. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="OpenStreetMap server">
<gpx_file id="836619" name="track.gpx" lat="52.0194" lon="8.51807" uid="1234" user="Hartmut Holzgraefe" pending="false" timestamp="2010-10-09T09:24:19Z">
<description>PHP upload test</description>
<tag>test</tag>
<tag>php</tag>
</gpx_file>
<gpx_file id="836620" name="track.gpx" lat="52.1194" lon="8.61807" uid="1234" user="Hartmut Holzgraefe" pending="false" timestamp="2010-10-09T09:27:31Z">
<description>PHP upload test 2</description>
<tag>test</tag>
<tag>php</tag>
</gpx_file>
</osm>
```
osm_permissions

See Also

Other get GPS’ functions: osm_get_data_gpx(), osm_get_gpx_metadata(), osm_get_points_gpx()

Examples

```r
## Not run:
traces <- osm_list_gpxs()
traces

## End(Not run)
```

---

**osm_permissions**   
**Retrieving permissions**

Description

Returns the permissions granted to the current API connection.

Usage

```r
osm_permissions(format = c("R", "xml", "json"))
```

Arguments

- `format` Format of the output. Can be "R" (default), "xml", or "json".

Details

Currently the following permissions can appear in the result, corresponding directly to the ones used in the OAuth 1.0a application definition:

- allow_read_prefs (read user preferences)
- allow_write_prefs (modify user preferences)
- allow_write_diary (create diary entries, comments and make friends)
- allow_write_api (modify the map)
- allow_read_gpx (read private GPS traces)
- allow_write_gpx (upload GPS traces)
- allow_write_notes (modify notes)

Value

If the API client is not authorized, an empty list of permissions will be returned. Otherwise, the list will be based on the granted scopes of the logged user.
Note

For compatibility reasons, all OAuth 2.0 scopes will be prefixed by "allow_", e.g. scope "read_prefs" will be shown as permission "allow_read_prefs".

See Also

Other API functions: authenticate_osmapi(), osm_api_versions(), osm_capabilities(), set_osmapi_connection()

Examples

```r
## Not run:
perms <- osm_permissions()
perms

## End(Not run)
```

## osm_query_changesets

### Query changesets

This is an API method for querying changesets. It supports querying by different criteria.

### Usage

```r
osm_query_changesets(
  bbox,
  user,
  time,
  time_2,
  open,
  closed,
  changeset_ids,
  order = c("newest", "oldest"),
  limit =getOption("osmapir.api_capabilities")$api$changesets["default_query_limit"],
  format = c("R", "xml", "json"),
  tags_in_columns = FALSE
)
```

### Arguments

- **bbox**: Find changesets within the given bounding box coordinates \((\text{left}, \text{bottom}, \text{right}, \text{top})\).
- **user**: Find changesets by the user with the given user id (numeric) or display name (character).
- **time**: Find changesets closed after this date and time. See details for the valid formats.
find changesets that were **closed** after time and **created** before time_2. In other words, any changesets that were open **at some time** during the given time range to time_2.

**open**

If TRUE, only finds changesets that are still **open** but excludes changesets that are closed or have reached the element limit for a changeset (10,000 at the moment `osm_capabilities()$api$changesets`).

**closed**

If TRUE, only finds changesets that are **closed** or have reached the element limit.

**changeset_ids**

Finds changesets with the specified ids.

**order**

If "newest" (default), sort newest changesets first. If "oldest", reverse order.

**limit**

Specifies the maximum number of changesets returned. 100 as the default value.

**format**

Format of the output. Can be "R" (default), "xml", or "json".

**tags_in_columns**

If FALSE (default), the tags of the changesets are saved in a single list column `tags` containing a `data.frame` for each changeset with the keys and values. If TRUE, add a column for each key. Ignored if format != "R".

**Details**

Where multiple queries are given the result will be those which match all of the requirements. The contents of the returned document are the changesets and their tags. To get the full set of changes associated with a changeset, use `osm_download_changeset()` on each changeset ID individually.

Modification and extension of the basic queries above may be required to support rollback and other uses we find for changesets.

This call returns latest changesets matching criteria. The default ordering is newest first, but you can specify order = "oldest" to reverse the sort order (see **ordered by created_at** – see the current state). Reverse ordering cannot be combined with time.

The valid formats for time and time_2 parameters are anything that `Time.parse` Ruby function will parse.

**Value**

If format = "R", returns a data frame with one OSM changeset per row.

**format = "xml"**:

Returns a `xml2::xml_document` with the following format:

```xml
<osm>
  <changeset id="10" created_at="2008-11-08T19:07:39+01:00" open="true" user="fred" uid="123" min_lon="7.0191821" min_lat="49.2785426" max_lon="7.0197485" max_lat="49.2793101" comments_count="3" changes_count="10">
    <tag k="created_by" v="JOSM 1.61"/>
    <tag k="comment" v="Just adding some streetnames"/>
    ...
  <discussion>
    <comment date="2015-01-01T18:56:48Z" uid="1841" user="metaodi">Did you verify those street names?</comment>
    ...
  </discussion>
  <comment date="2015-01-01T18:58:03Z" uid="123" user="fred">
```

<text>sure!</text>
</comment>
...
</discussion>
</changeset>
</changeset>
</osm>

format = "json":

Returns a list with the following json structure:

{  
"version": "0.6",
"elements": [  
{"type": "changeset",
  "id": 10,
  "created_at": "2005-05-01T16:09:37Z",
  "closed_at": "2005-05-01T17:16:44Z",
  "open": false,
  "user": "Petter Reinholdtsen",
  "uid": 24,
  "minlat": 59.9513092,
  "minlon": 10.7719727,
  "maxlat": 59.9561501,
  "maxlon": 10.7994537,
  "comments_count": 1,
  "changes_count": 10,
  "discussion": [{"date": "2022-03-22T20:58:30Z", "uid": 15079200, "user": "Ethan White of Cheriton"}, ...]
  }, ...
]

See Also

Other get changesets’ functions: osm_download_changeset(), osm_get_changesets()

Examples

## Not run:
chst_ids <- osm_query_changesets(changeset_ids = c(137627129, 137625624))
chst_ids
chsts <- osm_query_changesets(
  bbox = c(-1.241112, 38.0294955, 8.4203171, 42.9186456),
  user = "Mementomoristultus",
  time = "2023-06-22T02:23:23Z",
  time_2 = "2023-06-22T00:38:20Z"
)
chsts
chsts2 <- osm_query_changesets(
  bbox = c("-9.3015367,41.8073642,-6.7339533,43.790422"),
  user = "Mementomoristultus",
  closed = TRUE
)

chsts2

## End(Not run)

osm_read_bbox_notes

Retrieve notes by bounding box

Description

Returns the existing notes in the specified bounding box. The notes will be ordered by the date of their last change, the most recent one will be first.

Usage

osm_read_bbox_notes(
  bbox, limit = 100, closed = 7, format = c("R", "xml", "rss", "json", "gpx")
)

Arguments

bbox Coordinates for the area to retrieve the notes from (left,bottom,right,top). Floating point numbers in degrees, expressing a valid bounding box, not larger than the configured size limit, 25 square degrees, not overlapping the dateline.

limit Specifies the number of entries returned at max. A value between 1 and 10000 is valid. Default to 100.

closed Specifies the number of days a note needs to be closed to no longer be returned. A value of 0 means only open notes are returned. A value of -1 means all notes are returned. Default to 7.

format Format of the output. Can be "R" (default), "xml", "rss", "json" or "gpx".

Value

If format = "R", returns a data frame with one map note per row.

format = "xml":
Returns a xml2::xml_document with the following format:
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="OpenStreetMap server" copyright="OpenStreetMap and contributors" attribution="https://www.openstreetmap.org/copyright" license="https://opendatacommons.org/licenses/odbl/1-0/">
<note lon="0.1000000" lat="51.0000000">
<id>16659</id>
<url>https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659</url>
<comment_url>https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/comment</comment_url>
<close_url>https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/close</close_url>
<date_created>2019-06-15 08:26:04 UTC</date_created>
<status>open</status>
<comments>
<comment>
<date>2019-06-15 08:26:04 UTC</date>
<uid>1234</uid>
<user>userName</user>
<user_url>https://master.apis.dev.openstreetmap.org/user/userName</user_url>
<action>opened</action>
</comment>
...</comments>
</note>
...</osm>

format = "json";
Retuns a list with the following json structure:

{
"type": "FeatureCollection",
"features": [
{
"type": "Feature",
"geometry": {"type": "Point", "coordinates": [0.1000000, 51.0000000]},
"properties": {
"id": 16659,
"url": "https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659.json",
"comment_url": "https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/comment.json",
"close_url": "https://master.apis.dev.openstreetmap.org/api/0.6/notes/16659/close.json",
"date_created": "2019-06-15 08:26:04 UTC",
"status": "open",
"comments": [
{"date": "2019-06-15 08:26:04 UTC", "uid": 1234, "user": "userName", "user_url": "https://master.apis.dev.openstreetmap.org/user/userName"},
...]
}
}
]
For format in "rss", and "gpx", a xml2::xml_document with the corresponding format.

Note
The comment properties (uid, user, user_url) will be omitted if the comment was anonymous.

See Also
Other get notes' functions: osm_feed_notes(), osm_get_notes(), osm_search_notes()

Examples
notes <- osm_read_bbox_notes(bbox = c(3.7854767, 39.7837403, 4.3347931, 40.1011851), limit = 10)
## bbox as a character value also works (bbox = "3.7854767,39.7837403,4.3347931,40.1011851").
notes

Description
Used by the Data Working Group to hide old versions of elements containing data privacy or copy-right infringements. Only permitted for OSM accounts with the moderator role (DWG and server admins).

Usage
osm_redaction_object(
  osm_type = c("node", "way", "relation"),
  osm_id,
  version,
  redaction_id
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>osm_type</td>
<td>Object type (&quot;node&quot;, &quot;way&quot; or &quot;relation&quot;).</td>
</tr>
<tr>
<td>osm_id</td>
<td>Object id represented by a numeric or a character value.</td>
</tr>
<tr>
<td>version</td>
<td>Version of the object to redact.</td>
</tr>
<tr>
<td>redaction_id</td>
<td>If missing, then this is an unredact operation. If a redaction ID was specified, then set this element to be redacted in that redaction.</td>
</tr>
</tbody>
</table>

Details
The redaction_id is listed on https://www.openstreetmap.org/redactions. More information can be found in the source.
Value

Nothing is returned upon successful redaction or unredaction of an object.

See Also

Other functions for moderators: `osm_delete_note()`, `osm_hide_comment_changeset_discussion()`

Examples

```r
## Not run:
## WARNING: this example will edit the OSM (testing) DB with your user!
# You will need a user with moderator role in the server to use `osm_redaction_object()`
set_osmapi_connection(server = "testing") # setting https://master.apis.dev.openstreetmap.org
x <- data.frame(type = "node", lat = 0, lon = 0, name = "Test redaction.")
obj <- osmapi_objects(x, tag_columns = "name")
changeset_id <- osm_create_changeset(
  comment = "Test object redaction",
  hashtags = ":#testing;#osmapiR"
)
node_id <- osm_create_object(x = obj, changeset_id = changeset_id)
node_osm <- osm_get_objects(osm_type = "node", osm_id = node_id)
deleted_version <- osm_delete_object(x = node_osm, changeset_id = changeset_id)
redaction <- osm_redaction_object(
  osm_type = node_osm$type, osm_id = node_osm$id, version = 1, redaction_id = 1
)
unredaction <- osm_redaction_object(osm_type = node_osm$type, osm_id = node_osm$id, version = 1)

## End(Not run)
```

---

**osm_relations_object**

*Relations of an object*

Description

Returns all (not deleted) relations in which the given object is used.

Usage

```r
osm_relations_object(
  osm_type = c("node", "way", "relation"),
  osm_id,
  format = c("R", "xml", "json"),
  tags_in_columns = FALSE
)
```
Arguments

- **osm_type**: Object type ("node", "way" or "relation").
- **osm_id**: Object id represented by a numeric or a character value.
- **format**: Format of the output. Can be "R" (default), "xml", or "json".
- **tags_in_columns**: If FALSE (default), the tags of the objects are saved in a single list column `tags` containing a data.frame for each OSM object with the keys and values. If TRUE, add a column for each key. Ignored if format != "R".

Value

If format = "R", returns a data frame with one OSM object per row. If format = "xml", returns a `xml2::xml_document` following the OSM_XML format. If format = "json", returns a list with a json structure following the OSM_JSON format.

See Also

Other get OSM objects' functions: `osm_bbox_objects()`, `osm_get_objects()`, `osm_history_object()`, `osm_ways_node()`, `osmapi_objects()`

Examples

```r
node <- osm_relations_object(osm_type = "node", osm_id = 152364165)
node

way <- osm_relations_object(osm_type = "way", osm_id = 372011578)
way

rel <- osm_relations_object(osm_type = "relation", osm_id = 342792)
rel
```

Description

Returns notes that match the specified query. If no query is provided, the most recently updated notes are returned.
closed = 7,
sort = c("updated_at", "created_at"),
order = c("newest", "oldest"),
limit = getOption("osmapir.api_capabilities")$api$notes["default_query_limit"],
format = c("R", "xml", "rss", "json", "gpx")
)

Arguments

q
Text search query, matching either note text or comments.

user
Search for notes which the given user interacted with. The value can be the user id (numeric) or the display name (character).

bbox
Search area expressed as a string or a numeric vector of 4 coordinates of a valid bounding box (left,bottom,right,top) in decimal degrees. Area must be at most 25 square degrees (see osm_capabilities()$note_area and this line in settings for the current value).

from
Beginning date range for created_at or updated_at (specified by sort). Preferably in ISO 8601 date format.

to
End date range for created_at or updated_at (specified by sort). Preferably in ISO 8601 date format. Only works when from is supplied.

closed
Specifies the number of days a note needs to be closed to no longer be returned. A value of 0 means only open notes are returned. A value of -1 means all notes are returned. 7 is the default.

sort
Sort results by creation ("created_at") or update date ("updated_at", the default).

order
Sorting order. "oldest" is ascending order, "newest" is descending order (the default).

limit
Maximum number of results between 1 and 10000 (may change, see osm_capabilities()$api$notes for the current value). Default to 100.

format
Format of the the returned list of notes. Can be "R" (default), "xml", "rss", "json" or "gpx".

Details

The notes will be ordered by the date of their last change, the most recent one will be first.

Value

If format = "R", returns a data frame with one map note per row. If format = "json", returns a list with the json structure. For format in "xml", "rss", and "gpx", a xml2::xml_document with the corresponding format.

See Also

Other get notes’ functions: osm_feed_notes(), osm_get_notes(), osm_read_bbox_notes()
Examples

```r
notes <- osm_search_notes(
  q = "POI", bbox = "0.1594133,40.5229822,3.3222508,42.8615226",
  from = "2017-10-01", to = "2018-10-27T15:27A", limit = 10
)
notes

my_notes <- osm_search_notes(
  user = "jmaspons", bbox = c(-0.1594133, 40.5229822, 3.322251, 42.861523),
  closed = -1, format = "json"
)
my_notes
```

---

**osm_subscribe_changeset_discussion**

*Subscribe or unsubscribe to a changeset discussion*

**Description**

Subscribe or unsubscribe to a changeset discussion

**Usage**

```r
osm_subscribe_changeset_discussion(changeset_id)
```

```r
osm_unsubscribe_changeset_discussion(changeset_id)
```

**Arguments**

- `changeset_id` The id of the changeset represented by a numeric or a character value.

**Value**

Returns the changeset information.

**Functions**

- `osm_subscribe_changeset_discussion()`: Subscribe to the discussion of a changeset to receive notifications for new comments.
- `osm_unsubscribe_changeset_discussion()`: Unsubscribe from the discussion of a changeset to stop receiving notifications.

**See Also**

Other changeset discussion's functions: `osm_comment_changeset_discussion()`, `osm_hide_comment_changeset_discussion()`.
Examples

## Not run:
# set_osmapi_connection(server = "openstreetmap.org")
osm_subscribe_changeset_discussion(137595351)
osm_unsubscribe_changeset_discussion("137595351")
## End(Not run)

---

osm_update_gpx  Update GPS trace

Description

Use this to update a GPX info. Only usable by the owner account. Requires authentication.

Usage

```r
osm_update_gpx(
  gpx_id, name,
  description, tags,
  visibility = c("private", "public", "trackable", "identifiable")
)
```

Arguments

- **gpx_id**: The id of the track to update represented by a numeric or a character value.
- **name**: The file name of the track. Usually, the file name when using `osm_create_gpx()`.
- **description**: The trace description.
- **tags**: A string containing tags for the trace that will replace the current ones.
- **visibility**: One of the following: `private`, `public`, `trackable`, `identifiable`. For explanations see OSM trace upload page or Visibility of GPS traces).

Details

Missing arguments won’t be updated.

Value

Returns a data frame with the updated metadata of the GPS trace. The same format that `osm_get_gpx_metadata()` with format = "R".

See Also

Other edit GPS traces’ functions: `osm_create_gpx()`, `osm_delete_gpx()`
Examples

vignette("how_to_edit_gps_traces", package = "osmapiR")

---

### osm_update_object

*Update an OSM object*

#### Description
 Updates data from a preexisting element.

#### Usage

```r
osm_update_object(x, changeset_id)
```

#### Arguments

- `x`  
The new object data. Can be the path of an xml file, a `xml2::xml_document` or a data.frame inheriting or following the structure of an `osmapi_objects` object.
- `changeset_id`  
The ID of an open changeset where to create the object. If missing, `x` should define the changeset ID, otherwise it will be overwritten with `changeset_id`. Ignored if `x` is a path.

#### Details

A full representation of the element as it should be after the update has to be provided. Any tags, way-node refs, and relation members that remain unchanged must be in the update as well. A version number must be provided as well, it must match the current version of the element in the database.

If `x` is a data.frame, the columns `type`, `id`, `visible`, `version`, `changeset`, and `tags` must be present + column `members` for ways and relations + `lat` and `lon` for nodes. For the xml format, see the [OSM wiki](https://wiki.openstreetmap.org/).

If multiple elements are provided only the first is updated. The rest is discarded.

#### Value

Returns the new version number of the object.

#### Note

- This updates the bounding box of the changeset.
- To avoid performance issues when updating multiple objects, the use of the `osm_diff_upload_changeset()` is highly recommended. This is also the only way to ensure that multiple objects are updated in a single database transaction.

#### See Also

Other edit OSM objects' functions: `osm_create_object()`, `osm_delete_object()"
Examples

vignette("how_to_edit_osm", package = "osmapiR")

osm_ways_node

Ways of a node

Description

Returns all the (not deleted) ways in which the given node is used.

Usage

osm_ways_node(node_id, format = c("R", "xml", "json"), tags_in_columns = FALSE)

Arguments

node_id  Node id represented by a numeric or a character value.
format   Format of the output. Can be "R" (default), "xml", or "json".
tags_in_columns
          If FALSE (default), the tags of the objects are saved in a single list column tags containing a data.frame for each OSM object with the keys and values. If TRUE, add a column for each key. Ignored if format != "R".

Value

If format = "R", returns a data frame with one OSM object per row. If format = "xml", returns a xml2::xml_document following the OSM_XML format. If format = "json", returns a list with a json structure following the OSM_JSON format.

See Also

Other get OSM objects' functions: osm_bbox_objects(), osm_get_objects(), osm_history_object(), osm_relations_object(), osmapi_objects()

Examples

ways_node <- osm_ways_node(node_id = 35308286)
ways_node
set_osmapi_connection

Configure connections from osmapiR

Description
Functions to configure the connections. Probably, you should only use set_osmapi_connection.

Usage

```r
set_osmapi_connection(
  server = c("openstreetmap.org", "testing"),
  cache_authentication = TRUE
)

get_osmapi_url()

set_osmapi_url(osmapi_url)
```

Arguments

- **server**: If openstreetmap.org (default), the API calls will be performed to the servers in production. If testing, the calls will be against https://master.apis.dev.openstreetmap.org without affecting the main OSM data.
- **cache_authentication**: If TRUE, the authentication token will be cached on disk. This reduces the number of times that you need to re-authenticate at the cost of storing access credentials on disk. Cached tokens are encrypted and automatically deleted 30 days after creation. If missing (default), no changes will be applied. On package load time, the option is set to FALSE if it’s not yet set.
- **osmapi_url**: The desired API URL to send the calls.

Details

When testing your software against the API you should consider using https://master.apis.dev.openstreetmap.org instead of the live-api (set_osmapi_connection("testing"). Your account for the live service is not in the same database, so you probably need a new username and password for the test service: please visit that page in a browser to sign up.

set_osmapi_url() and get_osmapi_url only deal with the API base URL. On the other hand, set_osmapi_connection also configure the authentication parameters needed for PUT, POST and DELETE calls.

For further details, see https://wiki.openstreetmap.org/wiki/API_v0.6.

Value

Configure .Options[grep("osmapir\.[a-z]+(_?!secret$)", names(.Options), perl = TRUE)] :) and return osmapir.base_api_url.
See Also

Other API functions: authenticate_osmapi(), osm_api_versions(), osm_capabilities(), osm_permissions()

Examples

ori <- get_osmapi_url()
set_osmapi_connection(server = "testing")
get_osmapi_url()
set_osmapi_connection(server = "openstreetmap.org")
get_osmapi_url()

## Restore options
if (ori == "https://api.openstreetmap.org") {
  set_osmapi_connection(server = "openstreetmap.org")
} else if (ori == "https://master.apis.dev.openstreetmap.org") {
  set_osmapi_connection(server = "testing")
} else {
  warning("A non standard osmapiR connection detected (" , ori,
  "). If you configured manually options like \"osmapi.r.base_api_url\" or \"osmapi.r.oauth_id\", ",
  "configure it again."
}

---

**tags_list2wide**  
*Change tags from a list column <-> columns for each key in wide format*

Description

Objects of classes osmapi_objects and osmapi_changesets can represent the tags in a column with a list with a data.frame for each row with 2 columns for keys and values, or by columns for each key. These functions allow to change the format of the tags.

Usage

```r
tags_list2wide(x)
```

```r
tags_wide2list(x)
```

Arguments

`x`  
An osmapi_objects or osmapi_changesets objects as returned by, for example, osm_get_objects() or osm_get_changesets().
Details

Both formats have advantages. Tags in a list of data.frames is a more compact representation and there is no risk of clashes of column names and tag keys. Tags in columns make it easier to select rows by tags as in a regular data.frame. Column name clashes are resolved and the original key names restored when transformed to tags list format.

By default, functions returning `osmapi_objects` or `osmapi_changesets` objects, use the tags in a list column, but can return the results in a wide format using the parameter `tags_in_columns = TRUE`.

Value

A data frame with the same class and data than the original (`osmapi_objects` or `osmapi_changesets`) but with the specified tags' format.

Examples

```r
## Not run:
peaks_wide <- osm_get_objects(
  osm_type = "nodes", osm_id = c(35308286, 1935675367), tags_in_columns = TRUE
)
peaks_list <- tags_wide2list(peaks_wide)

# tags in list format
peaks_list$tags

# Select peaks with `prominence` tag
peaks_wide[!is.na(peaks_wide$prominence), ]
peaks_list[sapply(peaks_list$tags, function(x) any(x$key == "prominence")), ]

cities_list <- osm_get_objects(osm_type = "relations", osm_id = c("40581", "341530"))
# Column name clash:
cities_wide <- tags_list2wide(cities_list)

## End(Not run)
```
Index

* API functions
  authenticate_osmapi, 3
  osm_api_versions, 8
  osm_capabilities, 10
  osm_permissions, 41
  set_osmapi_connection, 55
* OsmChange's functions
  osm_diff_upload_changeset, 22
  osm_download_changeset, 24
  osmchange_create, 4
  osmchange_delete, 5
  osmchange_modify, 6
* changset discussion's functions
  osm_comment_changeset_discussion, 12
  osm_hide_comment_changeset_discussion, 38
  osm_subscribe_changeset_discussion, 51
* edit GPS traces' functions
  osm_create_gpx, 15
  osm_delete_gpx, 18
  osm_update_gpx, 52
* edit OSM objects' functions
  osm_create_object, 17
  osm_delete_object, 20
  osm_update_object, 53
* edit changeset's functions
  osm_create_changeset, 13
  osm_diff_upload_changeset, 22
* edit notes' functions
  osm_close_note, 11
  osm_create_comment_note, 14
  osm_create_note, 16
  osm_delete_note, 19
* functions for moderators
  osm_delete_note, 19
  osm_hide_comment_changeset_discussion, 38
  osm_redaction_object, 47
* get GPS' functions
  osm_get_data_gpx, 28
  osm_get_gpx_metadata, 29
  osm_get_points_gps, 33
  osm_list_gpxs, 40
* get OSM objects' functions
  osm_bbox_objects, 8
  osm_get_objects, 31
  osm_history_object, 39
  osm_relations_object, 48
  osm_ways_node, 54
  osmapi_objects, 3
* get changesets' functions
  osm_download_changeset, 24
  osm_get_changesets, 26
  osm_query_changesets, 42
* get notes' functions
  osm_feed_notes, 25
  osm_get_notes, 30
  osm_read_bbox_notes, 45
  osm_search_notes, 49
* methods
  tags_list2wide, 56
* users' functions
  osm_details_logged_user, 21
  osm_get_preferences_user, 35
  osm_get_user_details, 36
  authenticate_osmapi, 3, 8, 10, 42, 56
  get_osmapi_url(set_osmapi_connection), 55
  logout_osmapi(authenticate_osmapi), 3
  osm_api_versions, 3, 8, 10, 42, 56
  osm_bbox_objects, 4, 8, 33, 39, 49, 54
  osm_capabilities, 3, 8, 10, 42, 56
  osm_capabilities(), 24
osm_close_changeset
  (osm_create_changeset), 13
osm_close_note, 11, 15, 16, 19
osm_comment_changeset_discussion, 12, 38, 51
osm_create_changeset, 13, 24
osm_create_comment_note, 11, 14, 16, 19
osm_create_gpx, 15, 18, 52
osm_create_gpx(), 52
osm_create_note, 11, 15, 16, 19
osm_create_object, 17, 20, 53
osm_delete_gpx, 16, 18, 52
osm_delete_note, 11, 15, 16, 19, 38, 48
osm_delete_object, 18, 20, 53
osm_detailsLoggedUser, 21, 36, 37
osm_diff_upload_changeset, 5–7, 14, 22, 25
osm_diff_upload_changeset(), 5–7, 17, 20, 25
osm_download_changeset, 5–7, 24, 24, 27, 44
osm_download_changeset(), 26, 43
osm_feed_notes, 25, 31, 47, 49
osm_get_changesets, 25, 26, 44
osm_get_changesets(), 12, 24, 38, 56
osm_get_data_gpx, 28, 29, 34, 41
osm_get_gpx_metadata, 28, 29, 34, 41
osm_get_gpx_metadata(), 40, 52
osm_get_notes, 25, 30, 47, 50
osm_get_notes(), 11, 15, 16, 19
osm_get_objects, 4, 9, 31, 39, 49, 54
osm_get_objects(), 7, 56
osm_get_points_gpx, 28, 29, 33, 41
osm_get_preferences_user, 22, 35, 37
osm_get_user_details, 22, 36, 36
osm_hide_comment_changeset_discussion, 12, 19, 38, 48, 51
osm_history_object, 4, 9, 33, 39, 49, 54
osm_list_gpxs, 28, 29, 34, 40
osm_permissions, 3, 8, 10, 41, 56
osm_query_changesets, 25, 27, 42
osm_query_changesets(), 10
osm_read_bbox_notes, 25, 31, 45, 50
osm_read_bbox_notes(), 10
osm_redaction_object, 19, 38, 47
osm_relations_object, 4, 9, 33, 39, 48, 54
osm_reopen_note (osm_close_note), 11
osm_reopen_note(), 19
osm_search_notes, 25, 31, 47, 49
osm_search_notes(), 10
osm_set_preferences_user
  (osm_get_preferences_user), 35
osm_subscribe_changeset_discussion, 12, 38, 51
osm_unhide_comment_changeset_discussion
  (osm_hide_comment_changeset_discussion), 38
osm_unsubscribe_changeset_discussion
  (osm_subscribe_changeset_discussion), 51
osm_update_changeset
  (osm_create_changeset), 13
osm_update_gpx, 16, 18, 52
osm_update_object, 18, 20, 53
osm_ways_node, 4, 9, 33, 39, 49, 54
osmapi_objects, 4, 9, 33, 39, 49, 54
osmchange_create, 4, 6, 7, 24, 25
osmchange_delete, 5, 5, 7, 24, 25
osmchange_modify, 5, 6, 6, 24, 25
osmchange_modify(), 4
set_osmapi_connection, 3, 8, 10, 42, 55
set_osmapi_url (set_osmapi_connection), 55
tags_list2wide, 56
tags_wide2list (tags_list2wide), 56
xml2::write_xml(), 5–7
xml2::xml_document, 5–7, 9, 17, 20, 21, 23, 25, 26, 28–32, 34, 35, 37, 39, 40, 43, 45, 47, 49, 50, 53, 54