Package ‘sbtools’

January 8, 2020

Title USGS ScienceBase Tools
Maintainer David Blodgett <dblodgett@usgs.gov>
Version 1.1.9
Description Tools for interacting with U.S. Geological Survey ScienceBase
<https://www.sciencebase.gov> interfaces. ScienceBase is a data cataloging and
collaborative data management platform. Functions included for querying
ScienceBase, and creating and fetching datasets.
Imports jsonlite, curl, httr (>= 1.0.0), stringr, methods
Suggests testthat, xml2, sf, sp
License CC0
URL https://github.com/USGS-R/sbtools
BugReports https://github.com/USGS-R/sbtools/issues
RoxygenNote 6.1.1
Encoding UTF-8
NeedsCompilation no
Author David Blodgett [cre],
Tim Kern [aut],
Luke Winslow [aut],
Scott Chamberlain [aut],
Alison Appling [aut],
Jordan Read [aut]
Repository CRAN
Date/Publication 2020-01-08 09:10:03 UTC

R topics documented:
sbtools-package .......................................................... 3
authenticate_sb ......................................................... 3
current_session ......................................................... 4
folder_create .......................................................... 4
topics documented:

- identifier_exists .................................................. 5
- is_logged_in .......................................................... 6
- items_create .......................................................... 6
- items_update .......................................................... 8
- items_upsert .......................................................... 9
- item_append_files .................................................... 10
- item_create ............................................................ 10
- item_exists ............................................................ 11
- item_file_download ................................................... 12
- item_get ................................................................. 13
- item_get_fields ......................................................... 14
- item_get_parent ......................................................... 15
- item_get_wfs ........................................................... 15
- item_list_children ..................................................... 16
- item_list_files ......................................................... 17
- item_move ............................................................... 18
- item_rename_files ..................................................... 19
- item_replace_files .................................................... 19
- item_rm ................................................................. 20
- item_rm_files .......................................................... 21
- item_update ............................................................ 22
- item_update_identifier .............................................. 22
- item_upload_create .................................................... 23
- item_upsert ............................................................. 24
- query_items ............................................................. 25
- query_item_identifier .................................................. 28
- query_item_in_folder .................................................. 29
- query_sb ................................................................. 29
- query_sb_datatype ..................................................... 31
- query_sb_date ........................................................... 32
- query_sb_doi ............................................................ 33
- query_sb Spatial ........................................................ 34
- query_sb_text ........................................................... 35
- sbitem ................................................................. 35
- sb_datatypes ............................................................ 36
- sb_ping ................................................................. 37
- session_details ........................................................ 37
- session_logout ........................................................ 38
- session_renew ........................................................ 38
- session_validate ........................................................ 39
- set_endpoint ........................................................... 40
- user_id ................................................................. 41

Index ................................................................. 42
This package provides a rich interface to USGS’s ScienceBase [https://www.sciencebase.gov/](https://www.sciencebase.gov/) - a data cataloging and collaborative data management platform. For further information, see the sbtools manuscript [here](https://www.sciencebase.gov/). Functions are included for searching for data, retrieving, creating, and updating datasets.

**Details**

Functionality in this package allows all users to query ScienceBase for data using a variety of metadata types (`query_sb_text`, `query_sb_doi`, `query_sb_spatial`). Items and associated information can be requested by `item_get` including item parents `item_get_parent` and children `item_list_children`. Data and attached files can be accessed for all available items through provided functionality (e.g., `item_get_wfs` and `item_file_download`).

**Authentication**

See the function `authenticate_sb` to authenticate. You’ll be required to pass in your ScienceBase username and password.

Authenticated users can create, update, and remove items (`item_list_children`, `item_create`, `item_update`, `item_rm`).

**Feedback**

Report any feedback or bugs at [https://github.com/USGS-R/sbtools/issues](https://github.com/USGS-R/sbtools/issues)

### authenticate_sb

**Authorize to SB for subsequent calls**

**Description**

This connects to SB, authenticates and gets a session token for communicating with SB. If you do not supply a username or password, you will be prompted to enter them.

**Usage**

```r
authenticate_sb(username, password)
```

**Arguments**

- `username`       Sciencebase username
- `password`       Sciencebase password, prompts user if not supplied
### current_session

**Return current cached session**

**Description**

Returns the currently cached SB session. If there is no authenticated session, returns NULL. Emits a warning if the session has expired.

**Usage**

```python
current_session()
```

**Examples**

```python
session = current_session()
# null unless currently authenticated
session
```

### folder_create

**Create a folder**

**Description**

Create a special kind of item on ScienceBase that is intended to be a "folder" that contains one or more child items. This is similar to a standard item ([item_create](#item_create)) but defaults to showing child-items on the ScienceBase web interface.

**Usage**

```python
folder_create(parent_id = user_id(), name, ..., 
               session = current_session())
```

**Arguments**

- **parent_id**
  - An `sbitem` object or character ScienceBase ID corresponding to the parent item (folder)
- **name**
  - (character) the folder name
- **...**
  - Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- **session**
  - Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

**Value**

A `response` object
identifier_exists

Examples

```r
## Not run:
folder_create(name="foobar345")

## End(Not run)
```

---

**identifier_exists**  
*Check if identifier exists*

**Description**

This function quickly checks to see if an identifier exists. It does a quick head request to skip the overhead of item metadata retrieval. This will also return FALSE if the identifier exists but is associated with an item that is unavailable due to permission restrictions.

**Usage**

```r
identifier_exists(sb_id, ..., session = current_session())
```

**Arguments**

- `sb_id` An *sbitem* object or a character ScienceBase ID corresponding to the item
- `...` Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE
- `session` Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

**Value**

Logical, TRUE or FALSE

**Examples**

```r
# identifier exists
identifier_exists(sb_id = "4f4e4b24e4b07f02db6aea14")

# identifier does not exist
identifier_exists(sb_id = "aaaaaaakkkkkkkbbbbbb")
```
is_logged_in  
*Check whether you’re logged into a ScienceBase session*

**Description**

Check whether you’re logged into a ScienceBase session

**Usage**

```
is_logged_in(..., session = current_session())
```

**Arguments**

- `...`: Additional parameters are passed on to `GET`
- `session`: SB session object from `authenticate_sb`

**Value**

Logical, TRUE or FALSE

**Examples**

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

---

items_create  
*Create many new SB items*

**Description**

A method to create multiple ScienceBase items with a single call and a single HTTP service request. Can be useful for improving performance of creating a large number of items at once.

**Usage**

```
items_create(parent_id = user_id(), title, ..., info = NULL, 
session = current_session())
```
**items_create**

**Arguments**

- **parent_id**
  An `sbitem` object or character ScienceBase ID corresponding to the parent item (folder). This must be of length 1 or more. If length 1, then we recycle it for every item.

- **title**
  Two or more titles for the new SB items

- **info**
  (optional) list of metadata info for the new items. for each item include a named list of variables

- **session**
  Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

**Details**

The length of the `title` and `info` values must be the same length - however, the `parent_id` can be of length 1 or equal to the length of each of `title` and `info` parameters

**Value**

One or more objects of class `sbitem` in a list

**Examples**

```r
## Not run:
# helper function to make a random name
aname <- function() paste0(sample(letters, size = 5, replace = TRUE), collapse = "")

# Create some items - by default we use your user ID
items_create(title = c(aname(), aname()))

# add additional items in the info parameter - by default we use your user ID
items_create(title = c(aname(), aname()),
info = list(
  list(contacts = list(list(name = "Suzy"))),
  list(contacts = list(list(name = "Brandy")))
))

# another example with more information - by default we use your user ID
items_create(title = c(aname(), aname()),
info = list(
  list(contacts = list(list(name = "Suzy"))),
  list(contacts = list(list(name = "Brandy")))
))

# Pass an object of class sbitem
(x <- folder_create(user_id(), aname()))
items_create(x, title = c(aname(), aname()))
```

## End(Not run)
items_update  
**Update many SB items with new metadata**

**Description**

A method to update multiple ScienceBase items with a single call and a single HTTP service request. Can be useful for improving performance of updating a large number of items at once.

**Usage**

```
items_update(sb_id, info, ..., session = current_session())
```

**Arguments**

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item
- **info**: list of metadata info (key-value pairs) to change on the item
- **...**: Additional parameters are passed on to `PUT`
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

**Details**

If length of `sb_id` > 1, then length of `info` input must be the same

**Value**

One or more objects of class `sbitem` in a list

**Examples**

```r
## Not run:
# helper function to make a random name
aname <- function() paste0(sample(letters, size = 5, replace = TRUE), collapse = "")

res <- items_create(user_id(), title = c(aname(), aname()))
out <- items_update(res, info = list( list(title = aname()), list(title = aname()) ) )
vapply(out, "[[", "", "title")
## End(Not run)
```
items_upsert  

**Description**

Either creates or updates (if items already exist)

**Usage**

```r
items_upsert(parent_id = user_id(), title = NULL, ..., info = NULL, session = current_session())
```

**Arguments**

- `parent_id`: An `sbitem` object or character ScienceBase ID corresponding to the parent item (folder)
- `title`: The title of the new SB item
- `...`: Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- `info`: (optional) list of metadata info for the new item
- `session`: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

**Value**

An object of class `sbitem`

**Examples**

```r
## Not run:
# helper function to make a random name
aname <- function() paste0(sample(letters, size = 5, replace = TRUE), collapse = "")

# Create some item - by default we use your user ID
z1 <- item_create(title = aname())
z2 <- item_create(title = aname())

# Upsert items
(x <- items_upsert(list(z1, z2), title = c(aname(), aname())))

# Call item_upsert again, updates this time
items_upsert(x, info = list(
  contacts = list(list(name = "Suzy"))
))
```

## End(Not run)
### item_append_files  
**Upload File to Item**

**Description**

Adds a file to an item

**Usage**

```r
item_append_files(sb_id, files, ..., session = current_session())
```

**Arguments**

- `sb_id`  
  An `sbitem` object or a character ScienceBase ID corresponding to the item
- `files`  
  A file path to upload.
- `...`  
  Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- `session`  
  Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

**Value**

An object of class `sbitem`

**Examples**

```r
## Not run:
res <- item_create(user_id(), "testing 123")
cat("foo bar", file = "foobar.txt")
item_append_files(res$id, "foobar.txt")
## End(Not run)
```

### item_create  
**Create a new SB item**

**Description**

Create a new item on ScienceBase with the requested parent and item title. Info can be provided to populate metadata at the time of creation.

**Usage**

```r
item_create(parent_id = user_id(), title, ..., info,  
session = current_session())
```
Arguments

- **parent_id**: An `sbitem` object or character ScienceBase ID corresponding to the parent item (folder)
- **title**: The title of the new SB item
- ... Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- **info**: (optional) list of metadata info for the new item
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

Value

An object of class `sbitem`

Examples

```r
## Not run:
# Create an item - by default we use your user ID
item_create(title = "testing 123")

# Pass an object of class sbitem
x <- folder_create(user_id(), "foobar456")
item_create(x, "foobar456-item")
## End(Not run)
```

---

**item_exists**: check if identifier tuple already exists on SB

Description

returns TRUE if tuple already belongs to a sciencebase item, FALSE if not

Usage

```r
item_exists(scheme, type, key, ..., session = current_session())
```

Arguments

- **scheme**: the identifier scheme
- **type**: the identifier type
- **key**: the identifier key
- ... Additional parameters are passed on to `GET`
- **session**: an SB session
item_file_download

Value

boolean for whether item exists

Examples

## Not run:
item_exists('mda_streams','ts_doobs','nwis_01018035')
item_exists('mda_streams','site_root','nwis_01018035')

## End(Not run)

Description

Function to download files attached to an item on SB. Either files can be specified directly using the names and destinations parameters, or a dest_dir can be supplied where all attached files will be written with the names as stored on SB.

Usage

item_file_download(sb_id, ..., names, destinations, dest_dir,
session = current_session(), overwrite_file = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sb_id</td>
<td>An sbitem object or a character ScienceBase ID corresponding to the item</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE</td>
</tr>
<tr>
<td>names</td>
<td>String vector list of file names attached to item that you wish to download.</td>
</tr>
<tr>
<td>destinations</td>
<td>String vector list of destinations for requested files. Must be same length as names</td>
</tr>
<tr>
<td>dest_dir</td>
<td>A directory path for saving files when names parameter is omitted</td>
</tr>
<tr>
<td>session</td>
<td>Session object from authenticate_sb. Defaults to anonymous or last authenticated session</td>
</tr>
<tr>
<td>overwrite_file</td>
<td>Boolean indicating if file should be overwritten if it already exists locally</td>
</tr>
</tbody>
</table>

Value

Character vector of full paths to local files

Author(s)

Luke Winslow
item_get

Examples

## Not run:

# downloads two files attached to this item
item_file_download('548b2b31e4b03f6463662a4', dest_dir=tempdir())

# downloads a specific file attached to this item
item_file_download('548b2b31e4b03f6463662a4', names='gdp.txt',
                   destinations=file.path(tempdir(), 'fname.txt'))

## End(Not run)

---

item_get  

Retrieve SB item

Description

Retrieves an item and its metadata from ScienceBase based on its unique ID. Errors if the requested item ID does not exist or access is restricted due to permissions.

Usage

item_get(sb_id, ..., session = current_session())

Arguments

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item
- **...**: Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

Value

An object of class `sbitem`

Examples

# Get an item
item_get("4f4e4b24e4b07f02db6aea14")

# Search for item IDs, then pass to item_get
library(httr)
res <- query_items(list(s = "Search", q = "water", format = "json"))
ids <- vapply(httr::content(res)$items, '[[', '', "id")
lapply(ids[1:3], item_get)
Description

Retrieve specific fields from an SB item

Usage

item_get_fields(sb_id, fields, ..., drop = TRUE, session = current_session())

Arguments

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item
- **fields**: a vector of fields
- **...**: Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE
- **drop**: logical. If only one field is selected, should the list format be dropped?
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

Value

List serialization of chosen metadata for an SB item

Examples

```r
# Get certain fields from an item
item_get_fields("4f4e4b24e4b07f02db6aea14", c('title', 'citation', 'contacts'))

# If only 1 field selection, do or don't drop list format
item_get_fields("4f4e4b24e4b07f02db6aea14", 'title')
item_get_fields("4f4e4b24e4b07f02db6aea14", 'title', drop = FALSE)

# End(Not run)
```
**item_get_parent**

*Get an item’s parent ID*

**Description**

Retrieves the parent of a supplied item based on the ScienceBase item tree hierarchy.

**Usage**

```r
item_get_parent(sb_id, ..., session = current_session())
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sb_id</code></td>
<td>An <code>sbitem</code> object or a character ScienceBase ID corresponding to the item</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Additional parameters are passed on to <code>GET</code>, <code>POST</code>, <code>HEAD</code>, <code>PUT</code>, or <code>DELETE</code></td>
</tr>
<tr>
<td><code>session</code></td>
<td>Session object from <code>authenticate_sb</code>. Defaults to anonymous or last authenticated session</td>
</tr>
</tbody>
</table>

**Value**

An item object representing the parent of the supplied item.

**Examples**

```r
item_get_parent("4f4e4b24e4b07f02db6aea14")
```

```r
item_get("4f4e4b24e4b07f02db6aea14") %>% item_get_parent()
```

---

**item_get_wfs**

*Download and load from SB WFS service (Deprecated)*

**Description**

This function attempts to download the spatial layer data attached to the requested SB item. SB exposes discrete spatial objects (points, polygons) as web services based on the Open Geospatial Consortium, Web Feature Service (WFS) standardized interface. This requires the following libraries not by default installed with sbtools: `sf`, `httr`, and `xml2`. You can install them simply by running `install.packages(c("xml2","httr","sf"))`

**Usage**

```r
item_get_wfs(sb_id, as_sf = FALSE, ..., session)
```
Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sb_id</td>
<td>An sbitem object or a character ScienceBase ID corresponding to the item</td>
</tr>
<tr>
<td>as_sf</td>
<td>boolean, return data in sf format</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE</td>
</tr>
<tr>
<td>session</td>
<td>Session object from authenticate_sb. Defaults to anonymous or last authen-</td>
</tr>
<tr>
<td></td>
<td>ticated session</td>
</tr>
</tbody>
</table>

**item_list_children** Return IDs for all child items

Description

Returns a list of child IDs for a ScienceBase item

Usage

```r
item_list_children(sb_id, fields = c("id", "title"), ..., 
session = current_session(), limit = 20)
```

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sb_id</td>
<td>An sbitem object or a character ScienceBase ID corresponding to the item</td>
</tr>
<tr>
<td>fields</td>
<td>A character vector of requested data fields. Defaults to 'id' and 'title'. Full list of possible fields is available online in SB documentation.</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE</td>
</tr>
<tr>
<td>session</td>
<td>Session object from authenticate_sb. Defaults to anonymous or last authen-</td>
</tr>
<tr>
<td></td>
<td>ticated session</td>
</tr>
<tr>
<td>limit</td>
<td>Max children returned.</td>
</tr>
</tbody>
</table>

Value

List of sbitem for each child item.

Examples

```r
## Not run:
item_list_children(user_id())

as.sbitem('5060b03ae4b00fc20c4f3c8b') %>% item_list_children 
item_get('5060b03ae4b00fc20c4f3c8b') %>% item_list_children

## End(Not run)
```
item_list_files

Get list of files attached to SB item

Description
Lists all files attached to a SB item. Files can be downloaded from ScienceBase using item_file_download. (advanced) Recursive options lists all files attached to an item and all children items.

Usage
item_list_files(sb_id, recursive = FALSE, ..., session = current_session())

Arguments
- sb_id: An sbitem object or a character ScienceBase ID corresponding to the item
- recursive: (logical) List files recursively. Default: FALSE
- ...: Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE
- session: Session object from authenticate_sb. Defaults to anonymous or last authenticated session

Value
A data.frame with columns fname, size, and url. If item has no attached files, returns a zero row data.frame.

Examples
## Not run:
item_list_files("4f4e4b24e4b07f02db6aea14")

# list files recursively
## create item
id <- item_create(user_id(), title="some title")
## 1. create nested item w/ file
file <- system.file("examples", "books.json", package = "sbtools")
id2 <- item_create(id, title = "newest-thing")
item_upload_create(id2, file)
## 2. create nested item w/ file
file <- system.file("examples", "species.json", package = "sbtools")
id3 <- item_create(id, title = "a-new-thing")
item_upload_create(id3, file)
## 3. create nested item w/ file
file <- system.file("examples", "data.csv", package = "sbtools")
id4 <- item_create(id, title = "another-thing")
item_upload_create(id4, file)
item_list_files(id = '56562348e4b07e7ea53e09d', recursive = FALSE) # default
item_list_files(id = '56562348e4b07e7ea53e09d', recursive = TRUE)
**item_move**

**Move item from one folder to another**

**Description**

Move item from one folder to another

**Usage**

```r
item_move(sb_id, id_new, ..., session = current_session())
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sb_id</code></td>
<td>An <code>sbitem</code> object or a character ScienceBase ID corresponding to the item</td>
</tr>
<tr>
<td><code>id_new</code></td>
<td>Folder/item to move id to. A ScienceBase ID or something that can be coerced to a SB item ID by <code>as.sbitem</code></td>
</tr>
<tr>
<td><code>...</code></td>
<td>Additional parameters are passed on to <code>GET</code>, <code>POST</code>, <code>HEAD</code>, <code>PUT</code>, or <code>DELETE</code></td>
</tr>
<tr>
<td><code>session</code></td>
<td>Session object from <code>authenticate_sb</code>. Defaults to anonymous or last authenticated session</td>
</tr>
</tbody>
</table>

**Value**

An object of class `sbitem`. Same as id, but with new parent id

**Examples**

```r
## Not run:
# create 1st folder
(fold1 <- folder_create(user_id(), "bear123"))
(res <- item_create(fold1, "item-to-move"))

# create 2nd folder
(fold2 <- folder_create(user_id(), "bear456"))

# move item in 1st folder to 2nd folder
(res2 <- item_move(res, fold2))

# test identical
identical(res$parentId, fold2$id)

## End(Not run)
```
**item_rename_files**

**Rename item attached files**

**Description**

Renames files attached to an SB item.

**Usage**

```r
item_rename_files(sb_id, names, new_names, ..., 
session = current_session())
```

**Arguments**

- `sb_id` An sbitem object or a character ScienceBase ID corresponding to the item
- `names` List of names of files to rename
- `new_names` List of new file names to use
- `...` Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- `session` Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

**Examples**

```r
## Not run:
names = c("file1.txt", "file2.txt")
new_names = c("newname1.txt", "newname2.txt")

item_rename_files("sbid", names, new_names)

## End(Not run)
```

**item_replace_files**

**Replace files associated with an item**

**Description**

Replaces existing files associated with an item with a new one. (Currently does not support multi-file uploads.) This function will not append an existing collection of files. If that is desired, use `item_append_files`
Usage

```r
item_replace_files(sb_id, files, ..., all = FALSE,
session = current_session())
```

Arguments

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item
- **files**: A character vector of file paths
- **...**: Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- **all**: A boolean indicating if all attached files should be removed before uploading new files. FALSE if only files with matching names should be replaced. If you wish to upload files with duplicate names, see `item_append_files`. Defaults to FALSE.
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

---

**item_rm**  
*Remove item from SB*

Description

Remove an item from ScienceBase. This is not reversible and will delete an item and its attached files. (advanced) Recursive is to be used with care and could result in unexpected file deletion.

Usage

```r
item_rm(sb_id, ..., limit = 1000, recursive = FALSE,
session = current_session())
```

Arguments

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item
- **...**: Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- **limit**: The maximum number of child items to remove when called with `recursive=TRUE`.
- **recursive**: logical, FALSE by default. CAUTION: setting `recursive=TRUE` means that not only will this item be deleted, but so will all its child items and their child items and so on.
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

Value

- **htttr response** object
Examples

```r
## Not run:
res <- item_create(user_id(), "item-to-delete")
item_rm(res)

## End(Not run)
```

---

`item_rm_files`  
Remove files associated with an item

Description

Removes existing files associated with an item.

This function is the key way to remove files attached to SB items.

Usage

```r
item_rm_files(sb_id, files, ..., session = current_session())
```

Arguments

- `sb_id`  
  An `sbitem` object or a character ScienceBase ID corresponding to the item
- `files`  
  A character vector of file names to remove. If not supplied, defaults to removing all attached files.
- `...`  
  Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- `session`  
  Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

Value

An updated object of class `sbitem`

Examples

```r
## Not run:
res <- item_create(user_id(), "item456")
cat("foo bar", file = "foobar.txt")
item_append_files(res, "foobar.txt")
res <- item_get(res)
res$files[[1]]$name
res2 <- item_rm_files(res)
res2$files

## End(Not run)
```
item_update

Update a SB item with new metadata

Description

Updates metadata associated with a ScienceBase item based on supplied list of new or updated metadata elements.

Usage

item_update(sb_id, info, ..., session = current_session())

Arguments

sb_id          An sbitem object or a character ScienceBase ID corresponding to the item
info           list of metadata info (key-value pairs) to change on the item
...            Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE
session       Session object from authenticate_sb. Defaults to anonymous or last authenticated session

Value

An object of class sbitem

Examples

## Not run:
res <- item_create(user_id(), "item-to-update")
out <- item_update(res, list(title = "item-updated"))
out$title

## End(Not run)

item_update_identifier

Add custom identifier to an existing item

Description

Adds or updates an item’s alternative identifier. This can add additional identifiers or update those already in place. See query_item_identifier for finding items based on alternative identifier.

Usage

item_update_identifier(sb_id, scheme, type, key, ..., session = current_session())
## item_upload_create

Upload file(s) and create a new item

### Description

Create a new item with files attached, all in one call to SB

### Usage

```r
item_upload_create(parent_id, files, ..., session = current_session())
```

### Arguments

- **parent_id**: An `sbitem` object or character ScienceBase ID corresponding to the parent item (folder)
- **files**: A string vector of paths to files to be uploaded
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

### Value

An object of class `sbitem`
item_upsert

Examples

```r
# helper function to make a random name
aname <- function() paste0(sample(letters, size = 5, replace = TRUE), collapse = "")

# Create an item - by default we use your user ID
(x <- item_upsert(title = aname())))

# Call item_upsert again, updates this time
item_upsert(x, info = list(
  contacts = list(list(name = "Suzy"))))
```

## Description

Either creates or updates (if item already exists)

## Usage

```r
item_upsert(parent_id = user_id(), title = NULL, ..., info = NULL, 
  session = current_session())
```

## Arguments

- **parent_id**: An `sbitem` object or character ScienceBase ID corresponding to the parent item (folder)
- **title**: The title of the new SB item
- **...**: Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- **info**: (optional) list of metadata info for the new item
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session

## Value

An object of class `sbitem`
query_items

) )

## End(Not run)

---

query_items  

*Query SB for items using generic query parameters*

**Description**

Query SB for items using generic query parameters

**Usage**

```r
query_items(query_list, ..., session = current_session())
```

**Arguments**

- `query_list`: List of item query selectors. See Details.
- `...`: Additional parameters are passed on to `GET`.
- `session`: Session object from `authenticate_sb`.

**Details**

The following is a list of query parameters you can use in the `query_list` parameter.

- `s` (character): Only option: "Search"
- `format` (character): One of "json", "xml", "csv", or "atom"
- `q` (character): Query string
- `q` (character): Lucene query string
- `max` (integer): Number of records to return. Default: 20
- `offset` (integer): Record to start at. Default: 1
- `fields` (character): Character vector of fields to return
- `folderId` (character): Alphanumeric string representing folder ID
- `parentId` (character): Alphanumeric string representing folder ID. This can be used to return all children items within the folder, but not within sub-folders.
- `sort` (character): One of "firstContact", "dateCreated", "lastUpdated", or "title". By default sorted by search score
- `order` (character): One of "asc" or "desc"
- `ids` Vector of item ids.
- `ancestors` (character): Alphanumeric string representing folder ID. This can be used to return all children items within the folder, even within sub-folders. Used as a filter
- `tags` Filter by tags, e.g, "distribution". Used as a filter
query_items

- `browseCategory` One of .... Used as a filter
- `browseType` One of .... Used as a filter
- `dateRange` A json string with keys `dateType` and `choice`. Where `dateType` is one of Acquisition, Award, Collected, dateCreated, Received, Reported, Transmitted, Due, End, Info, lastUpdated, Publication, Release, or Start. And where `choice` is one of day, week, month, year, or range (if range selected, also supply start and end keys with dates of the form YYYY-MM-DD). Used as a filter
- `projectStatus` One of Active, Approved, Completed, In Progress, Proposed. Used as a filter
- `spatialQuery` A WKT string. Used as a filter
- `extentQuery` Use existing extents (footprints) to search against item bounding boxes and representational points. This is a alphanumeric string.

Value

An object of class `response`

See Also

`query_item_identifier`, `query_item_in_folder`

Examples

```r
## Not run:
# Basic query
library("httr")
res <- query_items(list(s = "Search", q = "water", format = "json"))
httr::content(res)

# Paging
## max - number of results
res <- query_items(list(s = "Search", q = "water", format = "json", max = 2))
length(httr::content(res)$items)
res <- query_items(list(s = "Search", q = "water", format = "json", max = 30))
length(httr::content(res)$items)

## offset - start at certain record
res <- query_items(list(s = "Search", q = "water", format = "json",
max = 30, offset = 10))
httr::content(res)

## links - use links given in output for subsequent queries
httr::content(http::GET(content(res)$nextlink$url))

# Return only certain fields
res <- query_items(list(s = "Search", q = "water", format = "json", fields = 'title'))
httr::content(res)$items[[1]]

# Search a folder ID
res <- query_items(list(s = "Search", q = "water", format = "json",
folderId = '504216b9e4b04b508bf637d'))
```
http::content(res)$items

# Filter by ancestor
query_items(list(s = "Search", ancestors = "4f831626e4b0e84f6086809b", format = "json"))

# Filter by tags
content(query_items(list(s = "Search", tags = "distribution", format = "json")))

# Filter by browse category
content(query_items(list(s = "Search", browseCategory = "Image", format = "json")))

# Filter by browse type
content(query_items(list(s = "Search", browseType = "Collection", format = "json")))

# Filter by WKT geometry string
wkt1 <- "POLYGON((-104.4 41.0,-95.1 41.0,-95.1 37.5,-104.4 37.5,-104.4 41.0))"

wkt2 <- "POLYGON((-104.4 38.3,-95.2 38.3,-95.2 33.7,-104.4 34.0,-104.4 38.3))"

content(query_items(list(s = "Search", spatialQuery = wkt1, format = "json")))

content(query_items(list(s = "Search", spatialQuery = wkt1, spatialQuery = wkt2, format = "json")))

# Project status
content(query_items(list(s = "Search", projectStatus = "Active", format = "json")))

# Date range
query_items(list(s = "Search",
dateRange = '{"dateType":"Collected","choice":"year"}', format = "json"))

query_items(list(s = "Search",
dateRange = '{"dateType":"lastUpdated","choice":"month"}', format = "json"))

query_items(list(s = "Search",
dateRange = '{"dateType":"Release","choice":"range","start":"2014-09-01","end":"2015-09-01"}', format = "json"))

# Extent query
## just a alphanumeric code
content(query_items(list(s = "Search", extentQuery = '2873462', format = "json")))

## with buffering, intersect
content(query_items(list(s = "Search", extentQuery = '{"extent":2873462, "relation":"intersects","buffer":"5"}', format = "json")))

## with buffering, within
content(query_items(list(s = "Search", extentQuery = '{"extent":2873462, "relation":"within","buffer":"5"}', format = "json")))

## with buffering, within
content(query_items(list(s = "Search", extentQuery = '{"extent":2873462, "relation":"disjoint","buffer":"5"}', format = "json")))

# Lucene query
## note, you have to pass the q parameter if you pass the lq parameter
content(query_items(list(s = "Search", q = '', lq = "sage OR grouse")))

## End(Not run)
query_item_identifier  Query SB for items based on custom identifier

Description
Find all items under a scheme or also query by for a specific type and key

Usage
query_item_identifier(scheme, type = NULL, key = NULL, ..., 
    session = current_session(), limit = 20)

Arguments
scheme  The identifier scheme
type    (optional) The identifier type
key     (optional) The identifier key
...     Additional parameters are passed on to GET
session (optional) SB Session to use, not provided queries public items only
limit   Max number of matching items to return

Value
The SB item id for the matching item. NULL if no matching item found.

Examples
## Not run:
authenticate_sb()

ex_item = item_create(title='identifier example')
item_update_identifier(ex_item, 'project1', 'dataset1', 'key1')
ex2_item = item_create(title='identifier example 2')
item_update_identifier(ex2_item, 'project1', 'dataset1', 'key2')

#query the specific item
query_item_identifier('project1', 'dataset1', 'key1')

#or get the collection of items based on the ID hierarchy
query_item_identifier('project1')

item_rm(ex_item)
item_rm(ex2_item)

## End(Not run)
query_item_in_folder   Search within an SB folder

Description
Search for text in the title, abstract, etc. within an SB folder and any subfolders.

Usage
query_item_in_folder(text, folder, ..., session = current_session(),
                     limit = 20)

Arguments
- `text`: text in the title, abstract, etc. of the desired item
- `folder`: an SB item ID for the folder to search in
- `...`: Additional parameters are passed on to GET
- `session`: (optional) SB Session to use, not provided queries public items only
- `limit`: Max number of matching items to return

Value
A list of matching items as sbitem objects.

---

query_sb   Query SB for items using generic query parameters

Description
Generic SB query function to construct advanced queries.

The following is a list of query parameters you can use in the `query_list` parameter.

- `q`: Query string
- `q`: Lucene query string
- `fields`: Character vector of fields to return
- `folderId`: Alphanumeric string representing folder ID
- `parentId`: Alphanumeric string representing folder ID. This can be used to return all children items within the folder, but not within sub-folders.
- `sort`: One of "firstContact", "dateCreated", "lastUpdated", or "title". By default sorted by search score
- `order`: One of "asc" or "desc"
- `ids`: Vector of item ids.
query_sb

- ancestors (character): Alphanumeric string representing folder ID. This can be used to return all children items within the folder, even within sub-folders. Used as a filter
- tags Filter by tags, e.g. "distribution". Used as a filter
- browseCategory One of .... Used as a filter
- browseType One of .... Used as a filter
- dateRange A json string with keys dateType and choice. Where dateType is one of Acquisition, Award, Collected, dateCreated, Received, Reported, Transmitted, Due, End, Info, lastUpdated, Publication, Release, or Start. And where choice is one of day, week, month, year, or range (if range selected, also supply start and end keys with dates of the form YYYY-MM-DD). Used as a filter
- projectStatus One of Active, Approved, Completed, In Progress, Proposed. Used as a filter
- spatialQuery A WKT string. Used as a filter
- extentQuery Use existing extents (footprints) to search against item bounding boxes and representational points. This is a alphanumeric string.

Usage

query_sb(query_list, ..., limit = 20, session = current_session())

Arguments

query_list List of item query selectors. See Details.
... Additional parameters are passed on to GET
limit Maximum number of returned items. Will do paging to retrieve results when limit is over 1000. Use with caution, queries 10k results are slow.
session Session object from authenticate_sb

Value

A list of sbitem objects

See Also

query_items

Examples

## Not run:
query_sb(list(q = "water"))

# Search by project status
query_sb(list(projectStatus = "Active"))

# Search a folder ID
query_sb(list(q = "water", folderId = '504216b9e4b04b508bfd337d'))

# Filter by ancestor
query_sb(list(ancestors = "4f831626e4b0e84f6086809b"))

# Filter by tags
query_sb(list(tags = "distribution"))

# Filter by browse category
query_sb(list(browseCategory = "Image"))

# Filter by browse type
query_sb(list(browseType = "Map Service"))

# Filter by WKT geometry string
wkt1 <- "POLYGON((-104.4 41.0,-95.1 41.0,-95.1 37.5,-104.4 37.5,-104.4 41.0))"
wkt2 <- "POLYGON((-104.4 38.3,-95.2 38.3,-95.2 33.7,-104.4 34.0,-104.4 38.3))"
query_sb(list(spatialQuery = wkt1))
query_sb(list(spatialQuery = wkt1, spatialQuery = wkt2))

# Date range
query_sb(list(dateRange = '{"dataType":"Collected","choice":"year"}'))
query_sb(list(dateRange = '{"dataType":"lastUpdated","choice":"month"}'))
query_sb(list(dateRange = '{"dataType":"Release","choice":"range","start":"2014-09-01","end":"2015-09-01"}'))

## End(Not run)

---

**query_sb_datatype**  
*Query SB for specific data type*

**Description**  
Queries ScienceBase for items with matching datatype.

**Usage**  
```
query_sb_datatype(datatype, ..., limit = 20,
                  session = current_session())
```

**Arguments**

- **datatype**  
  Character string indicating datatype. See `sb_datatypes` for full list of available datatypes.
- **...**  
  Additional parameters are passed on to `GET`
- **limit**  
  Maximum number of returned items. Will do paging to retrieve results when limit is over 1000. Use with caution, queries 10k results are slow.
- **session**  
  Session object from `authenticate_sb`
Value

A list of \texttt{sbitem} objects. List of length 0 means no matches were found.

Examples

```r
#query for items with WFS Layer data
query_sb_datatype('Static Map Image')

#query for US Topo maps
query_sb_datatype('Map Service')
```

---

**query\_sb\_date**  
**Query SB for items within a date range**

Description

Queries ScienceBase for items with timestamps within a certain date/time range.

Usage

```r
query_sb_date(start = as.POSIXct("1970-01-01"), end = Sys.time(),
              date_type = "lastUpdated", ..., limit = 20,
              session = current_session())
```

Arguments

- **start**  
  Start date as \texttt{POSIXct} object. Defaults to 1970-01-01

- **end**  
  End date as \texttt{POSIXct} object. Defaults to today.

- **date\_type**  
  Which object timestamp to query against. Options are (case sensitive): 'Acquisition', 'Award', 'Collected', 'dateCreated', 'Received', 'Reported', 'Transmitted', 'Due', 'End', 'Info', 'lastUpdated', 'Publication', 'Release', 'Repository Created', 'Repository Updated', 'Start'.

- **...**  
  Additional parameters are passed on to \texttt{GET}

- **limit**  
  Maximum number of returned items. Will do paging to retrieve results when limit is over 1000. Use with caution, queries 10k results are slow.

- **session**  
  Session object from \texttt{authenticate\_sb}

Examples

```r
## Not run:
# find items updated today
query_sb_date(Sys.time(), Sys.time())

# find items with publications from the 1970's
query_sb_date(as.POSIXct('1970-01-01'), as.POSIXct('1980-01-01'))
```
query_sb_doi

    query_sb_doi(doi, ..., limit = 20, session = current_session())

Arguments

    doi        DOI to search for as character
    ...    Additional parameters are passed on to GET
    limit    Maximum number of returned items. Will do paging to retrieve results when
              limit is over 1000. Use with caution, queries 10k results are slow.
    session  Session object from authenticate_sb

Value

    A list of sbitem objects. List of length 0 means no matches were found.

Examples

    # Two example DOI-specific queries
    query_sb_doi("10.5066/F7M043G7")

    query_sb_doi("10.5066/F7Z60M35")
**query_sb.spatial**

*Query SB based on spatial extent*

**Description**

Queries ScienceBase based on a spatial bounding box. Accepts either an sp spatial data object (uses the spatial object’s bounding box) or long/lat coordinates defining the bounding box limits.

**Usage**

```r
query_sb_spatial(bbox, long, lat, bb_wkt, ..., limit = 20,
                  session = current_session())
```

**Arguments**

- `bbox` An sp spatial data object. The bounding box of the object is used for the query.
- `long` A vector of longitude values that will define the boundaries of a bounding box. Min and Max of supplied longitudes are used. (alternate option to bbox).
- `lat` A vector of latitude values that will define the boundaries of a bounding box. Min and Max of supplied latitude are used. (alternate option to bbox).
- `bb_wkt` A character string using the Well Known Text (WKT) standard for defining spatial data. Must be a POLYGON WKT object.
- `...` Additional parameters are passed on to `GET`
- `limit` Maximum number of returned items. Will do paging to retrieve results when limit is over 1000. Use with caution, queries 10k results are slow.
- `session` Session object from `authenticate_sb`

**Examples**

```r
#specify the latitude and longitude points to define the bounding box range.
# This is simply bottom left and top right points
query_sb_spatial(long=c(-104.4, -95.1), lat=c(37.5, 41.0), limit=3)

#use a pre-formatted WKT polygon to grab data
query_sb_spatial(bb_wkt="POLYGON((-104.4 41.0,-95.1 41.0,-95.1 37.5,-104.4 37.5,-104.4 41.0))",
                 limit=3)
```
query_sb_text

Query SB for items containing specific text

Description

Queries for ScienceBase items that have matching text in the title or description

Usage

query_sb_text(text, ..., limit = 20, session = current_session())

Arguments

text Text string for search

... Additional parameters are passed on to GET

limit Maximum number of returned items. Will do paging to retrieve results when limit is over 1000. Use with caution, queries 10k results are slow.

session Session object from authenticate_sb

Value

A list of sbitem objects. List of length 0 means no matches were found.

Examples

# query for the package maintainer's name
query_sb_text('Luke Winslow')

# query for one of the old river gaging stations
query_sb_text('Lees Ferry')

sbitem

ScienceBase item class

Description

ScienceBase item class

Usage

as.sbitem(x, ...)

## Default S3 method:
as.sbitem(x, ...)

is.sbitem(x)
sb_datatypes

Arguments

x  
Input, variety of things, character, list, or sbitem class object

...  
Further args passed on to `item_get`, only in the method for character class inputs

Examples

```r
# Single item from item_get()
item_get("4f4e4b24e4b07f02dbb6ae14")

# Get many w/ e.g., an lapply() call
library("httr")
res <- query_items(list(s = "Search", q = "water", format = "json"))
ids <- vapply(httr::content(res)$items, "[[", "", "id")
(out <- lapply(ids[1:3], item_get))

# create item class from only an item ID
as.sbitem("4f4e4b24e4b07f02dbb6ae14")

# sbitem gives back itself
(x <- as.sbitem("4f4e4b24e4b07f02dbb6ae14"))
as.sbitem(x)
```

sb_datatypes

Query SB for all available datatypes

Description

Queries ScienceBase for the list of all available datatypes. This can be coupled with `query_sb_datatype` to query based on the type of data

Usage

```r
sb_datatypes(limit = 50, session = current_session())
```

Arguments

limit  
Maximum number of returned items. Will do paging to retrieve results when limit is over 1000. Use with caution, queries 10k results are slow.

session  
Session object from `authenticate_sb`

Examples

```r
#return all datatypes (limit 50 by default)
sb_datatypes()
```
**sb_ping**

Ping ScienceBase to see if it’s available

**Usage**

```
sb_ping(...)```

**Arguments**

`...` Additional parameters are passed on to `GET`

**Value**

Boolean (TRUE) indicating if a connection to ScienceBase can be established and if it is responding as expected. FALSE otherwise.

**Examples**

```
#TRUE if all is well and SB can be contacted
sb_ping()
```

---

**session_details**

Get session info

**Description**

Get the details associated with current ScienceBase user session.

**Usage**

```
session_details(..., session = current_session())```

**Arguments**

`...` Additional parameters are passed on to `GET`

`session` SB session object from `authenticate_sb`

**Value**

list, if not logged in states that, but if logged in, user details
session_renew

Examples

## Not run:
session_info()

## End(Not run)

session_logout

Logout of a ScienceBase session

Description

Logout of a ScienceBase session

Usage

session_logout(..., session = current_session())

Arguments

... Additional parameters are passed on to GET
session SB session object from authenticate_sb

Value

invisible, returns nothing if logged out, or errors with message

Examples

## Not run:
session_logout()

## End(Not run)

session_renew

Checks current session and re-authenticates if necessary

Description

Checks the state of your Sciencebase session, re-authenticates if the session is expired, and simply renews if the session is active.

Usage

session_renew(password, ..., username, session = current_session())
session_validate

Validate sbtools session state

Description

A session is considered valid if it is NULL or a true, non-expired SB session

Usage

```
session_validate(session = current_session())
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>sbtools session object (from authenticate_sb)</td>
</tr>
</tbody>
</table>

Details

This validates the underlying RCurl session. The session object becomes invalid if the R session has been saved to disk or persisted through an R restart. This verifies that the session object is either valid, or is a NULL object, which means no session state is being persisted. Note, this does not verify the credentials are valid or that you have permission to access the SB item, so it does not guarantee a successful request.
Value

TRUE/FALSE indicating if session is valid and can be used. Returns TRUE if session is NULL as well.

Examples

```r
## Not run:
session = authenticate_sb('user@usgs.gov')

#return true as underlying RCurl session is valid
session_validate(session)
## End(Not run)
```

---

**set_endpoint**

*Set SB endpoint*

Description

Sets the internal URLS used to either the production or development (beta) SB server. URLS are stored internally to the package.

Usage

```r
set_endpoint(endpoint = c("production", "development"))
```

Arguments

- `endpoint` Indicate which SB endpoint you want to use options: c('production','development')

Author(s)

Luke Winslow

Examples

```r
## Not run:
set_endpoint('prod')

# getting item from production SB servers
item_get('54b94448e4b0ed73b67550ea')

set_endpoint('dev')
# getting item from beta SB servers
item_get('54b94448e4b0ed73b67550ea')
```
user_id

## Get your parent ID

**Description**

Required for creating items

**Usage**

user_id(...) session = current_session()

**Arguments**

... Additional parameters are passed on to POST

session Session object from authenticate_sb

**Value**

A single character string, your user id

**Examples**

## Not run:

user_id()

## End(Not run)
Index

Topic package
  sbtools-package, 3
as.sbitem, 18
as.sbitem(sbitem), 35
authenticate_sb, 3, 3, 4–25, 30–39, 41
current_session, 4
DELETE, 4, 5, 7, 9–24
folder_create, 4
GET, 4–7, 9–25, 28–35, 37, 38
HEAD, 4, 5, 7, 9–24
identifier_exists, 5
is.sbitem(sbitem), 35
is_logged_in, 6
item_append_files, 10, 19, 20
item_create, 3, 4, 10
item_exists, 11
item_file_download, 3, 12, 17
item_get, 3, 13, 36
item_get_fields, 14
item_get_parent, 3, 15
item_get_wfs, 3, 15
item_list_children, 3, 16
item_list_files, 17
item_move, 18
item_rename_files, 19
item_replace_files, 19
item_rm, 3, 20
item_rm_files, 21
item_update, 3, 22
item_update_identifier, 22
item_upload_create, 23
item_upsert, 24
items_create, 6
items_update, 8
items_upsert, 9
POSIXct, 32
POST, 4, 5, 7, 9–24, 41
PUT, 4, 5, 7–24
query_item_identifier, 22, 26, 28
query_item_in_folder, 26, 29
query_items, 25, 30
query_sb, 29
query_sb_datatype, 31, 36
query_sb_date, 32
query_sb_doi, 3, 33
query_sb_spatial, 3, 34
query_sb_text, 3, 35
response, 4, 20, 26
sb_datatypes, 31, 36
sb_ping, 37
sbitem, 4, 5, 7–24, 30, 32, 33, 35, 35
sbtools(sbtools-package), 3
sbtools-package, 3
session_details, 37
session_logout, 38
session_renew, 38
session_validate, 39
set_endpoint, 40
user_id, 41