Package ‘sbtools’

May 1, 2023

Title USGS ScienceBase Tools
Maintainer David Blodgett <dblodgett@usgs.gov>
Version 1.2.0
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), methods, mime, utils, cli,
keyring
Suggests testthat, xml2, sf, sp, tools
License CC0
URL https://github.com/DOI-USGS/sbtools
BugReports https://github.com/DOI-USGS/sbtools/issues
RoxygenNote 7.2.3
Encoding UTF-8
NeedsCompilation no
Author David Blodgett [cre],
Luke Winslow [aut],
Scott Chamberlain [ctb],
Alison Appling [ctb],
Jordan Read [ctb]
Repository CRAN
Date/Publication 2023-04-30 23:00:02 UTC

R topics documented:

sbtools-package .......................................................... 3
authenticate_sb ....................................................... 3
current_session ...................................................... 4
folder_create ......................................................... 4
<table>
<thead>
<tr>
<th>topic</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>identifier_exists</td>
<td>5</td>
</tr>
<tr>
<td>is_logged_in</td>
<td>6</td>
</tr>
<tr>
<td>items_create</td>
<td>6</td>
</tr>
<tr>
<td>items_update</td>
<td>8</td>
</tr>
<tr>
<td>items_upsert</td>
<td>9</td>
</tr>
<tr>
<td>item_append_files</td>
<td>10</td>
</tr>
<tr>
<td>item_create</td>
<td>11</td>
</tr>
<tr>
<td>item_exists</td>
<td>12</td>
</tr>
<tr>
<td>item_file_download</td>
<td>12</td>
</tr>
<tr>
<td>item_get</td>
<td>14</td>
</tr>
<tr>
<td>item_get_fields</td>
<td>15</td>
</tr>
<tr>
<td>item_get_parent</td>
<td>15</td>
</tr>
<tr>
<td>item_list_children</td>
<td>16</td>
</tr>
<tr>
<td>item_list_files</td>
<td>17</td>
</tr>
<tr>
<td>item_move</td>
<td>18</td>
</tr>
<tr>
<td>item_publish_cloud</td>
<td>19</td>
</tr>
<tr>
<td>item_rename_files</td>
<td>20</td>
</tr>
<tr>
<td>item_replace_files</td>
<td>21</td>
</tr>
<tr>
<td>item_rm</td>
<td>22</td>
</tr>
<tr>
<td>item_rm_files</td>
<td>23</td>
</tr>
<tr>
<td>item_update</td>
<td>24</td>
</tr>
<tr>
<td>item_update_identifier</td>
<td>24</td>
</tr>
<tr>
<td>item_upload_cloud</td>
<td>25</td>
</tr>
<tr>
<td>item_upload_create</td>
<td>26</td>
</tr>
<tr>
<td>item_upsert</td>
<td>27</td>
</tr>
<tr>
<td>query_items</td>
<td>28</td>
</tr>
<tr>
<td>query_item_identifier</td>
<td>31</td>
</tr>
<tr>
<td>query_item_in_folder</td>
<td>32</td>
</tr>
<tr>
<td>query_sb</td>
<td>33</td>
</tr>
<tr>
<td>query_sb_datatype</td>
<td>35</td>
</tr>
<tr>
<td>query_sb_date</td>
<td>36</td>
</tr>
<tr>
<td>query_sb_doi</td>
<td>37</td>
</tr>
<tr>
<td>query_sb_spatial</td>
<td>37</td>
</tr>
<tr>
<td>query_sb_text</td>
<td>38</td>
</tr>
<tr>
<td>sbitem</td>
<td>39</td>
</tr>
<tr>
<td>sb_datatypes</td>
<td>40</td>
</tr>
<tr>
<td>sb_ping</td>
<td>41</td>
</tr>
<tr>
<td>session_details</td>
<td>41</td>
</tr>
<tr>
<td>session_logout</td>
<td>42</td>
</tr>
<tr>
<td>session_renew</td>
<td>43</td>
</tr>
<tr>
<td>session_validate</td>
<td>44</td>
</tr>
<tr>
<td>set_endpoint</td>
<td>45</td>
</tr>
<tr>
<td>user_id</td>
<td>46</td>
</tr>
</tbody>
</table>

**Index**

47
sbtools-package

Description

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://github.com/DOI-USGS/sbtools/issues).

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_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_list_children`, `item_create`, `item_update`, `item_rm`).

Feedback

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

---

**authenticate_sb**

*Authenticate 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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Sciencebase username</td>
</tr>
<tr>
<td>password</td>
<td>Sciencebase password, prompts user if not supplied and no password is returned by ‘keyring::key_get(&quot;sciencebase&quot;, username)’. See <a href="https://github.com/DOI-USGS/sbtools/issues">keyring-package</a> documentation for more details.</td>
</tr>
</tbody>
</table>
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
current_session()

Examples

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) but defaults to showing child-items on the ScienceBase web interface.

Usage
folder_create(parent_id = user_id(), name, ..., session = current_session())

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent_id</td>
<td>An sbitem object or character ScienceBase ID corresponding to the parent item (folder)</td>
</tr>
<tr>
<td>name</td>
<td>(character) the folder name</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 authenticated session</td>
</tr>
</tbody>
</table>

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
## 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

## 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

**Upsert many SB items**

**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")))
)```
item_append_files

Upload File to Item

Description

Adds a file to an item

Usage

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

Arguments

- `sb_id` An `sbitem` object or a character ScienceBase ID corresponding to the item
- `files` A string vector of paths to files to be uploaded
- `...` Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`
- `scrape_files` logical should the files be scraped for metadata? If TRUE, sciencebase will attempt to create extensions based on the files. For example, for shapefiles, this will result in a shapefile extension to be returned as a facet of the sciencebase item. See item: "58069258e4b0824b2d1d422e" for an example.
- `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 = "foobartxt")
item_append_files(res$id, "foobartxt")
item_rm(res)

## 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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent_id</td>
<td>An sbitem object or character ScienceBase ID corresponding to the parent item (folder)</td>
</tr>
<tr>
<td>title</td>
<td>The title of the new SB item</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters are passed on to GET, POST, HEAD, PUT, or DELETE</td>
</tr>
<tr>
<td>info</td>
<td>(optional) list of metadata info for the new item</td>
</tr>
<tr>
<td>session</td>
<td>Session object from authenticate_sb. Defaults to anonymous or last authenticated session</td>
</tr>
</tbody>
</table>

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 touple already exists on SB

Description

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

Usage

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

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)

item_file_download

Download files attached to item

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

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

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`
- **names**: String vector list of file names attached to item that you wish to download.
- **destinations**: String vector list of destinations for requested files. Must be same length as names.
- **dest_dir**: A directory path for saving files when `names` destinations parameter is not specified.
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session.
- **overwrite_file**: Boolean indicating if file should be overwritten if it already exists locally.

Value

Character vector of full paths to local files

Examples

```r
## Not run:
#downloads all files attached to this item
item_file_download('627f1572d34e3bef0c9a30d8', dest_dir=tempdir())

#downloads a specific file attached to this item
item_file_download('627f1572d34e3bef0c9a30d8', names='example.txt',
  destinations=file.path(tempdir(), 'out.txt'))

## End(Not run)
```
item_get

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"))
if(inherits(res, "response") && res$status != 404) {
  ids <- vapply(httr::content(res)$items, "[[", "", "id")
  lapply(ids[1:3], item_get)
}
```
**item_get_fields**

*Retrieve specific fields from an SB item*

**Description**

Retrieve specific fields from an SB item

**Usage**

```r
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)
```

---

**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())
```
item_list_children

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sb_id</td>
<td>An <code>sbitem</code> object or a character ScienceBase ID corresponding to the item</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters are passed on to <code>GET, POST, HEAD, PUT, or DELETE</code></td>
</tr>
<tr>
<td>session</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_parent(item_get("4f4e4b24e4b07f02db6aea14"))
```

---

**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>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sb_id</td>
<td>An <code>sbitem</code> 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 <code>GET, POST, HEAD, PUT, or DELETE</code></td>
</tr>
<tr>
<td>session</td>
<td>Session object from <code>authenticate_sb</code>. Defaults to anonymous or last authenticated 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())

## End(Not run)

item_list_children(as.sbitem('5060b03ae4b00fc20c4f3c8b'))
item_list_children(item_get('5060b03ae4b00fc20c4f3c8b'))
```

---

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.

NOTE: A sciencebase item can contain so-called "extensions". The sciencebase item data model refers to the information that describes an extension as a "facet". Some extension facets contain files (such as with a shapefile). The "facet" attribute of the return from this function will contain the name of the facet the file came from if the file was found in a facet.

Usage

```r
item_list_files(
  sb_id,
  recursive = FALSE,
  fetch_cloud_urls = TRUE,
  ...,  
  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
- **fetch_cloud_urls** (logical): fetch a tokenized cloud download URLs? Default: TRUE This option will take slightly longer but the 'url' attribute of the returned list will work for direct file downloads or use with other applications and libraries.
- **...**: 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, url, and facet. If item has no attached files, returns a zero row data.frame.

Examples

## Not run:

#regular files
item_list_files("4f4e4b24e4b07f02db6aea14")

# files in facets
item_list_files("5f6a285d82ce38aaa244912e")

# 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 = '56562348e4b071e7ea53e09d', recursive = FALSE) # default
item_list_files(id = '56562348e4b071e7ea53e09d', recursive = TRUE)

## End(Not run)
Arguments

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item.
- **id_new**: Folder/item to move id to. A ScienceBase ID or something that can be coerced to a SB item ID by `as.sbitem`.
- **...**: 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`. 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(res2$parentId, fold2$id)
## End(Not run)
```

---

**item_publish_cloud**

Publish file to public cloud S3 bucket

Description

moves a cloud file from the S3 bucket only available via ScienceBase authenticated services to a public S3 bucket.

Usage

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

Arguments

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item.
- **files**: A string vector of paths to files to be uploaded.
- **...**: 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.
web service response invisibly.

Examples

```r
## Not run:
res <- item_create(user_id(), "testing 123")
cat("foo bar", file = "foobar.txt")
item_upload_cloud(res$bid, "foobartxt")
item_publish_cloud(res$bid, "foobartxt")

## End(Not run)
```

---

**item_rename_files**  
Renam 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.

NOTE: This function will not replace files stored in facets. Until and if facet support is added, direct alteration of the science base item object is required to manipulate facets.

Usage

```r
item_replace_files(
  sb_id,
  files,
  ...,
  all = FALSE,
  scrape_files = 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.
- **scrape_files**: logical should the files be scraped for metadata? If TRUE, sciencebase will attempt to create extensions based on the files. For example, for shapefiles, this will result in a shapefile extension to be returned as a facet of the sciencebase item. See item: "58069258e4b0824b2d1d422e" for an example.
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session
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

- `httr 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.

NOTE: This function will not alter facets which can also contain facets. To manipulate facets, the facet element of a sciencebase item must be altered and updated with item_update.

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

Usage

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

## 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**

```r
## 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**

```{r}
item_update_identifier(
  sb_id,
  scheme,
  type,
  key,
  ...
  session = current_session()
)
```

**Arguments**

- `sb_id` An `sbitem` object or a character ScienceBase ID corresponding to the item
- `scheme` The identifier scheme
- `type` The identifier type
- `key` The identifier key
- `...` 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:

session = authenticate_sb("user@usgs.gov")
item_update_identifier("5485fd99e4b02acb4f0c7e81", "scheme", "type", "key", session=session)

## End(Not run)
```

---

**item_upload_cloud**  
**Upload File to Item Cloud Storage**

**Description**

Adds a file to an item in cloud storage

**Usage**

```{r}
item_upload_cloud(
  sb_id,
  files,
  ...
  status = TRUE,
  session = current_session()
)
```
Arguments

- **sb_id**: An `sbitem` object or a character ScienceBase ID corresponding to the item.
- **files**: A string vector of paths to files to be uploaded.
- **...**: Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`.
- **status**: Logical display upload status?
- **session**: Session object from `authenticate_sb`. Defaults to anonymous or last authenticated session.

Value

Success message invisibly. NOTE: cloud processing can take some time so the added file may not appear immediately. For this reason, a sciencebase item json is NOT returned as is done with other similar functions.

Examples

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

Description

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

Usage

```r
item_upload_create(
  parent_id,
  files,
  ...
  scrape_files = TRUE,
  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.
- **...**: Additional parameters are passed on to `GET`, `POST`, `HEAD`, `PUT`, or `DELETE`.
**item_upsert**

Upsert an SB item

**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
- `info` (optional) list of metadata info for the new item
- `session` Session object from **authenticate_sb**. Defaults to anonymous or last authenticated session

**Examples**

```r
## Not run:
# You'll need a parent id for a folder/item
# here, using your highest level parent folder
file <- system.file("examples", "books.json", package = "sbtools")
item_upload_create(user_id(), file)

## End(Not run)
```
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 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"))
))

## 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
query_items

- 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
- 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 an 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))
http::content(res)

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

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

# Search a folder ID
res <- query_items(list(s = "Search", q = "water", format = "json", folderId = '504216b9e4b04b04b508bfd337d'))
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")))
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.
Query Item In Folder

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', 'key1')

#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** (character): Query string
- **q** (character): Lucene query string
- **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
- **browseCategory** One of .... Used as a filter
- **browseType** One of .... Used as a filter
- **dateRange** A json string with keys `dateTimeType` and `choice`. Where `dateTimeType` 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**

```r
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 = "4f831626e4b0e84f6806809b"))

# 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 = '{"dateType":"Collected","choice":"year"}'))

query_sb(list(dateRange = '{"dateType":"lastUpdated","choice":"month"}'))

query_sb(list(dateRange = '{"dateType":"Release","choice":"range","start":"2014-09-01","end":"2015-09-01"}'))
query_sb_datatype

## End(Not run)

### query_sb_datatype

**Query SB for specific data type**

#### Description

Queries ScienceBase for items with matching datatype.

#### Usage

```r
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 `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

query_sb_date(
  start = as.POSIXct("1970-01-01"),
  end = Sys.time(),
  date_type = "lastUpdated",
  ...,  # Additional parameters are passed on to GET
  limit = 20,
  session = current_session()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Start date as POSIXct object. Defaults to 1970-01-01</td>
</tr>
<tr>
<td>end</td>
<td>End date as POSIXct object. Defaults to today.</td>
</tr>
<tr>
<td>date_type</td>
<td>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'.</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters are passed on to GET</td>
</tr>
<tr>
<td>limit</td>
<td>Maximum number of returned items. Will do paging to retrieve results when limit is over 1000. Use with caution, queries 10k results are slow.</td>
</tr>
<tr>
<td>session</td>
<td>Session object from authenticate_sb</td>
</tr>
</tbody>
</table>

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"),
  date_type="Publication", limit=1000)

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

*Query SB for specific DOI (Digital Object Identifier)*

**Description**

Queries for ScienceBase items with a specific DOI identifier. In ScienceBase, these are stored as additional unique identifiers.

**Usage**

```r
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**

```r
# 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

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

#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

Description

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

Usage

query_sb.text(text, ..., limit = 20, session = current_session())
sbitem

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

```r
# query for a person's name
query_sb_text("Luna Leopold")

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

---

**sbitem**  
*ScienceBase item class*

Description

ScienceBase item class

Usage

```r
as.sbitem(x, ...)

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

is.sbitem(x)
```

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("4f4e4b24e4b07f02db6aea14")

# Get many w/ e.g., an lapply() call
library("httr")
res <- query_items(list(s = "Search", q = "water", format = "json"))
if(res$status != 404) {
  ids <- vapply(httr::content(res)$items, "[[", "", "id")
  (out <- lapply(ids[1:3], item_get))
}
# create item class from only an item ID
as.sbitem("4f4e4b24e4b07f02db6aea14")

# sbitem gives back itself
(x <- as.sbitem("4f4e4b24e4b07f02db6aea14"))
as.sbitem(x)
```

---

### sb_datatypes

**Query SB for all available datatypes**

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
## Not run:
#return all datatypes (limit 50 by default)
sb_datatypes()

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

*Ping ScienceBase to see if it’s available*

**Description**

Ping ScienceBase to see if it’s available

**Usage**

```r
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**

```r
#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**

```r
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 If logged in, will include a "jossoSessionId" that can be used for authenticated file downloads by appending josso='jossoSessionId' to the url for the file requiring authentication. This is helpful when passing a url to a library that will handle the download of a remote file automatically such as jsonlite::readJSON or readr::read_csv.

Examples

```r
## Not run:

session_details()

# If logged in, can use jossoSessionId for downloads.
authenticate_sb()

temp_json <- tempfile(fileext = "json")

jsonlite::write_json(list(test = "test"), temp_json)

item <- item_upload_create(sbtools::user_id(), temp_json)

token <- session_details()$jossoSessionId

(base_url <- item$file[[1]]$downloadUri)

# will fail
try(jsonlite::read_json(base_url))

url <- paste0(base_url, "&josso=", token)

jsonlite::read_json(url)

item_rm(item$id)

## End(Not run)
```

---

**session_logout**

*Logout of a ScienceBase session*

Description

Logout of a ScienceBase session

Usage

`session_logout(., session = current_session())`
**session_renew**

**Arguments**

- *password*  
  The password to use, if needed, to renew the session.
- *...*  
  Any additional parameters are currently ignored.
- *username*  
  Optional. Used only to confirm that the current username is what you expect; if you want to switch usernames, use `authenticate_sb()` instead of this function.
- *session*  
  SB session object from `authenticate_sb`. Default is the current session.

**Value**

Returns the session object.
Examples

## Not run:

# an empty call is sufficient if the session is current,
# but will break if haven't been logged in before
session_renew()

# include a password if session may be expired
session_renew('newpass')

# optionally confirm the value of the current username
session_renew(username='olduser@usgs.gov', 'newpass')

## End(Not run)

---

**session_validate**

**Validate sbtools session state**

### Description

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

### Usage

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

### Arguments

- `session` sbtools session object (from `authenticate_sb`)

### 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

## Not run:

```r
session = authenticate_sb('user@usgs.gov')
```

#return true as underlying RCurl session is valid
```r
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
set_endpoint('prod')
# getting item from production SB servers
item_get('50f60b03ae4b00fc20c4f3c8b')

set_endpoint('dev')
# getting item from beta SB servers
item_get('521e46864b051c878dc35d0')
```
**user_id**  
*Get your parent ID*

---

### Description

Required for creating items

### Usage

```r
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

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

## End(Not run)
```
## Index

**package**
- sbtools-package, 3

as.sbitem, 19  
as.sbitem(sbitem), 39  
authenticate_sb, 3, 3–11, 13–17, 19–28, 34–41, 43, 44, 46

**current_session**, 4

DELETE, 4, 5, 7, 9–11, 13–17, 19–27

**folder_create**, 4


HEAD, 4, 5, 7, 9–11, 13–17, 19–27

**identifier_exists**, 5

is.sbitem(sbitem), 39

is_logged_in, 6

item_append_files, 10, 21

item_create, 3, 4, 11

item_exists, 12

item_file_download, 3, 12, 17

item_get, 3, 14, 39

item_get_fields, 15

item_get_parent, 3, 15

item_list_children, 3, 16

item_list_files, 17

item_move, 18

item_publish_cloud, 19

item_rename_files, 20

item_replace_files, 21

item_rm, 3, 22

item_rm_files, 23

item_update, 3, 23, 24

item_update_identifier, 24

item_upload_cloud, 25

item_upload_create, 26

item_upsert, 27

**items_create**, 6

**items_update**, 8

**items_upsert**, 9

**POST**, 4, 5, 7, 9–11, 13–17, 19–27, 46

**PUT**, 4, 5, 7–11, 13–17, 19–27

query_item_identifier, 24, 29, 31

query_item_in_folder, 29, 32

query_items, 28, 34

query_sb, 33

query_sb_datatype, 35, 40

query_sb_date, 36

query_sb_doi, 3, 37

query_sb.spatial, 3, 37

query_sb.text, 3, 38

response, 4, 22, 29

**sb_datatypes**, 35, 40

**sb_ping**, 41

**sbitem**, 4, 5, 7–11, 13–17, 19–27, 34, 35, 37, 39, 39

**sbtools** (sbtools-package), 3

sbtools-package, 3

**session_details**, 41

**session_logout**, 42

**session_renew**, 43

**session_validate**, 44

**set_endpoint**, 45

user_id, 46