iformr

May 2, 2019

Type Package

Title A set of tools to leverage the iFormBuilder API

Version 0.1.0.9009

Description A collection of tools to connect R to the iFormBuilder API. For now, these functions allow the most basic operations such as creating new option lists and downloading data. Additional functions will be added as time allows.

License MIT + file LICENSE

Depends R (>= 3.4.0)

Imports curl (>= 2.7),
httr (>= 1.2.1),
jsonlite (>= 1.5),
lubridate (>= 1.6.0),
openssl (>= 0.9.6),
dplyr (>= 0.7.1),
stringi (>= 1.1.5),
tibble (>= 1.4.2),
stringr (>= 1.3.0),
methods (>= 3.5.0)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

R topics documented:

`add_options_to_list` *Add option values to new option list*

Description

Sends a request to the iFormBuilder API to append a list of options in json format to an existing option list. Make sure that all options list key_values are unique or the list will not be posted.

Usage

```
add_options_to_list(server_name, profile_id, optionlist_id, option_values,
  access_token)
```

Arguments

`server_name`	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
`profile_id`	The id number of your profile
`optionlist_id`	The id number for the option list
`option_values`	A list of option values in json format
`access_token`	The access_token required to establish communication with the API

Value

A vector of option list element ids, one for each option in the list

Examples

```
# The locations dataset is an example of a segmented option list
head(locations, 5)

# Convert locations dataframe to json
location_json <- jsonlite::toJSON(locations, auto_unbox = TRUE)

## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Create a new empty option list
new_option_list_id <- create_new_option_list(
  server_name = "your_server_name",
  profile_id = 123456,
  option_list_name = "SGS-StreamLocations",
  access_token = access_token)

# Inspect new option list id
new_option_list_id

# Add option elements from locations dataset to the new option list
option_ids <- add_options_to_list(
  server_name = "your_server_name",
  profile_id = 123456,
  optionlist_id = new_option_list_id,
  option_values = location_json
  access_token = access_token)

# Inspect the first five new option list element ids.
head(option_ids, 5)

## End(Not run)
```

---

copy_page *Copy page*

Description

Copies a page to a new page in the profile.

Usage

```
copy_page(server_name, profile_id, access_token, page_id)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	Integer of the page ID to copy.

Value

Integer of the new page ID.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Copy page
new_page_id <- copy_page(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  page_id = "existing_page_id"

## End(Not run)
```

---

create_element *Add element to page.*

Description

Adds a new element to a page.

Usage

```
create_element(server_name, profile_id, access_token, page_id, name, label,
  description = "", data_type, data_size = 100, optionlist_id)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	Page ID where the new element will be created.

name	String of the element DCN. The provided name will be converted to be IFB database compliant (no special characters, all lowercase, spaces replaced with underscores.)
label	Label for the element.
description	Text description of the element.
data_type	Integer indicating data type of the element.
data_size	*Optional* - length of the element; defaults to 100.
optionlist_id	*Optional* - id of the option list to assign if a Select, Picklist, or Multi Picklist element is created.

Value

ID of the new element.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

See Also

List of iFormBuilder data types: https://iformbuilder.zendesk.com/hc/en-us/articles/201702880-Data-Types-and-Related-Details

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Create element
new_element <- create_element(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  page_id = "existing_page_id",
  name = "new_dcn_name",
  label = "new_element_label",
  description = "This is a new element.",
  data_type = 1

## End(Not run)
```

create_new_option_list *Create a new option list*

Description

Sends a request to the iFormBuilder API to create a new option list. The new option list will be created with the name you supply.

Usage

```
create_new_option_list(server_name, profile_id, option_list_name,
  access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
option_list_name	A character name for the new option list
access_token	The access_token required to establish communication with the API

Value

The id of the new option list

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Create a new empty option list
new_option_list_id <- create_new_option_list(
  server_name = "your_server_name",
  profile_id = 123456,
  option_list_name = "SGS-Streams",
  access_token = access_token)

# Inspect new option list id
option_list_id
# If successful, result should look something like:
# [1] 4423966

## End(Not run)
```

create_new_records *Create new records*

Description

Creates new records in an iFormBuilder page.

Usage

```
create_new_records(server_name, profile_id, page_id, access_token,
  record_data)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
page_id	Integer ID of the page to insert new records.
access_token	Access token produced by get_iform_access_token
record_data	A dataframe containing the data to be added to the page.

Value

The created record ID or IDs.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

## End(Not run)
```

create_page *Create page*

Description

Creates a new page in the given profile with the name and label specified. The name provided will be converted to iFormBuilder standards; punctuation and whitespace replaced with _ and all text to lowercase.

Usage

```
create_page(server_name, profile_id, access_token, name, label)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
name	String of new page name; coerced to iFormBuilder table name conventions.
label	String of the label for the new page.

Value

Integer of the new page ID.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Create new page
new_page_id <- create_page(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  name = "new_form_name",
  label = "New Form Label")

## End(Not run)
```

---

data2form *Create form from dataframe*

Description

Creates a form based on a dataframe. Dataframe classes are cast as element types in the form.

Usage

```
data2form(server_name, profile_id, access_token, name, label, data)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
name	String of new page name; coerced to iFormBuilder table name conventions.
label	String of the label for the new page.
data	A dataframe whose structure will be used to create the new form.

Value

The page ID of the created form.

Author(s)

Bill DeVoe, <William.DeVoe@maine.gov>

Examples

```
# Create a dataframe with some basic form fields
dat = tibble::tibble(survey_id = NA_integer_,
                     survey_datetime = as.POSIXct(NA, tz = "UTC"),
                     surveyor = NA_character_,
                     start_point = NA_real_,
                     fish_species = NA_integer_,
                     fish_count = NA_integer_,
                     end_point = NA_real_,
                     comment_text = NA_character_,
                     survey_completed = TRUE)

## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Create new form from dataframe
new_form <- data2form(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  name = "new_form_to_create",
  label = "New form based on an R dataframe",
  data = dat)

## End(Not run)
```

`data_feed_url` *Compose a url to get data via the data feed mechanism*

Description

Creates a url with username and password embedded that can be used to download data using the data-feed mechanism instead of the API. In general, this should be avoided, as the API mechanisms are much safer. Returns a json file with all records submitted since the specified `since_id`.

Usage

```
data_feed_url(server_name, parent_form_id, profile_id, parent_form_name,
  since_id, user_label, pw_label)
```

Arguments

`server_name`	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
`parent_form_id`	The id of the parent form
`profile_id`	The id number of your profile
`parent_form_name`	The name of the parent form
`since_id`	The record id indicating where to start downloading
`user_label`	The name given to the username in the .Renviron file
`pw_label`	Skips the offset number of records before beginning to return records

Value

A url that can be used to request form data

Examples

```
## Not run:
# Generate a url to retrieve data via the data feed mechanism
url <- data_feed_url(
  server_name = "your_server_name",
  parent_form_id = 456789,
  profile_id = 123456,
  parent_form_name = "spawning_ground_p",
  since_id = 3,
  user_label = "your_user_label",
  pw_label = "your_pw_label")

# Retrieve the form data into an R list
form_data <- jsonlite::fromJSON(url)

## End(Not run)
```

delete_options_in_list

Delete all or some options in an option list

Description

Sends a request to the iFormBuilder API to delete a list of option elements. The elements to delete are specified by a .json list of element ids. Sort order will automatically be reassigned after deleting specified elements.

Usage

```
delete_options_in_list(server_name, profile_id, optionlist_id,
  fields = "fields", id_values, limit = 1000, offset = 0,
  access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
optionlist_id	The id number for the option list
fields	Placeholder for fields to delete, not yet implemented
id_values	A .json list of ids for elements to delete
limit	The maximum number of option elements to delete
offset	Skips the offset number of options before beginning to delete
access_token	The access_token required to establish communication with the API

Value

A vector of option list elements that were deleted

Examples

```
## Not run:
# Define .json list of ids for elements to delete
# Replace example values below with your own
  id_values = data_frame(id = c(663487010, 663487013))
  id_values_json = jsonlite::toJSON(id_values, auto_unbox = TRUE)

# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Delete specified elements from option list
deleted_ids <- delete_options_in_list(
  server_name = "your_server_name",
  profile_id = 123456,
  optionlist_id = your_option_list_id,
  id_values = id_values_json,
```

```
  access_token = access_token)

# Inspect the first five deleted ids
head(deleted_ids, 5)

## End(Not run)
```

`delete_option_list` *Delete option list.*

Description

Deletes an option list from a profile. Deleting options and option lists should be done with consideration for existing data referencing the list. As an alternative, options can be disabled by setting their condition value to 'False'

Usage

```
delete_option_list(server_name, profile_id, access_token, option_list_id)
```

Arguments

`server_name`	String of the iFormBuilder server name.
`profile_id`	Integer of the iFormBuilder profile ID.
`access_token`	Access token produced by `get_iform_access_token`
`option_list_id`	ID of the option list to be deleted.

Value

ID of the option list to be deleted.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Delete option list
deleted_id <- delete_option_list(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  option_list_id

## End(Not run)
```

delete_record *Delete record*

Description

Delete a single record.

Usage

```
delete_record(server_name, profile_id, access_token, page_id, record_id)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	Integer - ID of the page from which to delete the record.
record_id	Integer - ID of the record to delete.

Value

ID of the record deleted.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

delete_records *Delete records*

Description

Delete a list of records.

Usage

```
delete_records(server_name, profile_id, access_token, page_id, record_ids)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	ID of the page from which to delete the record.
record_ids	Integer vector of the record IDs to delete.

Value

Integer vector of the deleted record IDs.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

`format_name` *Format a page or element name to be IFB compliant*

Description

Replaces whitespace and punctuation in an element/form name with _ and converts name to lowercase. Checks name against list of IFB reserved words, appending a '2' after the name if it is in the reserved word list.

Usage

```
format_name(name)
```

Arguments

`name`	String of new page or element name.

Value

IFB compliant name.

Author(s)

Bill DeVoe, <William.DeVoe@maine.gov>

`form_metadata` *Form metadata*

Description

Builds a Markdown document containing metadata for a given iFormBuilder form by querying the API for page and element level information. By utilizing the description fields during form building, detailed metadata can be built afterward using this function.

Usage

```
form_metadata(server_name, profile_id, access_token, page_id, filename,
  subforms = T, sub = F)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	ID of the form to get metadata from.
filename	Filename of the output Markdown file.
subforms	**Optional** - Indicates if metadata should be generated for subforms. Defaults to True.
sub	**Optional** - Defaults to False. Used by function to self-reference and append subform metadata to beginning file.

Value

Add this later.

Author(s)

Bill DeVoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Create metadata for form.
form_metadata(server_name, profile_id, access_token,
                page_id = 012345, filename = "metadata", subforms = T)

## End(Not run)
```

---

get_all_option_lists *Retrieve all option lists.*

Description

Retrieves all option lists in a profile in chunks of 100 (API call limit).

Usage

```
get_all_option_lists(server_name, profile_id, access_token)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token

Value

Tibble of two columns containing the option list IDs and option list names: id <int>, name <chr>

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id and name of all option lists in profile
option_lists <- get_all_option_lists(
  server_name = "your_server_name",
  profile_id = 123456,
  access_token = access_token)

# Inspect
option_lists

## End(Not run)
```

---

get_all_pages_list *Get list of all pages (i.e., form, or subform) in a profile*

Description

Retrieves a list of ALL pages in a profile, in chunks of 100 (limit). Returns a tibble with form id and form name.

Usage

```
get_all_pages_list(server_name, profile_id, access_token)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token

Value

Tibble of two columns containing the page ID and page name: id <int>, name <chr>

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

`get_all_records` *Get all records for a set of fields in a page (i.e., form, or subform)*

Description

Sends a request to the iFormBuilder API to get all records in a given form or subform for a specific set of fields (columns). This function can be used to exceed the API limit of 1000 records per request. It will loop through chunks of 1000 records at a time until all records have been retrieved. You can also specify chunk size (< 1000) using the `limit` parameter. Specify how many records to skip before starting to retrieve records using the `offset` parameter.

Usage

```
get_all_records(server_name, profile_id, page_id, fields = "fields",
  limit = 1000, offset = 0, access_token, field_string, since_id)
```

Arguments

`server_name`	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
`profile_id`	The id number of your profile
`page_id`	The id of the form or subform
`fields`	A set of data fields (columns) to return
`limit`	The maximum number of records to return
`offset`	Skips the offset number of records before beginning to return
`access_token`	The access_token required to establish communication with the API
`field_string`	A character string defining the data fields to return, including the id value where data retrieval should start. See the example above for how to define the `parent_form_fields` string when using the `get_selected_page_records()` function.
`since_id`	The id value defining the first record to retrieve.

Details

This function should be used with caution. It should only be used in those rare cases when you know that there are more than 1000 records that need to be retrieved in a single call. Be observant for warnings. It may on occassion throw errors.

Value

A dataframe of records for the specified fields (columns)

Warning

This function should **only** be used to request records from one form or subform at a time. **Do not** assume it will work if records from more than one form are incorporated in the `fields` parameter. If you request multiple fields but the function only returns the id value, and does not throw an error, this is a strong indication that you did not specify the fields correctly. You may have requested a field that does not exist.

Examples

```
# Specify the fields (columns) to be returned
field_list <- c("surveyors", "survey_start_datetime", "survey_method",
                "stream", "survey_end_time")

# Collapse vector of column names into a single string
form_fields <- paste(field_list, collapse = ',')

## Not run:
# Set id to ascending order and pull only records greater than the last_id
since_id <- 5
field_string <- paste0("id:<(>\"", since_id, "\"),", form_fields)

# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id of a single form in the profile given the form name
form_id <- get_page_id(
  server_name = "your_server_name",
  profile_id = 123456,
  page_name = "your_form_p",
  access_token = access_token)

# Get all existing records for a set of columns from a form or subform
parent_form_records <- get_all_records(
  server_name = "your_server_name",
  profile_id = 123456,
  page_id = form_id,
  fields = "fields",
  limit = 1000,
  offset = 0,
  access_token = access_token,
  field_string,
  since_id)

## End(Not run)
```

---

get_core_option_list_elements

Get core elements in an option list

Description

Sends a request to the iFormBuilder API to return the core option list elements. Function will return the id, sort_order, label, key_value, and condition_value.

Usage

```
get_core_option_list_elements(server_name, profile_id, optionlist_id,
  limit = 1000, offset = 0, access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
optionlist_id	The id number for the option list
limit	The maximum number of option list items to return
offset	Skips the offset number of options before beginning to return
access_token	The access_token required to establish communication with the API

Value

A dataframe of the core option list elements

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the core elements in an option list
core_elements <- get_core_option_list_elements(
  server_name = "your_server_name",
  profile_id = 123456,
  optionlist_id = your_option_list_id,
  access_token = access_token)

# Inspect the first five core elements
head(core_elements, 5)

## End(Not run)
```

---

get_iform_access_token

Request an access_token

Description

Sends a request to iFormBuilder for an access_token. This is needed in order to authorize communication with the iFormBuilder API. If you do not have a dedicated server then replace server_name with app.

Usage

```
get_iform_access_token(server_name, client_key_name, client_secret_name)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
client_key_name	The name given to the client_key in your .Renviron file
client_secret_name	The name given to the client_secret in your .Renviron file

Details

For client_key_name use the name you assigned to the client_key in your .Renviron file. For client_secret_name use the name you assigned to the client_secret in your .Renviron file. The client_key_name and client_secret_name, along with their respective values **must** be in your .Renviron file. This function will not work otherwise. Please see the README file at https://github.com/arestrom/iformr for additional information.

Value

An access_token that expires after ten minutes

Examples

```
## Not run:
# Get access_token, assuming you do not have a dedicated server
# Edit client_key and client_secret arguments as needed.
access_token <- get_iform_access_token(
  server_name = "app",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

## End(Not run)

## Not run:
# Get access_token, assuming your dedicated server is "wdfw"
access_token <- get_iform_access_token(
  server_name = "wdfw",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

## End(Not run)
```

---

get_option_lists *Get a listing of the first 100 option lists in a given profile*

Description

Sends a request to the iFormBuilder API to get a listing of the first 100 option lists currently posted in the given profile. The API call limit is currently set at 100 records.

Usage

```
get_option_lists(server_name, profile_id, limit = 100, offset = 0,
  access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
limit	The maximum number of option lists to return
offset	Skips the offset number of options before beginning to return
access_token	The access_token required to establish communication with the API

Value

A listing of all option lists in the given profile

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id and name of all option lists in profile
option_lists <- get_option_lists(
  server_name = "your_server_name",
  profile_id = 123456,
  access_token = access_token)

# Inspect
option_lists

## End(Not run)
```

---

get_option_list_element_ids

Get list of option_ids for a given element

Description

Sends a request to the iFormBuilder API to get a list of all element ids in an option list for a specific field. For example: key_value. Returns a dataframe with ids and attributes of the specified element.

Usage

```
get_option_list_element_ids(server_name, profile_id, optionlist_id,
  element, limit = 1000, offset = 0, access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
optionlist_id	The id number for the option list

element	The specific option list element. For example: "key_value".
limit	The maximum number of option element ids to return
offset	Skips the offset number of options before beginning to return
access_token	The access_token required to establish communication with the API

Value

A dataframe with id and attributes for the selected element.

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the element ids
element_ids <- get_option_list_element_ids(
  server_name = "your_server_name",
  profile_id = 123456,
  optionlist_id = your_option_list_id,
  element = "key_value",
  access_token = access_token)

# Inspect the first five element ids
head(element_ids, 5)

## End(Not run)
```

---

get_option_list_id *Get the id of a single option list given an option list name*

Description

Sends a request to the iFormBuilder API to get the id number for a single option list. You only need to supply the name of the option list.

Usage

```
get_option_list_id(server_name, profile_id, option_list_name,
  limit = 1000, offset = 0, access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
option_list_name	The name of the option list
limit	The maximum number of option lists to return
offset	Skips the offset number of options before beginning to return
access_token	The access_token required to establish communication with the API

Value

A listing of all option lists in the given profile

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id of a single option list given the name
streams_option_list_id <- get_option_list_id(
  server_name = "your_server_name",
  profile_id = 123456,
  option_list_name = "SGS-Streams",
  access_token = access_token)

## End(Not run)
```

---

get_pages_list *Get list of the first 100 pages (i.e., forms, or subforms) in a profile*

Description

Sends a request to the iFormBuilder API to get a listing of forms and subforms in the current profile. Returns a dataframe with form id and form name. Limited to 100 records per API call.

Usage

```
get_pages_list(server_name, profile_id, limit = 1000, offset = 0,
  access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
limit	The maximum number of form ids to return
offset	Skips the offset number of ids before beginning to return
access_token	The access_token required to establish communication with the API

Value

A dataframe of all forms (pages) in the given profile

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id and name of all forms in profile
forms_list <- get_pages_list(
  server_name = "your_server_name",
  profile_id = 123456,
  access_token = access_token)

# Inspect
forms_list

## End(Not run)
```

---

get_page_id *Get id of a single page (i.e., form, or subform) given a form name*

Description

Sends a request to the iFormBuilder API to get the id number of a single form. You only need to supply the name of the option list. Returns an integer id for the given form.

Usage

```
get_page_id(server_name, profile_id, page_name, limit = 1000,
  offset = 0, access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
page_name	The name of the form or subform
limit	The maximum number of form ids to return
offset	Skips the offset number of ids before beginning to return
access_token	The access_token required to establish communication with the API

Value

An integer id for the given form

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id of a single form in the profile given the form name
form_id <- get_page_id(
  server_name = "your_server_name",
  profile_id = 123456,
  page_name = "spawning_ground_p",
  access_token = access_token)

# Inspect the form_id
form_id

## End(Not run)
```

`get_page_record` *Get a single record from a page (i.e., form, or subform)*

Description

Sends a request to the iFormBuilder API to get a single record from a form or subform given a record id.

Usage

```
get_page_record(server_name, profile_id, page_id, record_id, access_token)
```

Arguments

`server_name`	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
`profile_id`	The id number of your profile
`page_id`	The id for the form
`record_id`	The id for the specific record to return
`access_token`	The access_token required to establish communication with the API

Value

Dataframe of a single record from the given form

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")
```

```
# Get the id of a single form in the profile given the form name
form_id <- get_page_id(
  server_name = "your_server_name",
  profile_id = 123456,
  page_name = "your_form_p",
  access_token = access_token)

# Get a list of all record ids in the specified form
record_ids <- get_page_record_list(
  server_name = "your_server_name",
  profile_id = 123456,
  page_id = form_id,
  access_token = access_token)

# Inspect the top five record_ids
head(record_ids, 5)

# Get the first record in the list
single_record_id = record_ids[1]

# Get a single record from a form or subform
single_form_record <- get_page_record(
  server_name = "your_server_name",
  profile_id = 123456,
  page_id = form_id,
  record_id = single_record_id,
  access_token = access_token)

# Inspect the first five columns of the single record dataframe
single_form_record[,1:5]

## End(Not run)
```

`get_page_record_list` *Get list of all record ids in a single page (i.e., form, or subform)*

Description

Sends a request to the iFormBuilder API to get a list of all record ids in a given form or subform.

Usage

```
get_page_record_list(server_name, profile_id, page_id, limit = 1000,
  offset = 0, access_token)
```

Arguments

`server_name`	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
`profile_id`	The id number of your profile
`page_id`	The id number of the form
`limit`	The maximum number of form ids to return
`offset`	Skips the offset number of ids before beginning to return
`access_token`	The access_token required to establish communication with the API

Value

A vector of record ids from the given form

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id of a single form in the profile given the form name
form_id <- get_page_id(
  server_name = "your_server_name",
  profile_id = 123456,
  page_name = "your_form_p",
  access_token = access_token)

# Get a list of all record ids in the specified form
record_ids <- get_page_record_list(
  server_name = "your_server_name",
  profile_id = 123456,
  page_id = form_id,
  access_token = access_token)

# Inspect the top five record_ids
head(record_ids, 5)

## End(Not run)
```

---

get_photos *Get photos from form*

Description

Downloads all of the photos taken with the photo element in a form to a local directory, naming the files based on another field in the form. Since the filenames produced in IFB are jibberish to people, the recommended practice is to include a dynamically calculated element to uniquely identify each photo. For example, in a workflow with a trip form, station subform, and station_photos subform, the station_photos form would include a photoid field that created unique photoids by concaternation of the trip ID, site ID, and index of the photo record.

Support is also provided for writing form data to the EXIF data of the downloaded image. This can be useful as images from iFormBuilder have limited EXIF data. To enable this functionality, provide the full path to exiftool.exe, available from https://www.sno.phy.queensu.ca/~phil/exiftool/ If ExifTool is available, the following form metadata will be added to the image file:

- Date the photo was taken (CREATED_DATE field) to #' EXIF tag *CreateDate*
- The iFormBuilder user who took the photo (CREATED_BY field) to EXIF tag *Artist*
- CREATED_DEVICE_ID field to EXIF tag *CameraSerialNumber*
- CREATED_LOCATION field will be parsed into EXIF tags:

  - *GPSLatitude* and *GPSLongitude*
  - *GPSLatitudeRef* and *GPSLongitudeRef* for N/S and E/W hemispheres respectively.
  - *GPSAltitude*
  - *GPSAltitudeRef* set to 0 if above sea level or 1 if below.
- EXIF tag *Software* will be set to "Zerion iFormBuilder"
- If the comment argument is provided, the text from the comment field in the form will be added to the EXIF tag *ImageDescription*

Usage

```
get_photos(server_name, profile_id, access_token, page_id, photo, photoid,
  output, exif, comment, overwrite = F)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	Integer ID of the form to download photos from.
photo	Character string of the photo element's DCN.
photoid	Character string of the DCN to use as a filename for the photo. Must contain unique values.
output	Path of the output directory for the downloaded photos.
exif	Optional; the path to exiftool.exe. If provided, ExifTool will be used to add metadata to the image files.
comment	Optional; the DCN of a field to append to the EXIF tag "ImageDescription".
overwrite	Optional; should photos already existing in the download directory be overwritten? Defaults to false.

Value

Boolean True if successfull.

Author(s)

Bill DeVoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Download photos from a form


## End(Not run)
```

`get_selected_page_records` *Get multiple records for a set of fields in a page (i.e., form, or subform)*

Description

Sends a request to the iFormBuilder API to get the first 1000 records or less in a given form or subform for a specific set of fields (columns). Specify how many records to retrieve using the `limit` parameter. Specify how many records to skip before starting to retrieve records using the `offset` parameter.

Usage

```
get_selected_page_records(server_name, profile_id, page_id,
  fields = "fields", limit = 100, offset = 0, access_token)
```

Arguments

`server_name`	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
`profile_id`	The id number of your profile
`page_id`	The id of the form or subform
`fields`	A set of data fields (columns) to return
`limit`	The maximum number of records to return
`offset`	Skips the offset number of records before beginning to return
`access_token`	The access_token required to establish communication with the API

Details

This will likely be the primary function used to retrieve records from the iFormBuilder API. When a set of records is downloaded using this function the retrieved data will contain the record id as the first column. By archiving these ids, each request for new records can incorporate the last downloaded id in the `fields` parameter to only pull newly submitted records. See example below.

Value

A dataframe of records for the specified fields (columns)

Warning

This function should **only** be used to request records from one form or subform at a time. **Do not** assume it will work if records from more than one form are incorporated in the `fields` parameter. If you request multiple fields but the function only returns the id value, and does not throw an error, this is a strong indication that you did not specify the fields correctly. You may have requested a field that does not exist.

Examples

```
# Specify the fields (columns) to be returned
field_list <- c(
 "surveyors", "survey_start_datetime", "survey_method",
 "stream", "survey_end_time")

# Collapse vector of column names into a single string
form_fields <- paste(field_list, collapse = ',')

## Not run:
# Set id to ascending order and pull only records greater than the last_id
since_id <- 5
parent_form_fields <- paste0("id:<(>\"", since_id, "\"),", form_fields)

# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the id of a single form in the profile given the form name
form_id <- get_page_id(
  server_name = "your_server_name",
  profile_id = 123456,
  page_name = "your_form_p",
  access_token = access_token)

# Get multiple records for a set of columns from a form or subform
parent_form_records <- get_selected_page_records(
  server_name = "your_server_name",
  profile_id = 123456,
  page_id = form_id,
  fields = parent_form_fields,
  access_token = access_token)

# Inspect the first three rows and first five columns of the dataframe
parent_form_records[1:3,1:5]

## End(Not run)
```

`idate_time` *Convert datetime values output by iFormBuilder*

Description

Date and time values output from iFormBuilder can be difficult to work with. The output will typically look like: `"2014-11-13T15:04:00+00.00"`. The `idate_time()` function converts datetime outputs to standard format `"2014-10-13 08:04:03"` taking timezone into consideration.

Usage

```
idate_time(dts, timezone = "America/Los_Angeles")
```

Arguments

dts	A vector of datetimes as string values
timezone	A character string in standard format specifying timezone. Use OlsonNames() for a list of standard time zone character strings.

See Also

Sys.timezone for timezones

Examples

```
# Create vector that straddles change in daylight savings time
# Daylight savings time occurred: 2017-03-12 02:00:00
dts = c("2017-03-12T01:10:00+00:00", "", NA, "2017-03-12T02:10:00+00:00")

# Convert to standard format
idate_time(dts)
```

---

jencode *Generate encoded request token*

Description

Generate encoded request token

Usage

```
jencode(jheader, jpayload, client_secret)
```

Arguments

jheader	The jwt_header in json format
jpayload	The jwt_payload in json format
client_secret	The client_secret assigned to the application

Value

The signed and encoded request token

jwt_payload *Define the jwt payload*

Description

Define the jwt payload

Usage

```
jwt_payload(client_key, iat = NULL, exp = NULL, token_uri,
  duration = 600L)
```

Arguments

client_key	The client_key assigned to the application
iat	The issue time
exp	The expiration time
token_uri	The uri for requesting the access_token
duration	The length of time between issue time and expiration time

Value

The json web token payload as a list

locations *Stream reach breaks for spawning ground surveys.*

Description

A dataset containing a list of stream reaches typically surveyed for returning salmon in NW Washington State. The dataset is formatted as an iFormBuilder segmented option list. One or more reach breaks are indicated for each stream as river mile location points. Streams are indicated by the unique stream_id value following the equals sign in condition_value.

Usage

```
locations
```

Format

A data frame with 186 rows and 4 variables:

key_value unique id

label point along stream, in river mile

condition_value stream filter, unique stream id after equals sign

sort_order unique integer sequence, starting at zero, incrementing by one

rename_page *Rename page*

Description

Renames a page given a page_id and new name and label. The name provided will be converted to iFormBuilder standards; punctuation and whitespace replaced with _ and all text to lowercase.

Usage

```
rename_page(server_name, profile_id, access_token, page_id, name, label)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	Integer of the page ID to rename.
name	String of renamed page name; coerced to iFormBuilder table name conventions.
label	String of the renamed page label.

Value

Integer of the page ID.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Rename page
rename_page_id <- rename_page(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  page_id = "existing_page_id",
  name = "new_page_name",
  label = "new_page_label"

## End(Not run)
```

`retrieve_element_list` *Retrieve a List of Elements*

Description

Retrieves a list of all the elements contained in a page.

Usage

```
retrieve_element_list(server_name, profile_id, access_token, page_id,
  fields = "all")
```

Arguments

`server_name`	String of the iFormBuilder server name.
`profile_id`	Integer of the iFormBuilder profile ID.
`access_token`	Access token produced by `get_iform_access_token`
`page_id`	Page ID where the new element will be created.
`fields`	*Optional* - Defaults to 'all', which returns all fields for each element. Optionally, a character vector of fields to return can be provided.

Value

Dataframe containing a row for each element and a column for each element field.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get list of elements in form
elements <- retrieve_element_list(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  page_id = "existing_page_id",
  fields = 'all'

## End(Not run)
```

retrieve_page *Retrieve page*

Description

Retrieves page details for a page ID.

Usage

```
retrieve_page(server_name, profile_id, access_token, page_id)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
access_token	Access token produced by get_iform_access_token
page_id	ID of the page to retrieve.

Value

List containing page details.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Retrieve page
page_data <- retrieve_page(
  server_name = "your_server_name",
  profile_id = "your_profile_id",
  access_token = access_token,
  page_id = page_id)

## End(Not run)
```

sync_table *Sync table*

Description

Syncs a dataframe with the contents of an IFB page. If the page does not yet exist, it will be created and populated with the source data. An example use case is syncing an existing database table with a Smart Table Search form in IFB. All columns in the source data with matching columns in the form data are synced. Data is synced in a parent-to-child fashion: - Rows in the source data not in the form data are added (based on the unique identifier from the `uid` argument.) - Rows in the form data that are not present in the source data can be optionally removed using the `delete` argument. - Rows in the form data are updated if any of their fields differ from the matching row in the source data. Updating can be disabled by passing False to the `update` argument.

Usage

```
sync_table(server_name, profile_id, access_token, data, form_name, label,
  uid, update = T, delete = F)
```

Arguments

`server_name`	String of the iFormBuilder server name.
`profile_id`	Integer of the iFormBuilder profile ID.
`access_token`	Access token produced by `get_iform_access_token`
`data`	A dataframe containing the data to be synced with the page.
`form_name`	The name of a page to sync the source data to; if the page does not exist, it will be created.
`label`	*Optional* - String of the label to be used if a new page is created. If a label is not provided and a new page is created, the form_name argument will be used to create a page label.
`uid`	The name of the column in the source and IFB data that uniquely identifies a record.
`update`	*Optional* Defaults to True - If True, records in the form data will be updated if the matching record in the source data is different.
`delete`	*Optional* Defaults to False - If True, records in the form data not present in the source data will be removed.

Value

The page ID of the existing or created form.

Author(s)

Bill DeVoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Add new data to form
sync_table(server_name, profile_id, access_token,
  data = "new_dataframe", form_name = "my_form",
  uid = "unique_id_col")

## End(Not run)
```

`truncate_form` *Truncate form*

Description

Removes all records from a page, leaving the page structure. USE WITH CAUTION!

Usage

```
truncate_form(server_name, profile_id, access_token, page_id)
```

Arguments

`server_name`	String of the iFormBuilder server name.
`profile_id`	Integer of the iFormBuilder profile ID.
`access_token`	Access token produced by `get_iform_access_token`
`page_id`	Integer ID of the form to truncate.

Value

Boolean True if succesful.

Author(s)

Bill DeVoe, <William.DeVoe@maine.gov>

update_options_in_list *Update values in an existing option list*

Description

Sends a request to the iFormBuilder API to update existing values in an option list. Option values for the specified fields will be updated to the new values supplied in the json object 'option_values'. Do not use the fields parameter. It has not been implemented yet.

Usage

```
update_options_in_list(server_name, profile_id, optionlist_id,
  option_values, fields = "fields", limit = 1000, offset = 0,
  access_token = access_token)
```

Arguments

server_name	The server name as encoded in the url: 'https//server_name.iformbuilder.com'
profile_id	The id number of your profile
optionlist_id	The id number for the option list
option_values	A json object containing new option list values
fields	Placeholder for fields to update, not yet implemented
limit	The maximum number of option list items to return
offset	Skips the offset number of options before beginning to update
access_token	The access_token required to establish communication with the API

Value

A vector of option ids for elements that were updated

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

# Get the core elements in the locations option list example.
core_elements <- get_core_option_list_elements(
  server_name = "your_server_name",
  profile_id = 123456,
  optionlist_id = your_locations_option_list_id,
  access_token = access_token)

# Inspect the first five core elements
head(core_elements, 5)

# Edit two of the core_elements we pulled above.
```

```
updated_options = core_elements[3:4,]
updated_options

# Edit. Assume RMs were incorrect.
updated_options$label[1] = "RM 0.20"
updated_options$label[2] = "RM 1.80"

# Convert location list dataframe to json
updated_options_json <- jsonlite::toJSON(updated_options, auto_unbox = TRUE)

# Send request
updated_ids <- update_options_in_list(
  server_name = "wdfw",
  profile_id = 417763,
  optionlist_id = your_locations_option_list_id,
  option_values = updated_options_json,
  access_token = access_token)

# Inspect the updated option list item ids.
updated_ids

## End(Not run)
```

`update_option_lists` *Update iFormBuilder option lists*

Description

For an input dataframe of option lists, adds new option lists to the profile, adds new options to existing option lists, and updates condition_value for existing options.

Usage

```
update_option_lists(access_token, server_name, profile_id, option_lists)
```

Arguments

`access_token`	Access token produced by `get_iform_access_token`
`server_name`	String of the iFormBuilder server name
`profile_id`	Integer of the iFormBuilder profile ID
`option_lists`	A dataframe containing option lists withname, key_value, label, condition_value, and sort_order.

Value

No return value.

update_records *Update records (up to 100)*

Description

Updates up to 100 records in an iFormBuilder page.

Usage

```
update_records(server_name, profile_id, page_id, access_token, record_data)
```

Arguments

server_name	String of the iFormBuilder server name.
profile_id	Integer of the iFormBuilder profile ID.
page_id	Integer ID of the page to perform the update on.
access_token	Access token produced by get_iform_access_token
record_data	A dataframe containing the update data.The dataframe must contain columns for the fields to update, along with an 'id' column containing the record IDs to update.

Value

The updated record ID or IDs.

Author(s)

Bill Devoe, <William.DeVoe@maine.gov>

Examples

```
## Not run:
# Get access_token
access_token <- get_iform_access_token(
  server_name = "your_server_name",
  client_key_name = "your_client_key_name",
  client_secret_name = "your_client_secret_name")

## End(Not run)
```

Index