Package ‘hubspot’

December 20, 2021

Title Work with HubSpot data

Version 0.3.6

Date 2021-12-20

Description Utilising the 'Hubspot CRM' API (see <https://developers.hubspot.com/docs> for more information), retrieve customer relationship data for reporting and analysis.

License MIT+ file LICENSE

Depends R (>= 4.1.2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports httr, dplyr, purrr, magrittr, tidyr, rlang, keyring, methods, tibble, anytime, memoise, ratelimitr, glue

Suggests rmarkdown, knitr, testthat (>= 2.1.0), covr, pkgload

Author Joost Kuckartz [aut, cph, cre],
Steph Locke [aut, cph],
Jessie Geahlen [ctb],
Locke Data [fnd] (https://itsalocke.com),
Maëlle Salmon [aut] (<https://orcid.org/0000-0002-2815-0399>),
Michael W. Kearney [ctb] (<https://orcid.org/0000-0002-0730-4694>)

Maintainer Joost Kuckartz <jkuckartz1984@hotmail.com>

R topics documented:

hubspot-package *hubspot*

Description

Utilizing the 'Hubspot CRM' API to retrieve customer relationship data for reporting and analysis.

Author(s)

Maintainer: Joost Kuckartz <jkuckartz1984@hotmail.com> [copyright holder]

Authors:

- Steph Locke <steph@itsalocke.com> [copyright holder]
- Maëlle Salmon <maelle.salmon@yahoo.se> (ORCID)

Other contributors:

- Jessie Geahlen <jgeahlen@gmail.com> [contributor]
- Locke Data (https://itsalocke.com) [funder]
- Michael W. Kearney <kearneymw@missouri.edu> (ORCID) [contributor]

hs_account_details *Get account details*

Description

Get the account details for the API key or OAuth token.

Usage

```
hs_account_details(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)
```

Arguments

token_path	Path to OAuth 2.0 token. See hubspot_token_create().
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().

Examples

```
account <- hs_account_details()
```

hs_build_property *Property create*

Description

Helper function to create a new object property with the right settings. Used within the creation or editing of schemas of objects.

Usage

```
hs_build_property(
  name,
  label,
  type,
  field_type = NULL,
  group_name = NULL,
  description = NULL,
  option_labels = NULL,
  option_values = NULL,
  display_order = -1,
  has_unique_value = FALSE,
  hidden = FALSE,
  mainpropertylist = NULL
)
```

Arguments

`name` The internal property name, which must be used when referencing the property from the API. Cannot contain spaces and is commonly lowercase letters and underscores only.

`label` A human-readable property label that will be shown in HubSpot. Start with capital letter.

`type` The type of property to create. Must be one of `enumeration`, `date`, `datetime`, `string`, `number` or `bool`.

`field_type` The method how the property appears in HubSpot or on a form. If provided, it must match the provided `type`:

- `type=enumeration`This field can only be either of `select`, `checkbox`, or `radio`.
- `type=date`This field can only be `date`.
- `type=datetime`This field can only be `date`.
- `type=string`This field can only be either of `text`, `textarea`, or `file`.
- `type=number`This field can only be `number`.
- `type=bool`This field can only be `booleancheckbox`.

If this parameter is not provided (`NULL`) or the `type` only supports one `field_type`, it will take the first value mentioned in the list above.

`group_name` The name of the group this property belongs to. The HubSpot API has no functionality to manage groups. Default=`NULL`, which does not assign this property to a group.

`description` A description of the property that will be shown as help text in HubSpot. Default=`NULL`, which leaves the description empty.

`option_labels` Vector of string labels for the available options of an enumeration property (`type=enumeration`). This field is only required for enumerated properties and is ignored for any other types.

`option_values` Vector of string values for the available options of an enumeration property (`type=enumeration`). Must be of same length as `optionlabels`. This field is only required for enumerated properties and is ignored for any other types.

`display_order` The order that this property should be displayed in the HubSpot user interface relative to other properties for this object type. Properties are displayed in order starting with the lowest positive integer value. A value of -1 (default) will cause the property to be displayed after any positive values.

`has_unique_value`

Logical. If `TRUE`, newly created objects must have this property be different from existing objects with this property. If `FALSE`, duplicate values of this property can occur (default).

`hidden` Logical. If `TRUE`, this property is invisible in the HubSpot user interface. If `FALSE`, this property can be seen and used in the interface (default).

`mainpropertylist`

Provide a list of already created properties to append the newly created property to. Default=`NULL`.

Value

A list with the property settings, or a list of list with properties settings when `mainpropertylist` is provided. Parameters that are left to its default are not returned in the list.

Examples

```
newprop=hs_build_property(
  name = "make",
  label = "Make",
  type = "string",
  field_type = "text",
  description = "The manufacturer of a car"
)
```

---

hs_companies_raw *Companies endpoint (raw and tidy)*

Description

Get raw and tidy results from the companies endpoint.

Usage

```
hs_companies_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = hs_company_properties_tidy(hs_company_properties_raw(token_path,
    apikey)),
  property_history = "true",
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_companies_tidy(companies = hs_companies_raw(), view = c("properties"))
```

Arguments

token_path	Path to OAuth 2.0 token. See hubspot_token_create().
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().
properties	Character vector of properties to request
property_history	Whether version history for properties should be returned
max_iter	The API is limited to a certain number of responses by page, use max_iter to limit how many pages of data will get returned
max_properties	Avoid URLs that are too long, limit the number of properties returned, if required.
offsetvalue	The offset value to use if continuing a previous call. Defaults to 0.
companies	A companies return from hs_companies_raw()
view	A view (see Details)

Details

Required scope(s) of the OAuth token: contacts.

Different view values and associated output.

- "properties": A tibble containing all properties of companies

Value

List with company data (hs_companies_raw())

A tibble with associated entities (hs_companies_tidy())

Examples

```
companies <- hs_companies_raw(
  property_history = "false", max_iter = 1,
  max_properties = 10
)
companies_properties <- hs_companies_tidy(
  companies,
  view = "properties"
)
```

---

hs_contacts_raw *Contacts endpoint (raw and tidy)*

Description

Get raw and tidy results from the contacts endpoint.

Usage

```
hs_contacts_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
 properties = hs_contact_properties_tidy(hs_contact_properties_raw(token_path =
    token_path, apikey = apikey)),
  property_history = "true",
  form_submission_mode = "newest",
  list_memberships = "false",
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_contacts_tidy(contacts = hs_contacts_raw(), view = c("properties"))
```

Arguments

token_path — Path to OAuth 2.0 token. See hubspot_token_create().

apikey — API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().

properties — Character vector of properties to request

property_history — Whether version history for properties should be returned

form_submission_mode — Which form submissions should be fetched

list_memberships — Indicate whether current list memberships should be fetched

max_iter — The API is limited to a certain number of responses by page, use max_iter to limit how many pages of data will get returned

max_properties — Avoid URLs that are too long, limit the number of properties returned, if required.

offsetvalue — The offset value to use if continuing a previous call. Defaults to 0.

contacts — A contacts return from hs_contacts_raw()

view — A view (see Details)

Details

Required scope(s) of the OAuth token: contacts.

Different view values and associated output.

- "properties": A tibble containing all properties of contacts

Value

List with contact data (hs_contacts_raw())

A tibble with associated entities (hs_contacts_tidy())

Examples

```
contacts <- hs_contacts_raw(
  property_history = "false", max_iter = 1,
  max_properties = 10
)
contacts_properties <- hs_contacts_tidy(
  contacts,
  view = "properties"
)
```

hs_contact_lists_raw *Contact lists endpoint (raw and tidy)*

Description

Retrieve raw and tidy results from the contact lists endpoint.

Usage

```
hs_contact_lists_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_contact_lists_tidy(
  contact_lists = hs_contact_lists_raw(),
  view = c("names")
)
```

Arguments

token_path	Path to OAuth 2.0 token. See hubspot_token_create().
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().
max_iter	The API is limited to a certain number of responses by page, use max_iter to limit how many pages of data will get returned
offsetvalue	The offset value to use if continuing a previous call. Defaults to 0.
contact_lists	A contact lists return from hs_contact_lists_raw()
view	A view (see Details)

Details

Required scope(s) of the OAuth token: contacts.

Different view values and associated output.

- "names": A tibble containing all IDs and names of contact lists

Value

List of contact lists (hs_contact_lists_raw())

A tibble with associated entities (hs_contact_lists_tidy())

Examples

```
contact_lists <- hs_contact_lists_raw()
hs_contact_lists_tidy(contact_lists)
```

hs_create_company *Object create*

Description

Create one or more objects with the given properties and return a copy of the object(s), including identifier(s).

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the 'contacts' object, where the limit is 10 objects per batch API call.

Usage

```
hs_create_company(
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_contact(
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_deal(
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_line_item(
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_product(
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_ticket(
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_object(
  objecttype,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)
```

Arguments

`properties`	Named list containing the properties to create or update. To create or update multiple objects, each named list of properties needs to be within a list (meaning, a list of named list with properties).
`token_path`	Path to OAuth 2.0 token. See `hubspot_token_create()`.
`apikey`	API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.
`objecttype`	String name of the main object type. Can be any of the default HubSpot objects (`contacts`, `companies`, `deals`, `tickets`, `line_items`, `products`, `quotes`) or a custom created object.

Value

List with created or updated object data (similar as `hs_get_xxx()`). The order of this list will NOT be the same as how you provided it into the function!

Examples

```
## Not run:
oneproduct <- hs_create_object(
  objecttype = "products",
  properties = list(name="New product",
                    description="Awesome product for everyone")
)
twoproducts <- hs_create_object(
  objecttype = "products",
  properties = list(list(name="First product",
                         description="Awesome product for everyone"),
                    list(name="Second product",
                         description="Even better product for everyone")
                   )
)

## End(Not run)
```

hs_create_company_association

Object create association

Description

Associate one or more objects with one or more other objects. The function will associate the first element of vector `identifier_from` to the first element of vector `identifier_to`, second element to second element, and so on. When you want to associate one base object to multiple other objects, you have to provide the `identifier_from` multiple times.

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the 'contacts' object, where the limit is 10 objects per batch API call.

Usage

```
hs_create_company_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_contact_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_deal_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_line_item_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_product_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_ticket_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_create_object_association(
  objecttype_from,
  identifiers_from,
  objecttype_to,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)
```

Arguments

`identifiers_from`
: The base object identifier(s) (either `vid` or `id`).

`objecttype_from`
: String name of the base object type, the object type to associate from. Can be any of the default HubSpot objects (`contacts`, `companies`, `deals`, `tickets`, `line_items`, `products`, `quotes`, `engagements`) or a custom created object.

`identifiers_to`
: The object identifier(s) (either `vid` or `id`) of the object(s) to associate with.

`association_type`
: String name of the association type. Follows the `object_to_object` string format. A full list is not available, but the CRM associations API table provides most association types. Custom objects create new association type names as can be found in the CRM Custom Objects API.

`token_path`
: Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey`
: API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`objecttype_to`
: String name of the object type to associate with.

Value

List with association data or `NULL` in case no object was found (error 404), or `NA` if there was an internal server error (error 500). All server error codes will display a warning.

The returned list of association data won't always be the same length as the provided identifiers. Identifiers that do not have associations are excluded from the returned list.

Examples

```
## Not run:
newcompanytocontactassociation <- hs_create_object_association(
  objecttype_from = "companies",
  identifiers_from = 1,
  objecttype_to = "contact",
  identifiers_to = 25,
  association_type = "company_to_contact"
)
newcompaniestocontactsassociations <- hs_create_object_association(
  objecttype_from = "companies",
  identifiers_from = c(1,2),
  objecttype_to = "contact",
  identifiers_to = c(25,26),
  association_type = "company_to_contact"
)

## End(Not run)
```

hs_create_schema *Schema (custom object) create*

Description

Define a new custom object schema, along with its properties and associations.

Usage

```
hs_create_schema(
  name,
  singular_label,
  plural_label,
  properties,
  display_properties,
  required_properties = NULL,
  searchable_properties = NULL,
  associated_objects,
  apikey = hubspot_key_get()
)

hs_create_custom_object(
  name,
  singular_label,
  plural_label,
  properties,
  display_properties,
  required_properties = NULL,
  searchable_properties = NULL,
  associated_objects,
  apikey = hubspot_key_get()
)
```

Arguments

`name` The schema (custom object) name, using characters only (no numerals). It's common to use lower case only. Cannot be modified after creation.

`singular_label` The human-readable object name (label) to use for display in its singular form. Start with capital letter. Cannot be modified after creation.

`plural_label` The human-readable object name (label) to use for display in its plural form. Start with capital letter. Cannot be modified after creation.

`properties` List of list containing all properties and their related property settings to create. Use `hs_build_property()` to create a set of properties.

`display_properties`
: Vector of property names (must exist within the `properties` list) that are used as display property (like `"firstname"`, `"lastname"`, `"email"` and `"phone"` for a Contact object). The first property in this list will be the primary display property, all remaining properties will be secondary display properties. Display properties are commonly those properties that are mostly used or define the gross of the object.

`required_properties`
: Vector of property names (must exist within the `properties` list) that are required to be filled when creating new custom objects. Leave `NULL` (default) if no required fields are needed (the Contact or Company object doesn't have any required properties for example).

`searchable_properties`
: Vector of property names (must exist within the `properties` list) that can be searched within. Common to be text based fields, like `"firstname"` or `"lastname"` for the Contact object. Leave `NULL` (default) if no searchable fields are needed.

`associated_objects`
: Vector of object names to associate the custom object with.

`apikey`
: API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

Details

When a new custom object is created, the HubSpot API requires you to refer to it through its `objectTypeId` or its fully qualified name. The `objectTypeId` is defined as `{meta-type}-{object_type_id}`, where the meta-type is always 2 (portal-specific). The fully qualified name is defined as `p{portal_id}_{object_name}`.

The functions in this package use an automatic detection system that automatically transform a custom object name (without numerals) to the fully qualified name. When it detects numerals in its object name, it assumes either an `objectTypeId` or fully qualified name is provided and does not transform it to a fully qualified name using the `portal_id`.

To make either the `objectTypeId` or fully qualified name yourself, you can store the returned identifier after creation of a custom object to make the `objectTypeId` or you can use the `portal_id` and the name used in its creation. The `portal_id` can be obtained through `hs_account_details()[["portalId"]]`.

Value

List with created or updated schema data (similar as `hs_get_schema()`).

Examples

```
## Not run:
proplist=hs_build_property(
  name = "make",
  label = "Make",
  type = "string",
  field_type = "text",
  description = "The manufacturer of a car"
)
proplist=hs_build_property(
  name = "model",
  label = "Model",
  type = "string",
  field_type = "text",
  description = "The model of a car",
  mainpropertylist = proplist
)
schema <- hs_create_schema(
  name = "car",
  singular_label = "Car",
  plural_label = "Cars",
  properties = proplist,
  display_properties = c("make","model"),
  associated_obects = c("contact","deal")
)

## End(Not run)
```

hs_create_schema_association *Schema create association*

Description

Associate a schema with another object.

Usage

```
hs_create_schema_association(
  object,
  association_objecttype,
  association_type = NULL,
  apikey = hubspot_key_get()
)
```

Arguments

`object` The object identifier (`objectTypeId`) or unique name of the object.

`association_objecttype` The object identifier (`objectTypeId`) or unique name of the object to associate with. In case the name is provided, it is always the singular version of the HubSpot object (contrary to the `objecttype` property).

`association_type` Unique string name for the new association type. Common association types follow the `"object_to_object"` string format. This parameter allows to create a custom association type name, but it is not recommended to do this. By not providing this parameter, it will automatically create the `"object_to_object"` name.

For example, a new association between a Car object and a Contact object will be named `"car_to_contact"`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

Value

List with association data or `NULL` in case no object was found (error 404), or `NA` if there was an internal server error (error 500). All server error codes will display a warning.

The returned list of association data won't always be the same length as the provided identifiers. Identifiers that do not have associations are excluded from the returned list.

Examples

```
## Not run:
association <- hs_create_schema_association(
  object = "car",
  association_objecttype = "contact"
)

## End(Not run)
```

hs_deals_raw *Deals endpoint (raw and tidy)*

Description

Get raw and tidy results from the deals endpoint.

Usage

```
hs_deals_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = hs_deal_properties_tidy(hs_deal_properties_raw(token_path, apikey)),
  property_history = "true",
  associations = "true",
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_deals_tidy(
  deals = hs_deals_raw(),
  view = c("associations", "properties history", "properties", "stages history")
)
```

Arguments

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`properties` Character vector of properties to request

`property_history` Whether version history for properties should be returned

`associations` Retrieve links to other entities

`max_iter` The API is limited to a certain number of responses by page, use `max_iter` to limit how many pages of data will get returned

`max_properties` Avoid URLs that are too long, limit the number of properties returned, if required.

`offsetvalue` The offset value to use if continuing a previous call. Defaults to 0.

`deals` A deals list object, typically produced by running `hs_deals_raw`

`view` A view (see Details)

Details

Required scope(s) of the OAuth token: contacts.

Different `view` values and associated output.

- "associations": A tibble with associated entities
- "properties history": A tibble of all field changes over time
- "properties": A tibble containing all properties of deals
- "stages history": A tibble containing all stages a deal has gone through

Value

List with deals data (`hs_deals_raw()`)

A tibble (`hs_deals_tidy()`)

Examples

```
deals <- hs_deals_raw(
  property_history = "false",
  max_iter = 1,
  max_properties = 10
)

deals_properties <- hs_deals_tidy(
  deals,
  view = "properties"
  )
```

`hs_deal_pipelines_raw` *Deal pipelines endpoint (raw and tidy)*

Description

Get raw and tidy results from the deal pipelines endpoint.

Usage

```
hs_deal_pipelines_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  include_inactive = "EXCLUDE_DELETED"
)

hs_deal_pipelines_tidy(
  deal_pipelines = hs_deal_pipelines_raw(),
  view = c("properties", "stages")
)
```

Arguments

token_path — Path to OAuth 2.0 token. See hubspot_token_create().

apikey — API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().

include_inactive — Should inactive entities be returned

deal_pipelines — A list of pipelines from hs_deal_pipelines_raw

view — A view (see Details)

Details

Inactive pipelines are defined as deleted pipelines.

Different view values and associated output.

- "properties": tibble of pipeline IDs and property values
- "stages": tibble of pipeline IDs and stages

Value

List with deal pipeline data (hs_deal_pipelines_raw())

A tibble with associated entities (hs_deal_pipelines_tidy())

Examples

```
deal_pipelines <- hs_deal_pipelines_raw()
properties <- hs_deal_pipelines_tidy(
  deal_pipelines,
  view = "properties"
)
stages <- hs_deal_pipelines_tidy(
  deal_pipelines,
  view = "stages"
)
```

hs_delete_company *Object delete*

Description

Delete one or multiple objects.

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the 'contacts' object, where the limit is 10 objects per batch API call.

Usage

```
hs_delete_company(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_contact(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_deal(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_line_item(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_product(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_ticket(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_object(
  objecttype,
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)
```

Arguments

`identifiers` One or more object identifiers (either `vid` or `id`) of the related object(s).

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`objecttype` String name of the main object type. Can be any of the default HubSpot objects (`contacts`, `companies`, `deals`, `tickets`, `line_items`, `products`, `quotes`) or a custom created object.

Value

This function always returns the same, an empty raw object. If an incorrect input is provided, no warning or error will be given.

To see if an object/schema is properly deleted, use hs_get_xxx to obtain the same object/schema or use hs_get_xxx_list to obtain multiple objects. If the returning information does not contain the object/schema, it is deleted correctly.

Examples

```
## Not run:
oneproduct <- hs_delete_object(
  objecttype = "products",
  identifiers = 1
)
twoproducts <- hs_delete_object(
  objecttype = "products",
  identifiers = c(1,2)
)

## End(Not run)
```

hs_delete_company_association

Object delete association (single)

Description

Remove an association between an object and another object. The function will remove the first object in vector identifier_from that's associated to the first object of vector identifier_to, second element to second element, and so on. When you want to remove multiple associations of one base object to multiple other objects, you have to provide the identifier_from multiple times.

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the 'contacts' object, where the limit is 10 objects per batch API call.

Usage

```
hs_delete_company_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_contact_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_deal_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_line_item_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_product_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_ticket_association(
  identifiers_from,
  objecttype_from,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_delete_object_association(
  objecttype_from,
  identifiers_from,
  objecttype_to,
  identifiers_to,
  association_type,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)
```

Arguments

`identifiers_from`

The base object identifier(s) (either `vid` or `id`).

`objecttype_from`
String name of the base object type, the object type to associate from. Can be any of the default HubSpot objects (`contacts`, `companies`, `deals`, `tickets`, `line_items`, `products`, `quotes`, `engagements`) or a custom created object.

`identifiers_to` The object identifier(s) (either `vid` or `id`) of the object(s) to associate with.

`association_type`
String name of the association type. Follows the `object_to_object` string format. A full list is not available, but the CRM associations API table provides most association types. Custom objects create new association type names as can be found in the CRM Custom Objects API.

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`objecttype_to` String name of the object type to associate with.

Value

This function always returns the same, an empty `raw` object. If an incorrect input is provided, no warning or error will be given. Obtain the association using the get association function (function `hs_get_xxx_association`) with the same input to this function to see if it was properly deleted (the returning association list would not have the `identifiers_to` listed anymore).

Examples

```
## Not run:
deletedcompanytocontactassociation <- hs_delete_object_association(
  objecttype_from = "companies",
  identifier = 1,
  objecttype_to = "contact",
  identifiers_to = 25,
  association_type = "company_to_contact"
)
deletedcompaniestocontactsassociations <- hs_delete_object_association(
  objecttype_from = "companies",
  identifier = c(1,2),
  objecttype_to = "contact",
  identifiers_to = c(25,26),
  association_type = "company_to_contact"
)

## End(Not run)
```

`hs_delete_schema` *Schema delete*

Description

Delete an object schema. Deletion is only possible after all instances of the object has been deleted.

Usage

```
hs_delete_schema(object, harddelete = FALSE, apikey = hubspot_key_get())
```

Arguments

object	The object identifier (objectTypeId) or unique name of the object.
harddelete	Logical. Set to TRUE to hard-delete the schema, so that a new custom object can be made with with the same name. Recommended to use, otherwise the custom object might linger around (and an account has a maximum of 10 custom objects). Default=FALSE.
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().

Value

This function always returns the same, an empty raw object. If an incorrect input is provided, no warning or error will be given.

To see if an object/schema is properly deleted, use hs_get_xxx to obtain the same object/schema or use hs_get_xxx_list to obtain multiple objects. If the returning information does not contain the object/schema, it is deleted correctly.

Examples

```
## Not run:
oneschema <- hs_delete_schema(
  object = "car"
)

## End(Not run)
```

hs_delete_schema_association

schema delete association

Description

Remove an association between the schema and another object.

Usage

```
hs_delete_schema_association(
  object,
  association_objecttype,
  apikey = hubspot_key_get()
)
```

Arguments

object	The object identifier (objectTypeId) or unique name of the object.
association_objecttype	The object identifier (objectTypeId) or unique name of the object to associate with. In case the name is provided, it is always the singular version of the HubSpot object (contrary to the objecttype property).
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().

Value

This function always returns the same, an empty raw object. If an incorrect input is provided, no warning or error will be given. Obtain the association using the get association function (function hs_get_xxx_association) with the same input to this function to see if it was properly deleted (the returning association list would not have the identifiers_to listed anymore).

Examples

```
## Not run:
result <- hs_delete_schema_association(
  object = "car",
  association_objecttype = "contact"
)

## End(Not run)
```

---

hs_engagements_raw *Engagements endpoint (raw and tidy)*

Description

Get all of the engagements in a portal. Returns a paginated set of engagements. from the Get all engagements endpoint.

Usage

```
hs_engagements_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_engagements_tidy(engagements = hs_engagements_raw(), view = "id")
```

Arguments

token_path	Path to OAuth 2.0 token. See hubspot_token_create().
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().
max_iter	The API is limited to a certain number of responses by page, use max_iter to limit how many pages of data will get returned
offsetvalue	The offset value to use if continuing a previous call. Defaults to 0.
engagements	An engagements return from hs_engagements_raw()
view	A view (see Details)

Details

Required scope(s) for the OAuth token: contacts.

view = "id" returns a list of all engagements, type, created and modified dates listed by ID.

view = "contacts" also returns associated contacts, one contact ID per row

view = "companies" also returns associated companies, one company ID per row

view = "deals" also returns associated deals, one deal ID per row

Value

A list (`hs_engagements_raw()`)

A tibble with associated entities (`hs_engagements_tidy()`)

Examples

```
engagements <- hs_engagements_raw()
hs_engagements_tidy(engagements)
```

---

hs_forms_from_a_portal_raw

WIP - Get all forms from a portal (raw and tidy)

Description

For a given portal, return all forms that have been created in the portal. from the Get all forms from a portal endpoint.

Usage

```
hs_forms_from_a_portal_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  all_form_types = FALSE
)

hs_forms_from_a_portal_tidy(
  forms_from_a_portal = hs_forms_from_a_portal_raw(),
  view = "forms"
)
```

Arguments

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`all_form_types` By default non-marketing forms are filtered out of this endpoint. To request all forms, use this paramter with the value of TRUE.

`forms_from_a_portal`
A forms_from_a_portal object, typically produced by running `hs_forms_from_a_portal_raw()`

`view` A view (see Details)

Details

Required scope(s) for the OAuth token: forms.

Different `view` values and associated output.

- "forms": tibble of form IDs, names, createdAt dates, upatedAt date, type, published (T/F) and number of fields
- "fields": tibble of all forms as above with form fields (one field per row)
- "options": tibble of all forms and fields as above, with options for multiple option fields (one option per row)

Value

A list (`hs_forms_from_a_portal_raw()`)

A tibble with associated entities (`hs_forms_from_a_portal_tidy()`)

Examples

```
hs_forms_from_a_portal_raw()
forms <- hs_forms_from_a_portal_tidy()
```

---

hs_get_company *Object get by id*

Description

Get one or more object.

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the 'contacts' object, where the limit is 10 objects per batch API call.

Usage

```
hs_get_company(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_properties = 100
)

hs_get_contact(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_properties = 100
)

hs_get_deal(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_properties = 100
)

hs_get_line_item(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_properties = 100
)

hs_get_product(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_properties = 100
)

hs_get_ticket(
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_properties = 100
)

hs_get_object(
  objecttype,
  identifiers,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_properties = 100
)
```

Arguments

`identifiers`	One or more object identifiers (either `vid` or `id`) of the related object(s).
`token_path`	Path to OAuth 2.0 token. See `hubspot_token_create()`.
`apikey`	API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.
`properties`	Character vector of properties to request
`max_properties`	Avoid URLs that are too long, limit the number of properties returned, if required.
`objecttype`	String name of the main object type. Can be any of the default HubSpot objects (`contacts`, `companies`, `deals`, `tickets`, `line_items`, `products`, `quotes`) or a custom created object.

Value

List with object data, NULL in case no object was found (error 404), or NA if there was an internal server error (error 500). All server error codes will display a warning. An empty list can also be returned if the API call was successful but no objects returned (e.g. the requested object does not exist anymore).

Examples

```
## Not run:
oneproduct <- hs_get_object(
  objecttype = "products",
  identifiers = 1
)
twoproducts <- hs_get_object(
  objecttype = "products",
  identifiers = c(1,2)
)

## End(Not run)
```

---

hs_get_company_association

Object get associations

Description

Get a list of associations one or multiple objects have.

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the 'contacts' object, where the limit is 10 objects per batch API call.

Usage

```
hs_get_company_association(
  identifiers_from,
  objecttype_to,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_get_contact_association(
  identifiers_from,
  objecttype_to,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_get_deal_association(
  identifiers_from,
  objecttype_to,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_get_line_item_association(
  identifiers_from,
  objecttype_to,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_get_product_association(
  identifiers_from,
  objecttype_to,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_get_ticket_association(
  identifiers_from,
  objecttype_to,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)

hs_get_object_association(
  objecttype_from,
  identifiers_from,
  objecttype_to,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  max_iter = 10,
  offsetvalue = 0
)
```

Arguments

`identifiers_from`
The base object identifier(s) (either `vid` or `id`).

`objecttype_to` String name of the object type to associate with.

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

max_iter The API is limited to a certain number of responses by page, use max_iter to limit how many pages of data will get returned

offsetvalue The offset value to use if continuing a previous call. Defaults to 0.

objecttype_from String name of the base object type, the object type to associate from. Can be any of the default HubSpot objects (contacts, companies, deals, tickets, line_items, products, quotes, engagements) or a custom created object.

max_properties Avoid URLs that are too long, limit the number of properties returned, if required.

Value

List with association data or NULL in case no object was found (error 404), or NA if there was an internal server error (error 500). All server error codes will display a warning.

The returned list of association data won't always be the same length as the provided identifiers. Identifiers that do not have associations are excluded from the returned list.

Examples

```
## Not run:
onecompanytocontactassociations <- hs_get_object_association(
  objecttype_from = "companies",
  identifiers_from = 1,
  objecttype_to = "contact"
)
twocompaniestocontactsassociations <- hs_get_object_association(
  objecttype_from = "companies",
  identifiers_from = c(1,2),
  objecttype_to = "contact"
)

## End(Not run)
```

hs_get_company_list *Get list of objects*

Description

Get a list of objects.

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the 'contacts' object, where the limit is 10 objects per batch API call.

Usage

```
hs_get_company_list(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
```

```
)

hs_get_contact_list(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_get_deal_list(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_get_line_item_list(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_get_product_list(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_get_ticket_list(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_get_object_list(
  objecttype,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = NULL,
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)
```

Arguments

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`properties` Character vector of properties to request

`max_iter` The API is limited to a certain number of responses by page, use `max_iter` to limit how many pages of data will get returned

`max_properties` Avoid URLs that are too long, limit the number of properties returned, if required.

`offsetvalue` The offset value to use if continuing a previous call. Defaults to 0.

`objecttype` String name of the main object type. Can be any of the default HubSpot objects (`contacts`, `companies`, `deals`, `tickets`, `line_items`, `products`, `quotes`) or a custom created object.

Value

List with object data, `NULL` in case no object was found (error 404), or `NA` if there was an internal server error (error 500). All server error codes will display a warning. An empty list can also be returned if the API call was successful but no objects returned (e.g. the requested object does not exist anymore).

Examples

```
products <- hs_get_object_list(
  objecttype = "products",
  max_iter = 1,
  max_properties = 10
)
```

hs_get_schema_list *Schema get one or a list*

Description

Get a list of custom schemas, or get a single one by name or identifier.

Usage

```
hs_get_schema_list(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_get_schema(
  object,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)
```

Arguments

token_path	Path to OAuth 2.0 token. See hubspot_token_create().
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().
object	The object identifier (objectTypeId) or unique name of the object.

Value

List with schema data, NULL in case no object was found (error 404), or NA if there was an internal server error (error 500). All server error codes will display a warning.

Examples

```
schemas <- hs_get_schema_list()
oneschema <- hs_get_schema(
  object = "contact"
)
```

hs_lineitems_raw *Line items endpoint (raw and tidy)*

Description

Get raw and tidy results from the line items endpoint.

Usage

```
hs_lineitems_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = hs_object_properties_tidy("line_items",
   hs_object_properties_raw("line_items", token_path = token_path, apikey = apikey)),
  property_history = "true",
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_lineitems_tidy(lineitems = hs_lineitems_raw(), view = c("properties"))
```

Arguments

token_path	Path to OAuth 2.0 token. See hubspot_token_create().
apikey	API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().
properties	Character vector of properties to request
property_history	Whether version history for properties should be returned

`max_iter` The API is limited to a certain number of responses by page, use `max_iter` to limit how many pages of data will get returned

`max_properties` Avoid URLs that are too long, limit the number of properties returned, if required.

`offsetvalue` The offset value to use if continuing a previous call. Defaults to 0.

`lineitems` A line items return from `hs_lineitems_raw()`

`view` A view (see Details)

Details

Required scope(s) of the OAuth token: e-commerce

Different `view` values and associated output.

- "properties": A tibble containing all properties of line items

Value

List with line items data (`hs_lineitems_raw()`)

A tibble with associated entities (`hs_lineitems_tidy()`)

Examples

```
lineitems <- hs_lineitems_raw(
  property_history = "false", max_iter = 1,
  max_properties = 10
)
lineitems_properties <- hs_lineitems_tidy(
  lineitems,
  view = "properties"
)
```

`hs_object_properties_raw`

Object properties endpoint (raw and tidy)

Description

Retrieve raw and tidy results from the object properties endpoint.

Usage

```
hs_object_properties_raw(
  objecttype,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_object_properties_tidy(objecttype, object_properties = NULL, view = "names")

hs_company_properties_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_company_properties_tidy(
  company_properties = hs_company_properties_raw(),
  view = "names"
)

hs_contact_properties_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_contact_properties_tidy(
  contact_properties = hs_contact_properties_raw(),
  view = "names"
)

hs_deal_properties_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_deal_properties_tidy(
  deal_properties = hs_deal_properties_raw(),
  view = "names"
)

hs_product_properties_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_product_properties_tidy(
  product_properties = hs_product_properties_raw(),
  view = "names"
)

hs_ticket_properties_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_ticket_properties_tidy(
  ticket_properties = hs_ticket_properties_raw(),
  view = "names"
)
```

Arguments

`objecttype` String name of the main object type. Can be any of the default HubSpot objects (`contacts`, `companies`, `deals`, `tickets`, `line_items`, `products`, `quotes`) or a custom created object.

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`object_properties`
Object properties as returned by `hs_object_properties_raw()`

`view` A view (see Details)

Details

Required scope(s) of the OAuth token: contacts.

Different `view` values and associated output.

- "names": A character vector of the names of available object properties.

Value

List of object properties (`hs_object_properties_raw()`)

Something tidy (`hs_object_properties_tidy()`)

Examples

```
properties <- hs_object_properties_raw("tickets")
property_names <- hs_object_properties_tidy("tickets", properties, view = "names")
```

`hs_owners_raw` *Owners endpoint (raw and tidy)*

Description

Retrieve raw and tidy results from the owners endpoint.

Usage

```
hs_owners_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  include_inactive = TRUE,
  email = NULL
)

hs_owners_tidy(owners = hs_owners_raw(), view = c("owners"))
```

Arguments

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`include_inactive`
Should inactive entities be returned

`email` Search for owners matching the specified email address (exact match).

`owners` Raw data from `hs_owners_raw()`

`view` A view (see Details)

Details

Inactive owners are defined as owners without any active remotes.

Required scope(s) of the OAuth token: contacts.

Different `view` values and associated output.

- "owners": A tibble with owners and related information.

Value

List of owners (`hs_owners_raw()`)

A tibble of owners (`hs_owners_tidy()`)

Examples

```
## Not run:
owners <- hs_owners_raw()
active_owners <- hs_owners_raw(include_inactive = FALSE)
hs_owners_raw(email = "demo@hubspot.com")

## End(Not run)
```

`hs_products_raw` *Products endpoint (raw and tidy)*

Description

Get raw and tidy results from the products endpoint.

Usage

```
hs_products_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = hs_object_properties_tidy("products",
   hs_object_properties_raw("products", token_path = token_path, apikey = apikey)),
  property_history = "true",
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_products_tidy(products = hs_products_raw(), view = c("properties"))
```

Arguments

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`properties` Character vector of properties to request

`property_history`
Whether version history for properties should be returned

`max_iter` The API is limited to a certain number of responses by page, use `max_iter` to limit how many pages of data will get returned

`max_properties` Avoid URLs that are too long, limit the number of properties returned, if required.

`offsetvalue` The offset value to use if continuing a previous call. Defaults to 0.

`products` A products return from `hs_products_raw()`

`view` A view (see Details)

Details

Required scope(s) of the OAuth token: e-commerce

Different `view` values and associated output.

- "properties": A tibble containing all properties of products

Value

List with products data (`hs_products_raw()`)

A tibble with associated entities (`hs_products_tidy()`)

Examples

```
products <- hs_products_raw(
  property_history = "false", max_iter = 1,
  max_properties = 10
)
products_properties <- hs_products_tidy(
  products,
  view = "properties"
)
```

`hs_tickets_raw` *Tickets endpoint (raw and tidy)*

Description

Get raw and tidy results from the tickets endpoint.

Usage

```
hs_tickets_raw(
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get(),
  properties = hs_object_properties_tidy("tickets", hs_object_properties_raw("tickets",
    token_path = token_path, apikey = apikey)),
  property_history = "true",
  max_iter = 10,
  max_properties = 100,
  offsetvalue = 0
)

hs_tickets_tidy(tickets = hs_tickets_raw(), view = c("properties"))
```

Arguments

`token_path` Path to OAuth 2.0 token. See `hubspot_token_create()`.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

`properties` Character vector of properties to request

`property_history` Whether version history for properties should be returned

`max_iter` The API is limited to a certain number of responses by page, use `max_iter` to limit how many pages of data will get returned

`max_properties` Avoid URLs that are too long, limit the number of properties returned, if required.

`offsetvalue` The offset value to use if continuing a previous call. Defaults to 0.

`tickets` A tickets return from `hs_tickets_raw()`

`view` A view (see Details)

Details

Required scope(s) of the OAuth token: tickets.

Different `view` values and associated output.

- "properties": A tibble containing all properties of tickets

Value

List with tickets data (`hs_tickets_raw()`)

A tibble with associated entities (`hs_tickets_tidy()`)

Examples

```
tickets <- hs_tickets_raw(
  property_history = "false", max_iter = 1,
  max_properties = 10
)
tickets_properties <- hs_tickets_tidy(
  tickets,
  view = "properties"
)
```

`hs_update_company` *Object update*

Description

Update one or more objects.

The order of `identifiers` is used to match to the same order of `properties`, meaning they have to be of equal length.

The functions take into account batch limits for the API. This limit is 100 objects per batch API call, except for the ’contacts’ object, where the limit is 10 objects per batch API call.

Usage

```
hs_update_company(
  identifiers,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_update_contact(
  identifiers,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_update_deal(
  identifiers,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_update_line_item(
  identifiers,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_update_product(
  identifiers,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_update_ticket(
  identifiers,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)

hs_update_object(
  objecttype,
  identifiers,
  properties,
  token_path = hubspot_token_get(),
  apikey = hubspot_key_get()
)
```

Arguments

identifiers — One or more object identifiers (either vid or id) of the related object(s).

properties — Named list containing the properties to create or update. To create or update multiple objects, each named list of properties needs to be within a list (meaning, a list of named list with properties).

token_path — Path to OAuth 2.0 token. See hubspot_token_create().

apikey — API key to work with Hubspot (only used if token_path is NULL). See hubspot_key_set().

objecttype — String name of the main object type. Can be any of the default HubSpot objects (contacts, companies, deals, tickets, line_items, products, quotes) or a custom created object.

Value

List with created or updated object data (similar as hs_get_xxx()). The order of this list will NOT be the same as how you provided it into the function!

Examples

```
## Not run:
oneproduct <- hs_update_object(
  objecttype = "products",
  identifiers = 1,
  properties = list(name="New product",
                    description="Awesome product for everyone")
)
twoproducts <- hs_update_object(
  objecttype = "products",
  identifiers = c(1,2),
  properties = list(list(name="First product",
                         description="Awesome product for everyone"),
                    list(name="Second product",
                         description="Even better product for everyone")
                   )
)

## End(Not run)
```

hs_update_schema *Schema update*

Description

Update the details for an existing object schema. Can only update the display, required, and searchable properties. To add new properties to an object schema, use hs_create_property(). To add new associations to an object schema, use hs_create_schema_association().

Usage

```
hs_update_schema(
  object,
  display_properties = NULL,
  required_properties = NULL,
  searchable_properties = NULL,
  apikey = hubspot_key_get()
)
```

Arguments

`object` The object identifier (`objectTypeId`) or unique name of the object.

`display_properties`
Vector of property names (must exist within the `properties` list) that are used as display property (like `"firstname"`, `"lastname"`, `"email"` and `"phone"` for a Contact object). The first property in this list will be the primary display property, all remaining properties will be secondary display properties. Display properties are commonly those properties that are mostly used or define the gross of the object.

`required_properties`
Vector of property names (must exist within the `properties` list) that are required to be filled when creating new custom objects. Leave `NULL` (default) if no required fields are needed (the Contact or Company object doesn't have any required properties for example).

`searchable_properties`
Vector of property names (must exist within the `properties` list) that can be searched within. Common to be text based fields, like `"firstname"` or `"lastname"` for the Contact object. Leave `NULL` (default) if no searchable fields are needed.

`apikey` API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

Value

List with created or updated schema data (similar as `hs_get_schema()`).

Examples

```
## Not run:
schema <- hs_update_schema(
  object = "car",
  display_properties = c("make","model","year")
)

## End(Not run)
```

`hubspot_api_usage` *Check daily API usage*

Description

Check daily API usage via the dedicated API endpoint.

Usage

```
hubspot_api_usage(apikey = hubspot_key_get())
```

Arguments

`apikey`	API key to work with Hubspot (only used if `token_path` is `NULL`). See `hubspot_key_set()`.

Value

A data.frame with API usage info

Examples

```
## Not run:
hubspot_api_usage()

## End(Not run)
```

`hubspot_key_get` *Set/retrieve the Hubspot secret API key*

Description

Get/set the Hubspot secret API key using the `keyring` package. For a more secure authorization method, without a daily limit on calls, see `hubspot_token_create()`

Usage

```
hubspot_key_get()

hubspot_key_set(key = NULL)
```

Arguments

`key`	API key. If not provided via this argument, the key will have to be provided interactively.

Details

To know how to create your Hubspot API key, refer to the corresponding section of Hubspot API docs. Hubspot recommends you to re-generate your API key every six months. Only accounts with Super Admin Access can create an API key (at the moment of writing).

Save the API key on your machine using the `hubspot_key_set()` function, that relies on the `keyring` package that stores the key in the system default credential store (see `keyring` docs). If you don't do that, you'll have to provide the API key as argument to `hubspot` functions calling Hubspot APIs.

```
# Either input the key as parameter
hubspot_key_set("yourapikey")

# Or just call the function,
# the key will be asked interactively
hubspot_key_set()
```

If no API key is set nor provided as an argument, the “demo” key is used and a message is printed to the screen. As noted earlier, this won’t give you access to your own Hubspot data.

You can check the daily usage related to the API key via `hubspot_api_usage()`. Calls in the package are rate limited (100 requests every 10 seconds) but there’s also a daily limit on calls.

```
hubspot_api_usage()
```

For a more secure authorization method, without a daily limit on calls, see `hubspot_token_create()`.

Authorization for Hubspot APIs:

The Hubspot API accepts authorization via

- an API key,
- OAuth 2.0.

OAuth 2.0 is the recommended method. However, this package supports both.

Note that if you do nothing the package will use the “demo” API token but this won’t give you access to your own Hubspot data. So you’ll need to spend a little time on setup:

- For rapid prototyping key, use a Hubspot API key, see `hubspot_key_set()`.
- For more secure use, without a daily limit on API calls, see `hubspot_token_create()` to create a Hubspot authorization token (OAuth 2.0).

If you have both saved an API key via `hubspot_key_set()` and a token via `hubspot_token_create()`, priority will be given to using the OAuth 2.0 token. If you don’t want that, explicitely pass `NULL` as value for the `token_path` argument of all functions.

Value

The key (a string)

See Also

Other auth: `default_ld_hubspot_app()`

Examples

```
hubspot_key_get()
## Not run:
hubspot_key_set("yourapikey")

## End(Not run)
```

Index