Package: qbr (via r-universe)

October 19, 2026

Title Access the 'Quickbase' JSON API

Version 1.4.0.9000

Description Programmatically access the 'Quickbase' JSON API <https://developer.quickbase.com>. You supply parameters for an API call, 'qbr' delivers an http request to the API endpoint and returns its response. Outputs follow 'tidyverse' philosophy.

URL https://github.com/BHII-KSC/qbr

BugReports https://github.com/BHII-KSC/qbr/issues

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports dplyr (>= 1.0.9), tidyr (>= 1.2.0), httr (>= 1.4.3), jsonlite (>= 1.8.0), purrr (>= 0.3.4), stringr (>= 1.4.0), tibble (>= 3.1.7), httr2 (>= 0.2.3), tidyselect (>= 1.2.0)

Suggests testthat (>= 3.0.0), keyring

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libicu-dev libssl-dev

Repository https://bhii-ksc.r-universe.dev

Date/Publication 2026-08-20 13:36:41 UTC

RemoteUrl https://github.com/bhii-ksc/qbr

RemoteRef HEAD

RemoteSha 17bfa3b2c953596817ad947c9c11b2ca053b7773

Contents

clone_token . . . 2
copy_app . . . 3
create_app . . . 4
deactivate_token . . . 6
delete_app . . . 7
delete_fields . . . 8
delete_records . . . 8
delete_token . . . 9
get_app . . . 10
get_app_events . . . 11
get_fields . . . 12
get_report . . . 13
get_reports . . . 14
get_tables . . . 14
get_users . . . 15
qb_run . . . 16
query_records . . . 18
run_report . . . 19
summarize_app . . . 21
update_records . . . 22

Index **24**

`clone_token` *Clone a user token*

Description

Make a copy of the supplied token and returns its value.

Usage

```
clone_token(
  subdomain,
  auth,
  agent = NULL,
  clone_name = NULL,
  clone_desc = NULL
)
```

Arguments

`subdomain`	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
`auth`	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).

agent	Optional. Character vector with one element. Describes user/agent making API call.
clone_name	Optional. Character vector with one element. Name the token clone.
clone_desc	Optional. Character vector with one element. Provide a description for the token clone.

Value

A character vector with one element containing the token clone.

References

Quickbase API documentation

Examples

```
## Not run:
   x <- clone_token(subdomain = "abc",
                    auth = keyring::key_get("qb_example"),
                    clone_name = "My new token",
                    clone_desc = "This clone was created using R")

## End(Not run)
```

---

copy_app *Copy an app*

Description

Copy an app. Provides options to copy data and users.

Usage

```
copy_app(
  subdomain,
  auth,
  app_id,
  app_name,
  app_desc = NULL,
  agent = NULL,
  users_and_roles = FALSE,
  keep_data = FALSE,
  exclude_files = TRUE,
  assign_user_token = TRUE
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
app_id	Character vector with one element. Unique identifier of an app. Found in the URL of the app's homepage after /db/
app_name	Character vector with one element. Name the app's copy.
app_desc	Optional. Character vector with one element. Describe the app's copy.
agent	Optional. Character vector with one element. Describes user/agent making API call.
users_and_roles	Logical. If true, users will be copied along with their assigned roles. If false, users and roles will be copied but roles will not be assigned.
keep_data	Logical. Whether to copy the app's data along with the schema.
exclude_files	Logical. If keep_data is true, whether to copy the file attachments as well. If keep_data is false, this parameter is ignored.
assign_user_token	Logical. Whether to add the user token used to make this request to the new app.

Value

A list.

Examples

```
## Not run:
   copy_app(subdomain = "abc",
            auth = keyring::key_get("qb_example"),
            app_id = "bn9d8f78g",
            app_name = "Copy of my app",
            keep_data = TRUE)

## End(Not run)
```

---

create_app *Create an app*

Description

Create a new Quickbase app

Usage

```
create_app(
  subdomain,
  auth,
  name,
  agent = NULL,
  assign_token = FALSE,
  description = NULL,
  security_properties = NULL,
  variables = NULL
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
name	Character. Name of the app.
agent	Optional. Character vector with one element. Describes user/agent making API call.
assign_token	Logical, defaults to false. If true, the created app is assigned to the user token used in the auth argument.
description	Vector of length 1. Describes the app created.
security_properties	Named list of security settings for the app. Configurable security properties currently include the following logicals: allowClone, allowExport, enableAppTokens, hideFromPubic, mustBeRealmApproved, and useIPFilter
variables	Named character vector of length 10 or less. The name and value of each element will be used to name a variable and assign it a value in the app created.

Value

A nested list which includes the resulting app ID.

Examples

```
## Not run:
create_app(
  subdomain = "abc",
  auth = keyring::key_get("qb_example"),
  name = "My new app",
  variables = c(
    "Theme Color" = "#74489D",
    "App start" = "2026-09-01"
  ),
  security_properties = list(
```

```
    enableAppTokens = TRUE,
    hideFromPublic = TRUE,
  )
)

## End(Not run)
```

`deactivate_token` *Deactivate a user token*

Description

Make an active user token inactive.

Usage

```
deactivate_token(subdomain, auth, agent = NULL)
```

Arguments

`subdomain`	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
`auth`	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
`agent`	Optional. Character vector with one element. Describes user/agent making API call.

Value

A message confirming deactivation was successful.

Examples

```
## Not run:
   x <- deactivate_token(subdomain = "abc",
                    auth = keyring::key_get("qb_example"))

## End(Not run)
```

delete_app *Delete an app*

Description

Delete an entire app, including all of the tables and data.

Usage

```
delete_app(subdomain, auth, app_id, app_name, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
app_id	Character vector with one element. Unique identifier of an app. Found in the URL of the app's homepage after /db/
app_name	Character vector with one element. The name of the app to be delete. Confirms you want to delete the app.
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

A list.

Examples

```
## Not run:
   delete_app(subdomain = "abc",
              auth = keyring::key_get("qb_example"),
              app_id = "bsf5hphe5",
              app_name = "R Testing copy")

## End(Not run)
```

delete_fields *Delete field(s) in a table*

Description

Delete a list of one or more fields in a table.

Usage

```
delete_fields(subdomain, auth, table_id, field_ids, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
table_id	Character vector with one element. Found in the URL of a Quickbase table.
field_ids	Character or numeric vector. Field identifier for fields to delete.
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

A tibble with the status of the fields passed to field_ids.

Examples

```
## Not run:
   delete_fields(subdomain = "abc",
               auth = keyring::key_get("qb_example"),
               table_id = "bsf5hphe5",
               field_ids = c(40:43, 45))

## End(Not run)
```

delete_records *Delete records*

Description

Delete one or more records from a table.

Usage

```
delete_records(subdomain, auth, from, where, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
from	Character vector. Identifier of the target table.
where	Character vector. Condition(s) which target one or more records for deletion. Use the Quickbase query language to construct your condition(s).
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

Named integer vector

Examples

```
## Not run:
  delete_records(subdomain = "abc",
              auth = keyring::key_get("qb_example"),
              from = "bn9d8iesz",
              where = "{6.EX.'105'}")

## End(Not run)
```

---

delete_token *Delete a user token*

Description

Permanently delete an active user token.

Usage

```
delete_token(subdomain, auth, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

A message confirming deactivation was successful.

Examples

```
## Not run:
   x <- delete_token(subdomain = "abc",
                     auth = keyring::key_get("qb_example"))

## End(Not run)
```

`get_app` *Get an app*

Description

Get metadata for an app.

Usage

```
get_app(
  subdomain,
  auth,
  app_id,
  agent = NULL,
  include_sec = T,
  include_vars = T
)
```

Arguments

`subdomain`	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
`auth`	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
`app_id`	Character vector with one element. Unique identifier of an app. Found in the URL of the app's homepage after /db/
`agent`	Optional. Character vector with one element. Describes user/agent making API call.
`include_sec`	Logical. Includes security properties if true.
`include_vars`	Logical. Includes app variables if true.

Value

A tibble.

Examples

```
## Not run:
   get_app(subdomain = "abc",
              auth = keyring::key_get("qb_example"),
              app_id = "bsf5hphe5")

## End(Not run)
```

`get_app_events` *Get app events*

Description

Get a tibble of events that can be triggered based on data or user actions in this application, includes: Email notification, Reminders, Subscriptions, QB Actions, Webhooks, record change triggered Automations (does not include scheduled).

Usage

```
get_app_events(subdomain, auth, app_id, agent = NULL)
```

Arguments

`subdomain`	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
`auth`	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
`app_id`	Character vector with one element. Unique identifier of an app. Found in the URL of the app's homepage after /db/
`agent`	Optional. Character vector with one element. Describes user/agent making API call.

Value

A tibble.

Examples

```
## Not run:
   get_app_events(subdomain = "abc",
                  auth = keyring::key_get("qb_example"),
                  app_id = "bn9d8f78g")

## End(Not run)
```

get_fields *Get all fields in a table*

Description

Get metadata for all fields in a table.

Usage

```
get_fields(
  subdomain,
  auth,
  table_id,
  agent = NULL,
  include_props = T,
  include_perms = F
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
table_id	Character vector with one element. Found in the URL of a Quickbase table.
agent	Optional. Character vector with one element. Describes user/agent making API call.
include_props	Logical. Includes field properties if true.
include_perms	Logical. Includes custom field permissions if true. Only returns data if custom permissions exist for at least 1 field in the table.

Value

A tibble.

Examples

```
## Not run:
   get_fields(subdomain = "abc",
              auth = keyring::key_get("qb_example"),
              table_id = "bsf5hphe5")

## End(Not run)
```

get_report *Get a report*

Description

Get metadata about a report.

Usage

```
get_report(subdomain, auth, table_id, report_id, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
table_id	Character vector with one element. Found in the URL of a Quickbase table.
report_id	Character vector with one element. Found in the 'Reports & Charts' page in Quickbase and in the report URL.
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

A named list.

Examples

```
## Not run:
   get_report(subdomain = "abc",
              auth = keyring::key_get("qb_example"),
              table_id = "bn9d8iesz",
              report_id = "7")

## End(Not run)
```

get_reports *Get all reports for a table*

Description

get_reports retrieves metadata for each report in a table.

Usage

```
get_reports(subdomain, auth, table_id, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
table_id	Character vector with one element. Found in the URL of a Quickbase table.
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

A tibble.

Examples

```
## Not run:
   get_reports(subdomain = "abc",
               auth = keyring::key_get("qb_example"),
               table_id = "bn9d8iesz")

## End(Not run)
```

get_tables *Get all tables*

Description

Get metadata for all tables in an app.

Usage

```
get_tables(subdomain, auth, app_id, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
app_id	Character vector with one element. Unique identifier of an app. Found in the URL of the app's homepage after /db/
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

A tibble.

Examples

```
## Not run:
   get_tables(subdomain = "abc",
              auth = keyring::key_get("qb_example"),
              app_id = "bsf5hphe5")

## End(Not run)
```

---

get_users *Get users*

Description

Get all users in an account. Provides options to limit values to a set of users and/or apps.

Usage

```
get_users(
  subdomain,
  auth,
  agent = NULL,
  account_id = NULL,
  user_emails = NULL,
  app_ids = NULL
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
agent	Optional. Character vector with one element. Describes user/agent making API call.
account_id	Optional. Positive integer. The account ID being used to get users. If no value is specified, the first account associated with the requesting user token is chosen.
user_emails	Optional. List of characters. Limit returned users to those specified in this list.
app_ids	Optional. List of characters. Limit returned users to those assigned to these app ID's. The provided app ID's should belong to the same account.

Value

A tibble.

Examples

```
## Not run:
   get_users(subdomain = "abc",
             auth = keyring::key_get("qb_example"))

## End(Not run)
```

---

qb_run *Run a Quickbase report*

Description

Run a report and get its data.

Usage

```
qb_run(
  subdomain,
  token,
  table_id,
  report_id,
  agent = NULL,
  skip = 0,
  top = 0,
  type_suffix = FALSE,
  paginate = TRUE
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
token	Character vector with one element. Created in 'My Preferences' under 'Manage user tokens' link.
table_id	Character vector with one element. Found in the URL of a Quickbase table between /db/ and ?
report_id	Character vector with one element. Found in the 'Reports & Charts' page in Quickbase and in the report URL.
agent	Optional. Character vector with one element. Describes user/agent making API call.
skip	Optional. Integer. The number of rows to skip from the top of a record set.
top	Optional. Integer. The limit on the number of records to pull starting at the top of a record set.
type_suffix	Optional. Logical. Set TRUE to append each field label with its Quickbase data type.
paginate	Optional. Logical. Set TRUE to recursively call the API until all report pages are collected

Value

A tibble.

References

Quickbase API documentation

Examples

```
## Not run:

    # Get all data in a report
    my_tibble <- qb_run(subdomain = "abc",
                        token = keyring::key_get("qb_example"),
                        table_id = "bn9d8iesz",
                        report_id = "1")

    # Get rows 3 to 6 from a report
    my_tibble <- qb_run(subdomain = "abc.quickbase.com",
                        token = keyring::key_get("qb_example"),
                        table_id = "bn9d8iesz",
                        report_id = "1",
                        skip = 2,
                        top = 3)

## End(Not run)
```

query_records *Query for data*

Description

Get tabular data from a Quickbase table using a query.

Usage

```
query_records(
  subdomain,
  auth,
  from,
  select = as.numeric(),
  where = NULL,
  group_by = NULL,
  sort_by = NULL,
  agent = NULL,
  skip = 0,
  top = 0,
  local_time = FALSE,
  type_suffix = FALSE
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
from	Character vector with one element. Table identifier.
select	Optional. Numeric vector containing field identifiers for columns to return. If omitted, the default columns for the table will be returned.
where	Optional. Character vector with one element. Use the Quickbase query language to set criteria for records to returned.
group_by	Optional. Nested named list with 'fieldId' and 'grouping' pairs for each field to group by.
sort_by	Optional. Nested named list with 'fieldId' and sort 'order' pairs for each field to sort by. See Quickbase JSON API documentation for details on sort order configuration.
agent	Optional. Character vector with one element. Describes user/agent making API call.
skip	Optional. Integer. The number of rows to skip from the top of a record set.
top	Optional. Integer. The limit on the number of records to pull starting at the top of a record set.

local_time	Logical. When true, date time fields are returned using app's local time. When false, date time fields are returned using UTC time.
type_suffix	Optional. Logical. Set TRUE to append each field label with its Quickbase data type.

Value

A tibble.

Examples

```
## Not run:

   # Get all data matching query specification
   my_tibble <- query_records(subdomain = "abc",
       auth = keyring::key_get("qb_example"),
       from = "bn9d8iesz",
       select = c(3, 6:9),
       where = "{8.EX.'6-month'}")

   # Query data, group, then sort it
   my_tibble <- query_records(subdomain = "bhi",
       auth = keyring::key_get("qb_example"),
       from = "bn9d8iesz",
       select = c(3, 8:12),
       sort_by = list(list(fieldId = 12, order = "ASC"),
                      list(fieldId = 3, order = "DESC")),
       group_by = list(list(fieldId = 4, grouping = "equal-values"),
                       list(fieldId = 9, grouping = "equal-values")))

## End(Not run)
```

---

run_report *Run a report*

Description

Run a report and get its data.

Usage

```
run_report(
  subdomain,
  auth,
  table_id,
  report_id,
  agent = NULL,
  skip = 0,
  top = 0,
```

```
  type_suffix = FALSE,
  paginate = TRUE
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
table_id	Character vector with one element. Found in the URL of a Quickbase table.
report_id	Character vector with one element. Found in the 'Reports & Charts' page in Quickbase and in the report URL.
agent	Optional. Character vector with one element. Describes user/agent making API call.
skip	Optional. Integer. The number of rows to skip from the top of a record set.
top	Optional. Integer. The limit on the number of records to pull starting at the top of a record set.
type_suffix	Optional. Logical. Set TRUE to append each field label with its Quickbase data type.
paginate	Optional. Logical. Set TRUE to recursively call the API until all report pages are collected.

Value

A tibble.

References

Quickbase API documentation

Examples

```
## Not run:

   # Get all data in a report
   my_tibble <- run_report(subdomain = "abc",
       auth = keyring::key_get("qb_example"),
       table_id = "bn9d8iesz",
       report_id = "1")

   # Get rows 3 to 6 from a report
   my_tibble <- run_report(subdomain = "abc.quickbase.com",
       auth = keyring::key_get("qb_example"),
       table_id = "bn9d8iesz",
       report_id = "1",
       skip = 2,
       top = 3)
```

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

---

summarize_app *Summarize an app*

Description

Get metadata for an app, its tables, and its fields.

Usage

```
summarize_app(subdomain, auth, app_id, agent = NULL)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
app_id	Character vector with one element. Unique identifier of an app. Found in the URL of the app's homepage after /db/
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

A list of tibbles.

Examples

```
## Not run:
   summarize_app(subdomain = "abc",
                 auth = keyring::key_get("qb_example"),
                 app_id = "bsf5hphe5")

## End(Not run)
```

update_records *Insert/Update records*

Description

Insert and/or update record(s) in a table. Update can use the key field on the table, or any other supported unique field. Refer to the Field types page for more information about how each field type should be formatted. This operation allows for incremental processing of successful records, even when some of the records fail. This endpoint supports a maximum payload size of 10MB.

Usage

```
update_records(
  subdomain,
  auth,
  to,
  records,
  mergeFieldId = 3,
  fieldsToReturn = list(3),
  agent = NULL
)
```

Arguments

subdomain	Character vector with one element. Found at the beginning of the Quickbase URL. Realm specific.
auth	Character vector with one element. The Quickbase authentication scheme you are using to authenticate the request (e.g., user token).
to	Character vector. Identifier of the target table.
records	Tibble containing the data you want to insert/update in the target table, where column names correspond to field identifiers. Alternatively, a list containing record data as json if your data contain User or List-user data types.
mergeFieldId	Character vector. Field identifier of the key field or unique field to merge upon. Defaults to 3, representing Record ID#.
fieldsToReturn	Character vector of field identifiers. Returns data for field 3 (Record ID#) in addition to any field identifiers supplied.
agent	Optional. Character vector with one element. Describes user/agent making API call.

Value

List of JSON containing return data requested via the fieldsToReturn argument and metadata regarding created/updated records, referenced but unchanged records, and records having any errors while being processed.

Examples

```
## Not run:
  new_data <- dplyr::tibble(`3` = c(5, 7), `6` = c("A", "B"))
  update_records(subdomain = "bhi",
                 auth = keyring::key_get("qb_example"),
                 to = "bn9d8iesz",
                 records = new_data)

## End(Not run)
```

Index

clone_token, 2
copy_app, 3
create_app, 4

deactivate_token, 6
delete_app, 7
delete_fields, 8
delete_records, 8
delete_token, 9

get_app, 10
get_app_events, 11
get_fields, 12
get_report, 13
get_reports, 14
get_tables, 14
get_users, 15

qb_run, 16
query_records, 18

run_report, 19

summarize_app, 21

update_records, 22