Package 'fireData'

September 29, 2026

Title Connect to 'Google Firebase'

Version 2.0.2

Description Provides an interface to 'Google Firebase' services <https://firebase.google.com/>, including 'Firebase Realtime Database', 'Cloud Firestore', 'Firebase Authentication', and 'Cloud Storage for Firebase'. Supports interactive use and 'shiny' applications as well as automated server-side workflows. Data frames and objects can be stored and retrieved, users can be authenticated, and files can be managed through the services' application programming interfaces.

Depends R (>= 4.1.0)

License MIT + file LICENSE

URL https://github.com/Kohze/fireData

BugReports https://github.com/Kohze/fireData/issues

Encoding UTF-8

RoxygenNote 7.3.2

Imports httr (>= 1.4.0), jsonlite (>= 1.8.0), curl (>= 5.0.0), R6 (>= 2.5.0), openssl (>= 2.0.0), yaml (>= 2.3.0)

Suggests testthat (>= 3.2.0), shiny (>= 1.7.0), shinyjs, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Robin Gounder [aut, cre] (ORCID: <https://orcid.org/0009-0008-7755-7081>), Paul Spende [aut], Kevin McGinley [ctb]

Maintainer Robin Gounder <robin@gounder.com>

Repository CRAN

Date/Publication 2026-09-29 14:20:24 UTC

Contents

`anonymous_login` *Anonymous Login (Legacy)*

Description

Legacy anonymous login. Use auth_anonymous() instead.

Usage

```
anonymous_login(project_api)
```

Arguments

`project_api`	The Firebase Project API key

Value

Authentication response

`auth` *User Authentication (Legacy)*

Description

Legacy authentication function. Use auth_sign_in() instead.

Usage

```
auth(projectAPI, email = "prompt", password = "prompt")
```

Arguments

`projectAPI`	The Firebase Project API key
`email`	User email
`password`	User password

Value

Authentication response

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
auth(projectAPI = "AIzaSy...", email = "user@example.com", password = "password")
}
```

`auth_anonymous` *Sign In Anonymously*

Description

Signs in a user anonymously. Anonymous users can be upgraded to full accounts later by linking credentials.

Usage

```
auth_anonymous(conn = NULL, api_key = NULL)
```

Arguments

`conn`	Firebase connection object (or NULL to use config)
`api_key`	Firebase API key (used if conn is NULL)

Value

Authentication response containing idToken, refreshToken, and localId

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "AIzaSy...")
result <- auth_anonymous(conn)
}
```

auth_create_user *Create User Account*

Description

Creates a new user account with email and password.

Usage

```
auth_create_user(conn = NULL, email = NULL, password = NULL, api_key = NULL)
```

Arguments

conn	Firebase connection object (or NULL to use config)
email	User email address
password	User password (min 6 characters)
api_key	Firebase API key (used if conn is NULL)

Value

Registration response containing idToken, email, refreshToken, expiresIn, and localId

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "AIzaSy...")
result <- auth_create_user(conn, "newuser@example.com", "password123")
}
```

auth_delete_user *Delete User Account*

Description

Deletes the authenticated user's account.

Usage

```
auth_delete_user(conn = NULL, id_token = NULL, api_key = NULL)
```

Arguments

conn	Firebase connection object with valid token
id_token	ID token (used if conn doesn't have token)
api_key	Firebase API key

Value

Empty response on success

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
auth_delete_user(conn)
}
```

---

auth_get_user *Get Current User Data*

Description

Retrieves account info for the currently authenticated user.

Usage

```
auth_get_user(conn = NULL, id_token = NULL, api_key = NULL)
```

Arguments

conn	Firebase connection object with valid token
id_token	ID token (used if conn doesn't have token)
api_key	Firebase API key

Value

User account information

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
user_info <- auth_get_user(conn)
}
```

auth_google *Sign In with Google*

Description

Initiates Google OAuth flow and signs in to Firebase.

Usage

```
auth_google(
  conn = NULL,
  client_id = NULL,
  client_secret = NULL,
  request_uri,
  redirect_uri = httr::oauth_callback(),
  return_idp_credential = TRUE,
  cache = FALSE,
  api_key = NULL
)
```

Arguments

conn	Firebase connection object (or NULL to use config)
client_id	Google OAuth client ID
client_secret	Google OAuth client secret
request_uri	Request URI for OAuth callback
redirect_uri	Redirect URI (defaults to httr callback)
return_idp_credential	Whether to return OAuth credentials
cache	Whether to cache the OAuth token
api_key	Firebase API key

Value

Authentication response

Examples

```
if (interactive()) {
# Requires a configured Firebase project and OAuth client credentials.
result <- auth_google(
  conn,
  client_id = "your-client-id.apps.googleusercontent.com",
  client_secret = "your-secret",
  request_uri = "http://localhost"
)
}
```

auth_oauth *Sign In with OAuth Credentials*

Description

Signs in a user with OAuth credentials from a third-party provider.

Usage

```
auth_oauth(
  conn = NULL,
  request_uri,
  post_body,
  return_idp_credential = TRUE,
  api_key = NULL
)
```

Arguments

conn	Firebase connection object (or NULL to use config)
request_uri	The URI to which the IDP redirects the user back
post_body	OAuth credential (ID token or access token) and provider ID
return_idp_credential	Whether to return the OAuth credential
api_key	Firebase API key

Value

Authentication response

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# After obtaining OAuth token from provider
post_body <- "id_token=<google_id_token>&providerId=google.com"
result <- auth_oauth(conn, "http://localhost", post_body)
}
```

auth_refresh_token *Refresh Authentication Token*

Description

Refreshes an expired ID token using a refresh token.

Usage

```
auth_refresh_token(conn = NULL, refresh_token, api_key = NULL)
```

Arguments

conn	Firebase connection object (or NULL to use config)
refresh_token	The refresh token from initial authentication
api_key	Firebase API key (used if conn is NULL)

Value

Response containing new id_token, refresh_token, expires_in, token_type, and user_id

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# Refresh using stored refresh token
new_tokens <- auth_refresh_token(conn, old_refresh_token)
}
```

auth_reset_password *Send Password Reset Email*

Description

Sends a password reset email to the specified address.

Usage

```
auth_reset_password(conn = NULL, email, api_key = NULL)
```

Arguments

conn	Firebase connection object (or NULL to use config)
email	User email address
api_key	Firebase API key (used if conn is NULL)

Value

Response indicating success

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
auth_reset_password(conn, "user@example.com")
}
```

---

auth_sign_in *Sign In with Email and Password*

Description

Authenticates a user with email and password credentials.

Usage

```
auth_sign_in(conn = NULL, email = NULL, password = NULL, api_key = NULL)
```

Arguments

conn	Firebase connection object (or NULL to use config)
email	User email address
password	User password
api_key	Firebase API key (used if conn is NULL)

Value

Authentication response containing idToken, refreshToken, localId, email, and expiresIn

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# With connection object
conn <- firebase_connect(project_id = "my-project", api_key = "AIzaSy...")
result <- auth_sign_in(conn, "user@example.com", "password123")

# Update connection with token
conn <- firebase_set_token(conn, result)

# Without connection (uses config/environment)
result <- auth_sign_in(email = "user@example.com", password = "password123")
}
```

auth_update_profile *Update User Profile*

Description

Updates the display name and/or photo URL of the authenticated user.

Usage

```
auth_update_profile(
  conn = NULL,
  display_name = NULL,
  photo_url = NULL,
  id_token = NULL,
  api_key = NULL
)
```

Arguments

conn	Firebase connection object with valid token
display_name	New display name (or NULL to keep current)
photo_url	New photo URL (or NULL to keep current)
id_token	ID token (used if conn doesn't have token)
api_key	Firebase API key

Value

Updated user data

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
auth_update_profile(conn, display_name = "John Doe")
}
```

createUser *Create User (Legacy)*

Description

Legacy user creation. Use auth_create_user() instead.

Usage

```
createUser(projectAPI, email = "prompt", password = "prompt")
```

Arguments

projectAPI	The Firebase Project API key
email	User email
password	User password

Value

Registration response

create_token *Create Firebase Token from Auth Response*

Description

Factory function to create a FirebaseToken from an authentication response.

Usage

```
create_token(auth_response, api_key = NULL)
```

Arguments

auth_response	Response from Firebase auth API
api_key	Firebase API key for token refresh

Value

A FirebaseToken object

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# After successful authentication
response <- auth_sign_in(conn, "user@example.com", "password")
token <- create_token(response, api_key = "your-api-key")
}
```

---

dataBackup *Database Backup (Legacy)*

Description

Legacy backup function. Use rtdb_backup() instead.

Usage

```
dataBackup(projectURL, secretKey = "prompt", fileName)
```

Arguments

projectURL	Firebase database URL
secretKey	Admin secret key
fileName	Output file name

Value

File name

---

delete *Delete Data (Legacy)*

Description

Legacy delete function. Use rtdb_delete() instead.

Usage

```
delete(x, projectURL, directory = "main", token = "none")
```

Arguments

x	Data (unused, for signature compatibility)
projectURL	Firebase database URL
directory	Database path
token	Authentication token

Value

Path

---

delete_storage *Delete from Storage (Legacy)*

Description

Legacy function. Use storage_delete() instead.

Usage

```
delete_storage(
  bucket_name,
  object_name,
  web_client_id = "prompt",
  web_client_secret = "prompt"
)
```

Arguments

bucket_name	Storage bucket name
object_name	Object name to delete
web_client_id	OAuth client ID
web_client_secret	OAuth client secret

Value

NULL on success

deploy_rmarkdown *Deploy R Markdown (Legacy)*

Description

Legacy function to deploy R Markdown to storage. Rendering uses a temporary directory for output, intermediate files, and the knitting working directory. Use absolute paths for input resources in the document. Temporary render files are removed after the upload attempt.

Usage

```
deploy_rmarkdown(
  rmarkdown_path,
  bucket_name,
  object_name,
  web_client_id = "prompt",
  web_client_secret = "prompt"
)
```

Arguments

rmarkdown_path	Path to R Markdown file
bucket_name	Storage bucket
object_name	Output object name
web_client_id	OAuth client ID
web_client_secret	OAuth client secret

Value

Upload result

download *Download Data (Legacy)*

Description

Legacy download function. Use rtdb_get() instead.

Usage

```
download(
  projectURL,
  fileName,
  secretKey = "none",
  token = "none",
  isClass = FALSE
)
```

Arguments

projectURL	Firebase database URL
fileName	Database path
secretKey	Admin secret key
token	Authentication token
isClass	Whether data is S4 class

Value

Downloaded data

---

dynlink_create *Create Dynamic Link*

Description

Legacy interface for the discontinued Firebase Dynamic Links service. Requests to the upstream service no longer succeed.

Usage

```
dynlink_create(
  conn = NULL,
  link,
  domain_uri_prefix,
  short = TRUE,
  social_title = NULL,
  social_description = NULL,
  social_image_link = NULL,
  android_package_name = NULL,
  android_fallback_link = NULL,
  ios_bundle_id = NULL,
  ios_fallback_link = NULL,
  api_key = NULL
)
```

Arguments

conn	Firebase connection object (or NULL to use config)
link	The URL you want to shorten/wrap
domain_uri_prefix	Your Firebase Dynamic Links domain (e.g., "https://example.page.link")
short	Whether to create a short link (TRUE) or unguessable link (FALSE)
social_title	Title for social media previews
social_description	Description for social media previews
social_image_link	Image URL for social media previews
android_package_name	Android app package name
android_fallback_link	Fallback URL for Android
ios_bundle_id	iOS app bundle ID
ios_fallback_link	Fallback URL for iOS
api_key	Firebase API key (used if conn is NULL)

Value

This function always raises an error because the upstream service has been shut down.

Examples

```
# The retired service is unavailable; this reports an error without a request.
try(dynlink_create(
  link = "https://example.com/page",
  domain_uri_prefix = "https://example.page.link"
))
```

FirebaseToken *Firebase Token Class*

Description

Represents an authenticated Firebase token with automatic refresh capability.

Public fields

`id_token` Firebase ID token (JWT)

`refresh_token` Refresh token for obtaining new ID tokens

`local_id` User's local ID (UID)

`email` User's email (if available)

`expires_at` POSIXct timestamp when the token expires

`api_key` Firebase API key (needed for refresh) Create a new FirebaseToken

Methods

Public methods:

- FirebaseToken$new()
- FirebaseToken$is_expired()
- FirebaseToken$refresh()
- FirebaseToken$get_token()
- FirebaseToken$print()
- FirebaseToken$clone()

Method new():

Usage:

```
FirebaseToken$new(auth_response, api_key = NULL)
```

Arguments:

auth_response Response from Firebase auth API

api_key Firebase API key for token refresh

Returns: A new FirebaseToken object Check if token is expired

Method is_expired():

Usage:

```
FirebaseToken$is_expired(buffer_seconds = 300)
```

Arguments:

buffer_seconds Seconds before actual expiry to consider expired

Returns: TRUE if token is expired or will expire within buffer Refresh the token if expired

Method refresh():

Usage:

```
FirebaseToken$refresh(force = FALSE)
```

Arguments:

force Force refresh even if not expired

Returns: Self (invisibly) Get the current valid token
Automatically refreshes if expired.

Method get_token():

Usage:

```
FirebaseToken$get_token()
```

Returns: The ID token string Print token summary

Method print():

Usage:

```
FirebaseToken$print(...)
```

Arguments:

... Ignored

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FirebaseToken$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`firebase_auth` *Firebase Authentication Service*

Description

Functions for user authentication with Firebase Auth

`firebase_close` *Close Firebase Connection*

Description

Clears the connection's authentication state. Note that this doesn't actually close any persistent connections since the REST API is stateless.

Usage

```
firebase_close(conn)
```

Arguments

`conn`	Firebase connection object

Value

Updated connection with cleared auth state

`firebase_config` *Configuration Management for fireData*

Description

Provides configuration loading and management for Firebase settings

firebase_config_clear *Clear Firebase Configuration*

Description

Clears session configuration values.

Usage

```
firebase_config_clear(keys = NULL)
```

Arguments

keys	Specific keys to clear, or NULL to clear all

Value

Invisibly returns NULL

Examples

```
# Clear specific keys
firebase_config_clear(c("api_key", "project_id"))

# Clear all
firebase_config_clear()
```

firebase_config_get *Get Firebase Configuration Value*

Description

Retrieves a configuration value using the following priority:

1. Explicitly passed value
2. Session configuration (set via firebase_config_set)
3. Environment variables
4. Config file (~/.firedata/config.yml or .firedata.yml)
5. Default value

Usage

```
firebase_config_get(key, value = NULL, default = NULL, profile = "default")
```

Arguments

key	Configuration key to retrieve
value	Explicit value (highest priority)
default	Default value if not found
profile	Configuration profile (for config file)

Value

The configuration value or default

Examples

```
# Get API key from environment or config
api_key <- firebase_config_get("api_key")

# Get with explicit value override
api_key <- firebase_config_get("api_key", value = "my-api-key")
```

---

firebase_config_load *Load Configuration from File*

Description

Loads configuration from a YAML file into session configuration.

Usage

```
firebase_config_load(path = NULL, profile = "default")
```

Arguments

path	Path to config file. If NULL, searches in standard locations.
profile	Configuration profile to load (default: "default")

Value

Invisibly returns the loaded configuration

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# Load from default locations
firebase_config_load()

# Load specific file
```

```
firebase_config_load("path/to/config.yml")

# Load specific profile
firebase_config_load(profile = "production")
}
```

`firebase_config_set` *Set Firebase Configuration Values for Session*

Description

Sets configuration values that persist for the current R session.

Usage

```
firebase_config_set(...)
```

Arguments

`...`	Named configuration values (project_id, api_key, database_url, storage_bucket)

Value

Invisibly returns the previous values

Examples

```
firebase_config_set(
  project_id = "my-project",
  api_key = "AIzaSy..."
)
# Clear the example's session settings when finished
firebase_config_clear(c("project_id", "api_key"))
```

`firebase_config_show` *Get All Current Configuration*

Description

Returns all currently set configuration values from all sources, with sensitive values masked. Use `print()` to display the returned list.

Usage

```
firebase_config_show(profile = "default")
```

Arguments

profile	Configuration profile for file lookup

Value

Named list of all configuration values

Examples

```
# View current configuration
config <- firebase_config_show()
print(config)
```

---

firebase_config_wizard

Interactive Configuration Wizard

Description

Guides the user through setting up Firebase configuration interactively.

Usage

```
firebase_config_wizard(save = FALSE, path = NULL)
```

Arguments

save	Whether to offer to save configuration to a file. Defaults to FALSE, which only configures the current session.
path	Explicit output file path, required when save = TRUE. The parent directory must already exist. No default file path is used.

Value

Invisibly returns the configuration list

Examples

```
if (interactive()) {
# Run interactive setup
firebase_config_wizard()

# Optionally save to an explicitly chosen file
config_file <- tempfile(fileext = ".yml")
firebase_config_wizard(save = TRUE, path = config_file)
unlink(config_file)
}
```

`firebase_connect` *Create a Firebase Connection*

Description

Creates a connection object that stores Firebase project configuration and can be passed to various fireData functions. The connection handles configuration resolution from multiple sources (arguments, environment, config files).

Usage

```
firebase_connect(
  project_id = NULL,
  api_key = NULL,
  database_url = NULL,
  storage_bucket = NULL,
  credentials = NULL,
  token = NULL
)
```

Arguments

`project_id`	Firebase project ID (e.g., "my-project-12345")
`api_key`	Firebase Web API key
`database_url`	Realtime Database URL. Supply the exact URL shown in the Firebase console for regional or non-default databases. If NULL, a US Central default URL is constructed from `project_id`.
`storage_bucket`	Cloud Storage bucket name. If NULL, defaults to the current `project_id.firebasestorage.app` format. Projects whose default bucket was created before September 2024 should pass their existing `project_id.appspot.com` bucket explicitly.
`credentials`	Service account credentials (path to JSON file or ServiceAccountCredentials object)
`token`	User authentication token (FirebaseToken object or token string)

Value

A firebase_connection S3 object

Examples

```
# Create connection with explicit values
conn <- firebase_connect(
  project_id = "my-project",
  api_key = "AIzaSy..."
)
```

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# These alternatives require your own configuration or credentials.
# Create connection from environment variables
# (Set FIREBASE_PROJECT_ID, FIREBASE_API_KEY, etc.)
conn <- firebase_connect()

# Create connection with service account
conn <- firebase_connect(
  project_id = "my-project",
  credentials = "path/to/service-account.json"
)
}
```

---

firebase_connection *Firebase Connection Object*

Description

Provides a connection object for Firebase operations

---

firebase_credentials *Credential Management for fireData*

Description

Provides credential storage, retrieval, and token management

---

firebase_dynamic_links *Firebase Dynamic Links Service*

Description

Functions for creating Firebase Dynamic Links (short URLs)

Note

Firebase Dynamic Links shut down on August 25, 2025. These functions are retained for compatibility, but the upstream service no longer creates or serves links. Use another deep-linking provider for new projects.

firebase_firestore *Cloud Firestore Service*

Description

Functions for interacting with Google Cloud Firestore NoSQL database

Overview

Cloud Firestore is a flexible, scalable NoSQL cloud database. Unlike the Realtime Database which stores data as one large JSON tree, Firestore stores data in documents organized into collections.

Data Model

- **Collections**: Containers for documents (like folders)
- **Documents**: Individual records containing fields (like files)
- **Fields**: Key-value pairs within documents
- **Subcollections**: Collections nested within documents

Authentication

Firestore requires authentication. Use either:

- Service account credentials (recommended for server-side)
- User authentication token from `auth_sign_in()`

firebase_rtdb *Firebase Realtime Database Service*

Description

Functions for CRUD operations on Firebase Realtime Database

firebase_set_token *Update Connection Token*

Description

Updates the authentication token on a connection.

Usage

```
firebase_set_token(conn, token)
```

Arguments

conn	Firebase connection object
token	New token (FirebaseToken object or string)

Value

Updated connection object

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "...")
auth_result <- auth_sign_in(conn, email, password)
conn <- firebase_set_token(conn, auth_result)
}
```

firebase_shiny *Shiny Integration for fireData*

Description

Functions to integrate Firebase authentication with Shiny apps

firebase_storage *Firebase Cloud Storage Service*

Description

Functions for file storage operations with Firebase/Google Cloud Storage

firebase_utils *Utility Functions for fireData*

Description

Internal utility functions for data conversion and path handling

firestore_add *Add Firestore Document*

Description

Creates a new document with an auto-generated ID.

Usage

```
firestore_add(conn = NULL, collection, data, project_id = NULL, token = NULL)
```

Arguments

conn	Firebase connection object
collection	Collection path
data	List of fields to store
project_id	Project ID (uses connection or config if NULL)
token	Authentication token (uses connection if NULL)

Value

The created document data including the generated ID

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project")

# Add document with auto-generated ID
result <- firestore_add(conn, "messages", list(
  text = "Hello, Firestore!",
  timestamp = Sys.time(),
  author = "R User"
))

# Get the generated document ID
print(result$.id)
}
```

`firestore_delete` *Delete Firestore Document*

Description

Deletes a document from Cloud Firestore.

Usage

```
firestore_delete(
  conn = NULL,
  collection,
  document,
  project_id = NULL,
  token = NULL
)
```

Arguments

`conn`	Firebase connection object
`collection`	Collection path
`document`	Document ID
`project_id`	Project ID (uses connection or config if NULL)
`token`	Authentication token (uses connection if NULL)

Value

TRUE if successful

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project")

# Delete a document
firestore_delete(conn, "users", "user123")
}
```

`firestore_get` *Get Firestore Document*

Description

Retrieves a single document from Cloud Firestore.

Usage

```
firestore_get(
  conn = NULL,
  collection,
  document,
  project_id = NULL,
  token = NULL
)
```

Arguments

`conn`	Firebase connection object
`collection`	Collection path (e.g., "users" or "users/uid/posts")
`document`	Document ID
`project_id`	Project ID (uses connection or config if NULL)
`token`	Authentication token (uses connection if NULL)

Value

Document data as a list, or NULL if not found

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project")
conn <- firebase_set_token(conn, auth_sign_in(conn, "user@example.com", "password"))

# Get a single document
user <- firestore_get(conn, "users", "user123")
print(user$name)

# Get from subcollection
post <- firestore_get(conn, "users/user123/posts", "post456")
}
```

`firestore_list` *List Firestore Documents*

Description

Lists documents in a collection.

Usage

```
firestore_list(
  conn = NULL,
  collection,
  page_size = 100,
  page_token = NULL,
  order_by = NULL,
  project_id = NULL,
  token = NULL
)
```

Arguments

`conn`	Firebase connection object
`collection`	Collection path
`page_size`	Maximum number of documents to return (default 100)
`page_token`	Token for pagination (from previous response)
`order_by`	Field to order by (prefix with "-" for descending)
`project_id`	Project ID (uses connection or config if NULL)
`token`	Authentication token (uses connection if NULL)

Value

List containing documents and optional nextPageToken

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project")

# List all users (up to 100)
result <- firestore_list(conn, "users")
for (user in result$documents) {
  print(user$name)
}

# Paginate through results
```

```
result <- firestore_list(conn, "users", page_size = 10)
while (!is.null(result$nextPageToken)) {
  result <- firestore_list(conn, "users", page_size = 10,
                           page_token = result$nextPageToken)
}
}
```

`firestore_query` *Query Firestore Documents*

Description

Runs a structured query against a Firestore collection.

Usage

```
firestore_query(conn = NULL, collection, project_id = NULL, token = NULL)
```

Arguments

`conn`	Firebase connection object
`collection`	Collection path to query
`project_id`	Project ID (uses connection or config if NULL)
`token`	Authentication token (uses connection if NULL)

Value

A firestore_query object for building the query

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project")

# Query with filters
results <- firestore_query(conn, "users") |>
  fs_where("age", ">=", 18) |>
  fs_where("active", "==", TRUE) |>
  fs_order_by("name") |>
  fs_limit(10) |>
  fs_execute()
}
```

`firestore_set` *Set Firestore Document*

Description

Creates or overwrites a document in Cloud Firestore.

Usage

```
firestore_set(
  conn = NULL,
  collection,
  document,
  data,
  project_id = NULL,
  token = NULL
)
```

Arguments

`conn`	Firebase connection object
`collection`	Collection path
`document`	Document ID
`data`	List of fields to store
`project_id`	Project ID (uses connection or config if NULL)
`token`	Authentication token (uses connection if NULL)

Value

The created/updated document data

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project")

# Create or overwrite a document
firestore_set(conn, "users", "user123", list(
  name = "John Doe",
  email = "john@example.com",
  age = 30,
  active = TRUE
))
}
```

`firestore_update` *Update Firestore Document*

Description

Updates specific fields in a document without overwriting the entire document.

Usage

```
firestore_update(
  conn = NULL,
  collection,
  document,
  data,
  project_id = NULL,
  token = NULL
)
```

Arguments

`conn`	Firebase connection object
`collection`	Collection path
`document`	Document ID
`data`	List of fields to update
`project_id`	Project ID (uses connection or config if NULL)
`token`	Authentication token (uses connection if NULL)

Value

The updated document data

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project")

# Update only specific fields
firestore_update(conn, "users", "user123", list(
  lastLogin = Sys.time(),
  loginCount = 42
))
}
```

fs_execute *Execute Firestore Query*

Description

Executes a Firestore query and returns the results.

Usage

```
fs_execute(query)
```

Arguments

query	A firestore_query object

Value

List of documents matching the query

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
results <- firestore_query(conn, "products") |>
  fs_where("price", "<", 100) |>
  fs_order_by("price", "asc") |>
  fs_limit(20) |>
  fs_execute()
}
```

fs_limit *Limit Firestore Query Results*

Description

Limits the number of documents returned by a query.

Usage

```
fs_limit(query, n)
```

Arguments

query	A firestore_query object
n	Maximum number of documents to return

Value

Updated firestore_query object

---

fs_offset *Offset Firestore Query Results*

Description

Skips a number of documents before returning results.

Usage

```
fs_offset(query, n)
```

Arguments

query	A firestore_query object
n	Number of documents to skip

Value

Updated firestore_query object

---

fs_order_by *Add Order By to Firestore Query*

Description

Adds ordering to a Firestore query.

Usage

```
fs_order_by(query, field, direction = "asc")
```

Arguments

query	A firestore_query object
field	Field name to order by
direction	Sort direction: "asc" or "desc"

Value

Updated firestore_query object

fs_select *Select Fields in Firestore Query*

Description

Specifies which fields to return in query results.

Usage

```
fs_select(query, ...)
```

Arguments

query	A firestore_query object
...	Field names to select

Value

Updated firestore_query object

fs_where *Add Where Filter to Firestore Query*

Description

Adds a field filter to a Firestore query.

Usage

```
fs_where(query, field, op, value)
```

Arguments

query	A firestore_query object
field	Field name to filter on
op	Comparison operator: "==", "!=", "<", "<=", ">", ">=", "array-contains", "array-contains-any", "in", "not-in"
value	Value to compare against

Value

Updated firestore_query object

Examples

```
conn <- firebase_connect(project_id = "my-project")
query <- firestore_query(conn, "users") |>
  fs_where("age", ">=", 21) |>
  fs_where("status", "==", "active")
```

`get_dynamic_link` *Create Dynamic Link (Legacy)*

Description

Legacy function. Use dynlink_create() instead.

Usage

```
get_dynamic_link(
  project_api,
  domain,
  link,
  short = TRUE,
  social_title = "",
  social_description = "",
  social_image_link = ""
)
```

Arguments

`project_api`	Firebase API key
`domain`	Dynamic Links domain
`link`	URL to shorten
`short`	Whether to create short link
`social_title`	Social preview title
`social_description`	Social preview description
`social_image_link`	Social preview image

Value

This function always raises an error because the upstream service has been shut down.

get_storage *Get Storage Metadata (Legacy)*

Description

Legacy function. Use storage_get_metadata() instead.

Usage

```
get_storage(
  bucket_name,
  object_name,
  web_client_id = "prompt",
  web_client_secret = "prompt"
)
```

Arguments

bucket_name	Storage bucket name
object_name	Object name
web_client_id	OAuth client ID
web_client_secret	OAuth client secret

Value

Object metadata

get_url *Get Storage URL (Legacy)*

Description

Legacy function. Use storage_get_url() instead.

Usage

```
get_url(
  bucket_name,
  object_name,
  web_client_id = "prompt",
  web_client_secret = "prompt"
)
```

Arguments

bucket_name	Storage bucket name
object_name	Object name
web_client_id	OAuth client ID
web_client_secret	OAuth client secret

Value

Download URL

---

get_user_token *Get User Token*

Description

Gets the Firebase token from authenticated user in Shiny context.

Usage

```
get_user_token(user)
```

Arguments

user	The reactiveValues object used with shiny_auth_server

Value

The ID token string or NULL if not authenticated

---

google_devstorage_read_only *Get Storage Token with Read-Only Access (Legacy)*

Description

Legacy function for OAuth token.

Usage

```
google_devstorage_read_only(
  web_client_id = "prompt",
  web_client_secret = "prompt"
)
```

Arguments

web_client_id	OAuth client ID
web_client_secret	OAuth client secret

Value

OAuth token

---

google_devstorage_read_write *Get Storage Token with Read/Write Access (Legacy)*

Description

Legacy function for OAuth token.

Usage

```
google_devstorage_read_write(
  web_client_id = "prompt",
  web_client_secret = "prompt"
)
```

Arguments

web_client_id	OAuth client ID
web_client_secret	OAuth client secret

Value

OAuth token

---

google_firestore *Get Firestore Token (Legacy)*

Description

Legacy function for Firestore OAuth token.

Usage

```
google_firestore(
  web_client_id = "prompt",
  web_client_secret = "prompt",
  cache = FALSE
)
```

Arguments

web_client_id	OAuth client ID
web_client_secret	OAuth client secret
cache	Whether to cache token

Value

OAuth token

---

google_login *Google Login (Legacy)*

Description

Legacy Google login. Use auth_google() instead.

Usage

```
google_login(
  project_api,
  web_client_id = "prompt",
  web_client_secret = "prompt",
  request_uri,
  redirect_uri = httr::oauth_callback(),
  return_idp_credential = TRUE,
  cache = FALSE
)
```

Arguments

project_api	Firebase API key
web_client_id	Google OAuth client ID
web_client_secret	Google OAuth client secret
request_uri	Request URI
redirect_uri	Redirect URI
return_idp_credential	Whether to return IDP credential
cache	Whether to cache token

Value

Authentication response

---

is_authenticated *Check Authentication Status*

Description

Helper to check if user is authenticated in Shiny context.

Usage

```
is_authenticated(user)
```

Arguments

user	The reactiveValues object used with shiny_auth_server

Value

TRUE if user is logged in, FALSE otherwise

---

is_firebase_connection *Check if Object is a Firebase Connection*

Description

Check if Object is a Firebase Connection

Usage

```
is_firebase_connection(x)
```

Arguments

x	Object to check

Value

TRUE if x is a firebase_connection

list_storage *List Storage (Legacy)*

Description

Legacy function. Use storage_list() instead.

Usage

```
list_storage(
  bucket_name,
  web_client_id = "prompt",
  web_client_secret = "prompt"
)
```

Arguments

bucket_name	Storage bucket name
web_client_id	OAuth client ID
web_client_secret	OAuth client secret

Value

List of objects

load_service_account *Load Service Account Credentials*

Description

Creates ServiceAccountCredentials from a file or environment.

Usage

```
load_service_account(path = NULL)
```

Arguments

path	Path to service account JSON file (optional)

Value

A ServiceAccountCredentials object

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# From file
creds <- load_service_account("path/to/service-account.json")

# From GOOGLE_APPLICATION_CREDENTIALS environment variable
creds <- load_service_account()
}
```

`oauth_helpers` *OAuth2 Token Helpers*

Description

Functions to obtain OAuth2 tokens for various Google/Firebase services.

`o_auth_login` *OAuth Login (Legacy)*

Description

Legacy OAuth login. Use auth_oauth() instead.

Usage

```
o_auth_login(project_api, request_uri, post_body, return_idp_credential)
```

Arguments

`project_api`	Firebase API key
`request_uri`	Request URI
`post_body`	OAuth post body
`return_idp_credential`	Whether to return IDP credential

Value

Authentication response

patch *Patch Data (Legacy)*

Description

Legacy patch function. Use rtdb_update() instead.

Usage

```
patch(x, projectURL, directory = "main", token = "none")
```

Arguments

x	Data to update
projectURL	Firebase database URL
directory	Database path
token	Authentication token

Value

Path

path_sanitize *Sanitize Firebase Path*

Description

Replaces disallowed characters in Firebase paths with hyphens. Firebase paths cannot contain: . $ # [ ] /

[ ]: R:%20

Usage

```
path_sanitize(path)

path_check(path)
```

Arguments

path	The database path string

Value

The sanitized path string

Examples

```
# Path with invalid characters
path_sanitize("users/john.doe")
# Returns: "users-john-doe" with a warning

# Valid path passes through unchanged
path_sanitize("users-johndoe")
# Returns: "users-johndoe"
```

`print.firebase_connection` *Print Firebase Connection*

Description

Print Firebase Connection

Usage

```
## S3 method for class 'firebase_connection'
print(x, ...)
```

Arguments

`x`	A firebase_connection object
`...`	Additional arguments (ignored)

Value

Invisibly returns x

`print.firestore_query` *Print Firestore Query*

Description

Print Firestore Query

Usage

```
## S3 method for class 'firestore_query'
print(x, ...)
```

Arguments

`x`	A firestore_query object
`...`	Additional arguments (ignored)

Value

Invisibly returns x

`print.rtdb_query` *Print Query*

Description

Print Query

Usage

```
## S3 method for class 'rtdb_query'
print(x, ...)
```

Arguments

`x`	Query object
`...`	Ignored

Value

Invisibly returns x

`put` *Put Data (Legacy)*

Description

Legacy put function. Use rtdb_set() instead.

Usage

```
put(x, projectURL, directory = "main", token = "none")
```

Arguments

`x`	Data to write
`projectURL`	Firebase database URL
`directory`	Database path
`token`	Authentication token

Value

Path

query_end_at *End Query at Value*

Description

End Query at Value

Usage

```
query_end_at(query, value)
```

Arguments

query	Query builder object
value	Ending value (inclusive)

Value

Updated query builder

query_equal_to *Filter Query by Exact Value*

Description

Filter Query by Exact Value

Usage

```
query_equal_to(query, value)
```

Arguments

query	Query builder object
value	Value to match exactly

Value

Updated query builder

query_execute *Execute Query*

Description

Execute Query

Usage

```
query_execute(query, token = NULL)
```

Arguments

query	Query builder object
token	Authentication token (optional)

Value

Query results

query_limit_to_first *Limit Query to First N Results*

Description

Limit Query to First N Results

Usage

```
query_limit_to_first(query, n)
```

Arguments

query	Query builder object
n	Number of results

Value

Updated query builder

query_limit_to_last *Limit Query to Last N Results*

Description

Limit Query to Last N Results

Usage

```
query_limit_to_last(query, n)
```

Arguments

query	Query builder object
n	Number of results

Value

Updated query builder

query_order_by *Order Query by Field*

Description

Order Query by Field

Usage

```
query_order_by(query, field)
```

Arguments

query	Query builder object
field	Field to order by ("$key", "$value", "$priority", or child key)

Value

Updated query builder

query_start_at *Start Query at Value*

Description

Start Query at Value

Usage

```
query_start_at(query, value)
```

Arguments

query	Query builder object
value	Starting value (inclusive)

Value

Updated query builder

resetPassword *Reset Password (Legacy)*

Description

Legacy password reset. Use auth_reset_password() instead.

Usage

```
resetPassword(projectAPI, email)
```

Arguments

projectAPI	The Firebase Project API key
email	User email

Value

Success or warning message

rtdb_backup *Backup Realtime Database*

Description

Downloads the entire database to a JSON file.

Usage

```
rtdb_backup(conn, file_name, token = NULL)
```

Arguments

conn	Firebase connection object
file_name	Output file path
token	Authentication token (typically admin/service account)

Value

The file name where backup was saved

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(
  project_id = "my-project",
  credentials = "service-account.json"
)
backup_file <- tempfile(fileext = ".json")
rtdb_backup(conn, backup_file)
unlink(backup_file)
}
```

rtdb_delete *Delete Data from Realtime Database*

Description

Removes data at the specified path.

Usage

```
rtdb_delete(conn, path, token = NULL)
```

Arguments

conn	Firebase connection object
path	Database path to delete
token	Authentication token (overrides connection token)

Value

The path that was deleted

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "...")

# Delete user
rtdb_delete(conn, "users/user123")
}
```

---

rtdb_get *Get Data from Realtime Database*

Description

Retrieves data from the specified path in the Realtime Database.

Usage

```
rtdb_get(conn, path, shallow = FALSE, is_class = FALSE, token = NULL)
```

Arguments

conn	Firebase connection object
path	Database path (e.g., "users/user123")
shallow	If TRUE, returns only keys without values (for large datasets)
is_class	If TRUE, data is decoded as an S4 class object
token	Authentication token (overrides connection token)

Value

The data at the specified path, or NULL if not found

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "...")

# Get all users
users <- rtdb_get(conn, "users")

# Get specific user
user <- rtdb_get(conn, "users/user123")

# Get only keys (shallow read)
keys <- rtdb_get(conn, "users", shallow = TRUE)
}
```

`rtdb_push` *Push Data to Realtime Database (Create with Auto-ID)*

Description

Adds data to a list with an auto-generated unique key.

Usage

```
rtdb_push(conn, path, data, token = NULL)
```

Arguments

`conn`	Firebase connection object
`path`	Database path (parent location)
`data`	Data to push
`token`	Authentication token (overrides connection token)

Value

The full path including the generated key

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "...")

# Push new message (auto-generates key)
path <- rtdb_push(conn, "messages", list(text = "Hello", timestamp = Sys.time()))
```

```
# Returns something like "messages/-NxYz..."
}
```

`rtdb_query` *Create Database Query*

Description

Creates a query builder for filtering and sorting database queries.

Usage

```
rtdb_query(conn, path)
```

Arguments

`conn`	Firebase connection object
`path`	Database path to query

Value

A query builder object

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "...")

# Query users ordered by age, limited to 10
results <- rtdb_query(conn, "users") |>
  query_order_by("age") |>
  query_start_at(18) |>
  query_limit_to_first(10) |>
  query_execute()
}
```

rtdb_set *Set Data in Realtime Database (Overwrite)*

Description

Writes data to the specified path, overwriting any existing data. Use rtdb_update() for partial updates.

Usage

```
rtdb_set(conn, path, data, token = NULL)
```

Arguments

conn	Firebase connection object
path	Database path
data	Data to write (data.frame, list, or S4 object)
token	Authentication token (overrides connection token)

Value

The path where data was written

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "...")

# Set user data
rtdb_set(conn, "users/user123", list(name = "John", age = 30))

# Set data frame
rtdb_set(conn, "data/mtcars", mtcars)
}
```

rtdb_update *Update Data in Realtime Database (Partial Update)*

Description

Updates specific fields at the specified path without overwriting other existing data. Use rtdb_set() to completely overwrite.

Usage

```
rtdb_update(conn, path, data, token = NULL)
```

Arguments

conn	Firebase connection object
path	Database path
data	Data to update (only specified fields will be changed)
token	Authentication token (overrides connection token)

Value

The path where data was updated

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", api_key = "...")

# Update only the age field
rtdb_update(conn, "users/user123", list(age = 31))
}
```

shiny_auth_server *Create Shiny Authentication Server*

Description

Creates a login overlay for Shiny applications with support for email/password, Google OAuth, and anonymous authentication.

Usage

```
shiny_auth_server(
  user,
  input,
  output,
  credentials = TRUE,
  goauth = TRUE,
  anonymous = TRUE,
  project_api = NULL,
  web_client_id = NULL,
  web_client_secret = NULL,
  request_uri = NULL
)
```

Arguments

user	A reactiveValues object with a Logged field (e.g., reactiveValues(Logged = FALSE))
input	Shiny input object
output	Shiny output object
credentials	Enable email/password login (default: TRUE)
goauth	Enable Google OAuth login (default: TRUE)
anonymous	Enable anonymous login (default: TRUE)
project_api	Firebase API key
web_client_id	Google OAuth client ID (required if goauth = TRUE)
web_client_secret	Google OAuth client secret (required if goauth = TRUE)
request_uri	OAuth redirect URI

Value

A Shiny UI element containing the login form

Examples

```
if (interactive()) {
library(shiny)
library(fireData)

ui <- fluidPage(
  shinyjs::useShinyjs(),
  uiOutput("app")
)

server <- function(input, output, session) {
  USER <- reactiveValues(Logged = FALSE)
```

```
  output$app <- renderUI({
    if (!USER$Logged) {
      shiny_auth_server(
        user = USER,
        input = input,
        output = output,
        project_api = "your-api-key",
        web_client_id = "your-client-id",
        web_client_secret = "your-client-secret",
        request_uri = "http://localhost"
      )
    } else {
      # Your main app UI
      tagList(
        h1("Welcome!"),
        actionButton("logout", "Logout")
      )
    }
  })
}

shinyApp(ui, server)
}
```

`shiny_auth_ui` *Shiny Authentication UI Helper*

Description

Creates a styled login panel for Shiny apps. Use with shiny_auth_server() for complete authentication.

Usage

```
shiny_auth_ui(title = "Sign In")
```

Arguments

`title`	Title shown on the login panel

Value

A Shiny UI element

Examples

```
if (interactive()) {
ui <- shiny::fluidPage(
  shinyjs::useShinyjs(),
  shiny_auth_ui("Sign In")
```

```
)
}
```

`storage_delete` *Delete File from Storage*

Description

Deletes a file from Firebase Cloud Storage.

Usage

```
storage_delete(conn, object_name, client_id = NULL, client_secret = NULL)
```

Arguments

`conn`	Firebase connection object
`object_name`	Name/path of the object to delete
`client_id`	OAuth client ID
`client_secret`	OAuth client secret

Value

NULL on success

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
storage_delete(conn, "images/old-photo.jpg")
}
```

`storage_download` *Download File from Storage*

Description

Downloads a file from Firebase Cloud Storage.

Usage

```
storage_download(
  conn,
  object_name,
  dest_file = NULL,
  client_id = NULL,
  client_secret = NULL
)
```

Arguments

conn	Firebase connection object
object_name	Name/path of the object in storage
dest_file	Local destination path (if NULL, returns content)
client_id	OAuth client ID
client_secret	OAuth client secret

Value

If dest_file is NULL, returns file content; otherwise returns dest_file path

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# Download to file
dest_file <- tempfile(fileext = ".jpg")
storage_download(conn, "images/photo.jpg", dest_file)
unlink(dest_file)

# Download to memory
content <- storage_download(conn, "data/config.json")
}
```

---

storage_get_metadata *Get Storage Object Metadata*

Description

Retrieves metadata for a storage object.

Usage

```
storage_get_metadata(conn, object_name, client_id = NULL, client_secret = NULL)
```

Arguments

conn	Firebase connection object
object_name	Name/path of the object
client_id	OAuth client ID
client_secret	OAuth client secret

Value

Object metadata

---

storage_get_url *Get Download URL for Storage Object*

Description

Generates a download URL for a storage object.

Usage

```
storage_get_url(conn, object_name, client_id = NULL, client_secret = NULL)
```

Arguments

conn	Firebase connection object
object_name	Name/path of the object
client_id	OAuth client ID
client_secret	OAuth client secret

Value

Download URL string

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
url <- storage_get_url(conn, "images/photo.jpg")
if (interactive()) {
  utils::browseURL(url)
}
}
```

`storage_list` *List Files in Storage*

Description

Lists all files in a storage bucket or prefix.

Usage

```
storage_list(
  conn,
  prefix = NULL,
  delimiter = NULL,
  max_results = 1000,
  client_id = NULL,
  client_secret = NULL
)
```

Arguments

`conn`	Firebase connection object
`prefix`	Filter results to objects with this prefix
`delimiter`	Use "/" to list only immediate children
`max_results`	Maximum number of results
`client_id`	OAuth client ID
`client_secret`	OAuth client secret

Value

List of storage objects

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
# List all files
files <- storage_list(conn)

# List files in a "folder"
images <- storage_list(conn, prefix = "images/")
}
```

`storage_upload` *Upload File to Storage*

Description

Uploads a file to Firebase Cloud Storage.

Usage

```
storage_upload(
  conn,
  file_path,
  object_name,
  content_type = NULL,
  predefined_acl = NULL,
  client_id = NULL,
  client_secret = NULL
)
```

Arguments

`conn`	Firebase connection object
`file_path`	Local path to the file to upload
`object_name`	Name/path for the object in storage (e.g., "images/photo.jpg")
`content_type`	MIME type (auto-detected if NULL)
`predefined_acl`	Optional predefined object ACL (for example, "publicRead" or "private"). The default NULL uses bucket IAM and is compatible with uniform bucket-level access.
`client_id`	OAuth client ID (for user auth)
`client_secret`	OAuth client secret (for user auth)

Value

Storage object metadata including URL

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
conn <- firebase_connect(project_id = "my-project", storage_bucket = "my-bucket")

# Upload with service account
result <- storage_upload(conn, "local/photo.jpg", "images/photo.jpg")

# Get the public URL
```

```
print(result$url)
}
```

`storage_upload_folder` *Upload Folder to Storage*

Description

Uploads all files from a local folder to storage.

Usage

```
storage_upload_folder(
  conn,
  folder_path,
  prefix = "",
  client_id = NULL,
  client_secret = NULL
)
```

Arguments

`conn`	Firebase connection object
`folder_path`	Local folder path
`prefix`	Storage prefix/path for uploaded files
`client_id`	OAuth client ID
`client_secret`	OAuth client secret

Value

List of upload results

Examples

```
# Requires your own Firebase configuration, credentials and example resources.
# Adapt the placeholders and set FIREDATA_RUN_EXAMPLES=true to opt in.
if (identical(Sys.getenv("FIREDATA_RUN_EXAMPLES"), "true")) {
results <- storage_upload_folder(conn, "local/images", "images")
}
```

upload *Upload Data (Legacy)*

Description

Legacy upload function. Use rtdb_push() instead.

Usage

```
upload(x, projectURL, directory = "main", token = "none")
```

Arguments

x	Data to upload
projectURL	Firebase database URL
directory	Database path
token	Authentication token

Value

Path with generated key

upload_folder *Upload Folder (Legacy)*

Description

Legacy function. Use storage_upload_folder() instead.

Usage

```
upload_folder(
  bucket_name,
  web_client_id = "prompt",
  web_client_secret = "prompt",
  folder_path
)
```

Arguments

bucket_name	Storage bucket name
web_client_id	OAuth client ID
web_client_secret	OAuth client secret
folder_path	Local folder path

Value

List of upload results

`upload_storage` *Upload to Storage (Legacy)*

Description

Legacy function. Use storage_upload() instead.

Usage

```
upload_storage(
  bucket_name,
  object_name,
  web_client_id = "prompt",
  web_client_secret = "prompt",
  file_path = NULL,
  predefined_acl = "publicRead"
)
```

Arguments

`bucket_name`	Storage bucket name
`object_name`	Object name in storage
`web_client_id`	OAuth client ID
`web_client_secret`	OAuth client secret
`file_path`	Local file path
`predefined_acl`	Access control

Value

Storage object metadata

Index