Package ‘ebx’

March 16, 2026

Type Package

Title 'Earth Blox' API Client

Version 1.0.0

Maintainer Neil Mayo <n.mayo@earthblox.io>

Description Client library for the 'Earth Blox' API (<https://api.earthblox.io/>). Provides authentication and endpoints for interacting with 'Earth Blox' geospatial analytics services. Compatible with 'Shiny' applications.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports httr2 (>= 1.0.0), jsonlite (>= 1.7.0), R6 (>= 2.5.0), base64enc (>= 0.1-3)

Suggests testthat (>= 3.0.0), withr (>= 2.5.0), shiny (>= 1.7.0)

RoxygenNote 7.3.3

NeedsCompilation no

Author Neil Mayo [aut, cre], Quosient Ltd. [cph]

Repository CRAN

Date/Publication 2026-03-16 19:10:07 UTC

Contents

`AbstractAuthentication` *AbstractAuthentication*

Description

Base R6 class for authentication methods

Public fields

`auth_token` The authentication token

`config` The client configuration

Methods

Public methods:

- `AbstractAuthentication$new()`
- `AbstractAuthentication$has_expired()`
- `AbstractAuthentication$refresh()`
- `AbstractAuthentication$get_headers()`

- AbstractAuthentication$clone()

Method `new()`**:** Create a new AbstractAuthentication object

Usage:

```
AbstractAuthentication$new(config)
```

Arguments:

`config` The client configuration

Returns: A new AbstractAuthentication object

Method `has_expired()`**:** Check if the token has expired

Usage:

```
AbstractAuthentication$has_expired()
```

Returns: TRUE if expired, FALSE otherwise

Method `refresh()`**:** Refresh the authentication token

Usage:

```
AbstractAuthentication$refresh()
```

Returns: self

Method `get_headers()`**:** Get the headers for HTTP requests

Usage:

```
AbstractAuthentication$get_headers()
```

Returns: A named character vector of headers

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`AuthToken` *AuthToken*

Description

R6 class for authentication tokens

Public fields

`token` The access token

`expires` The expiration datetime

Methods

Public methods:

- AuthToken$new()
- AuthToken$save()
- AuthToken$to_list()
- AuthToken$to_json()
- AuthToken$clone()

Method `new()`**:** Create a new AuthToken object

Usage:

```
AuthToken$new(token = NULL, expires = NULL)
```

Arguments:

`token` The access token

`expires` The expiration datetime (as POSIXct or string)

Returns: A new AuthToken object

Method `save()`**:** Save the token to disk

Usage:

```
AuthToken$save(config, filename)
```

Arguments:

`config` The client config

`filename` The filename to save to

Returns: self (invisible)

Method `to_list()`**:** Convert AuthToken to a plain list

Usage:

```
AuthToken$to_list(include_token = FALSE)
```

Arguments:

`include_token` Whether to include the actual token (default: FALSE for security)

Returns: A list containing auth token fields

Method `to_json()`**:** Convert AuthToken to JSON string

Usage:

```
AuthToken$to_json(pretty = TRUE, include_token = FALSE, ...)
```

Arguments:

`pretty` Whether to pretty-print the JSON (default: TRUE)

`include_token` Whether to include the actual token (default: FALSE for security)

`...` Additional arguments passed to jsonlite::toJSON

Returns: A JSON string representation of the auth token

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

auth_token_exists *Check if saved token exists*

Description

Check if a token file exists

Usage

```
auth_token_exists(config, filename)
```

Arguments

config	The client config
filename	The filename to check

Value

TRUE if the file exists, FALSE otherwise

auth_using *Authenticate using an authenticator*

Description

Low-level authentication function

Usage

```
auth_using(authenticator, config = NULL, name = NULL)
```

Arguments

authenticator	The authentication object
config	The client configuration (optional)
name	The client name (optional)

Value

An EbxClient object

auth_using_creds *Authenticate using saved credentials*

Description

Authenticate using saved OAuth credentials from disk

Usage

```
auth_using_creds(filename = API_SECRETS_FILE, name = NULL, config = NULL)
```

Arguments

filename	The credentials filename (optional)
name	The client name (optional)
config	The client configuration (optional)

Value

An EbxClient object

auth_using_env *Authenticate using environment variable*

Description

Authenticate using the EBX_API_TOKEN environment variable

Usage

```
auth_using_env(name = NULL, config = NULL)
```

Arguments

name	The client name (optional)
config	The client configuration (optional)

Value

An EbxClient object

auth_using_oauth *Authenticate using OAuth*

Description

Authenticate using OAuth client credentials

Usage

```
auth_using_oauth(
  client_id = NULL,
  client_secret = NULL,
  name = NULL,
  config = NULL
)
```

Arguments

client_id	The client ID (optional, can be from env)
client_secret	The client secret (optional, can be from env)
name	The client name (optional)
config	The client configuration (optional)

Value

An EbxClient object

BasicAuth *BasicAuth*

Description

Authentication using username and password

Super class

ebx::AbstractAuthentication -> BasicAuth

Public fields

email The email address

password The password

Methods

Public methods:

- BasicAuth$new()
- BasicAuth$has_expired()
- BasicAuth$refresh()
- BasicAuth$get_headers()
- BasicAuth$clone()

Method `new()`**:** Create a new BasicAuth object

Usage:

```
BasicAuth$new(config, email, password)
```

Arguments:

`config` The client configuration

`email` The email address

`password` The password

Returns: A new BasicAuth object

Method `has_expired()`**:** Check if token has expired (always FALSE for basic auth)

Usage:

```
BasicAuth$has_expired()
```

Returns: FALSE

Method `refresh()`**:** Refresh (not supported for basic auth)

Usage:

```
BasicAuth$refresh()
```

Returns: self

Method `get_headers()`**:** Get the headers for HTTP requests

Usage:

```
BasicAuth$get_headers()
```

Returns: A named character vector of headers

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

ClientConfig *ClientConfig*

Description

R6 class for Earth Blox API client configuration

Public fields

`base_url` The base URL for the API

`api_prefix` The version prefix for the API

`oauth_path` The path for the OAuth flow

`persistence_driver` The persistence driver for storing credentials

Methods

Public methods:

- `ClientConfig$new()`
- `ClientConfig$get_api_base_url()`
- `ClientConfig$get_oauth_url()`
- `ClientConfig$get_persistence_driver()`
- `ClientConfig$set_persistence_driver()`
- `ClientConfig$clone()`

Method `new()`**:** Create a new ClientConfig object

Usage:

```
ClientConfig$new(persistence_driver = NULL)
```

Arguments:

`persistence_driver` Optional persistence driver to use for storing credentials. If NULL (default), a `LocalFilePersistence` writing to a sub-directory of `tempdir()` is used. Pass a `MemoryPersistence` instance for hosted environments such as shinyapps.io where file persistence might be undesirable.

Returns: A new ClientConfig object

Method `get_api_base_url()`**:** Get the full API base URL including version prefix

Usage:

```
ClientConfig$get_api_base_url()
```

Returns: The full API base URL

Method `get_oauth_url()`**:** Get the OAuth URL

Usage:

```
ClientConfig$get_oauth_url()
```

Returns: The OAuth URL

Method `get_persistence_driver()`**:** Get the persistence driver

Usage:

```
ClientConfig$get_persistence_driver()
```

Returns: The persistence driver

Method `set_persistence_driver()`**:** Set the persistence driver

Usage:

```
ClientConfig$set_persistence_driver(driver)
```

Arguments:

`driver` A persistence driver object (e.g. `LocalFilePersistence` or `MemoryPersistence`)

Returns: self (invisible)

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`create_oauth_client` *Create OAuth Client*

Description

Register a new OAuth client with the API

Usage

```
create_oauth_client(email, password, name, description = "", scopes = c())
```

Arguments

`email`	The email address of the user
`password`	The password of the user
`name`	The name for the new client
`description`	The description for the new client (optional)
`scopes`	The scopes for the new client (optional)

Value

An OAuthClient object

create_run *Create Run*

Description

Create a new run using the specified project

This function supports two modes: 1. Pass a complete project_spec (list or Project object) with variables 2. Pass individual parameters (project_id + optional substitutions)

Usage

```
create_run(
  project_spec = NULL,
  project_id = NULL,
  start_date = NULL,
  end_date = NULL,
  study_area = NULL,
  include_geometry = FALSE,
  generate_thumbnails = FALSE
)
```

Arguments

project_spec	Optional complete project specification (list or Project object). If provided, this takes precedence over individual parameters.
project_id	The project ID (ignored if project_spec provided)
start_date	Optional start date (datetime or string) - uses deprecated substitutions API
end_date	Optional end date (datetime or string) - uses deprecated substitutions API
study_area	Optional study area - uses deprecated substitutions API
include_geometry	Whether to include geometry in output (default: FALSE)
generate_thumbnails	Whether to generate thumbnails for every layer (default: FALSE)

Value

The run ID as a string

Examples

```
## Not run:
# Method 1: Using project_spec with variables (recommended)
spec <- Project$new(
  id = "project123",
  variables = list(
    list(key = "var_1", type = "area", value = geojson_data)
```

```
  )
)
run_id <- create_run(project_spec = spec)

# Method 2: Using individual parameters with substitutions (deprecated)
run_id <- create_run(
  project_id = "project123",
  start_date = "2024-01-01",
  end_date = "2024-12-31",
  study_area = geojson_data
)

## End(Not run)
```

EbxClient *EbxClient*

Description

R6 class for Earth Blox API client

Public fields

`name` The name of the client

`config` The client configuration

`authenticator` The authentication method

Methods

Public methods:

- `EbxClient$new()`
- `EbxClient$get_headers()`
- `EbxClient$parse_response()`
- `EbxClient$get()`
- `EbxClient$post()`
- `EbxClient$clone()`

Method `new()`**:** Create a new EbxClient object

Usage:

```
EbxClient$new(authenticator = NULL, config = NULL, name = NULL)
```

Arguments:

`authenticator` The authentication method

`config` The client configuration (optional)

`name` The client name (optional)

Returns: A new EbxClient object

Method `get_headers()`**:** Get headers for HTTP requests

Usage:

```
EbxClient$get_headers()
```

Returns: A named character vector of headers

Method `parse_response()`**:** Parse response body based on Content-Type header

Usage:

```
EbxClient$parse_response(response)
```

Arguments:

`response` The httr2 response object

Returns: The parsed response data

Method `get()`**:** Make a GET request to the API

Usage:

```
EbxClient$get(url, query_params = NULL, headers = NULL, timeout = NULL)
```

Arguments:

`url` The endpoint URL

`query_params` Query parameters (optional)

`headers` Additional headers (optional)

`timeout` Request timeout in seconds (optional)

Returns: The parsed response data

Method `post()`**:** Make a POST request to the API

Usage:

```
EbxClient$post(url, payload = NULL, headers = NULL, timeout = NULL)
```

Arguments:

`url` The endpoint URL

`payload` The request payload (optional)

`headers` Additional headers (optional)

`timeout` Request timeout in seconds (optional)

Returns: The parsed response data

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

EnvAuthentication *EnvAuthentication*

Description

Authentication using environment variable token

Super class

`ebx::AbstractAuthentication` -> `EnvAuthentication`

Methods

Public methods:

- `EnvAuthentication$new()`
- `EnvAuthentication$has_expired()`
- `EnvAuthentication$refresh()`
- `EnvAuthentication$get_headers()`
- `EnvAuthentication$clone()`

Method `new()`**:** Create a new EnvAuthentication object

Usage:

```
EnvAuthentication$new(config)
```

Arguments:

`config` The client configuration

Returns: A new EnvAuthentication object

Method `has_expired()`**:** Check if token has expired (always FALSE for env auth)

Usage:

```
EnvAuthentication$has_expired()
```

Returns: FALSE

Method `refresh()`**:** Refresh the token from environment

Usage:

```
EnvAuthentication$refresh()
```

Returns: self

Method `get_headers()`**:** Get the headers for HTTP requests

Usage:

```
EnvAuthentication$get_headers()
```

Returns: A named character vector of headers

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`follow_run` *Follow Run*

Description

Follow a run's progress (polling)

Usage

```
follow_run(run_id, interval = 5, max_attempts = 60)
```

Arguments

`run_id`	The run ID
`interval`	Polling interval in seconds (default: 5)
`max_attempts`	Maximum number of polling attempts (default: 60)

Value

The final Run object

`get_charts` *Get Charts*

Description

Get charts from a run

Usage

```
get_charts(run_id, filter = NULL)
```

Arguments

`run_id`	The run ID
`filter`	Optional filter to apply to chart titles

Value

A list of chart data

get_client *Get or create a client*

Description

Get an existing client or the current default client

Usage

```
get_client(name = NULL)
```

Arguments

name	The client name (optional)

Value

An EbxClient object

get_layers *Get Layers*

Description

Get layers from a run

Usage

```
get_layers(run_id, filter = NULL)
```

Arguments

run_id	The run ID
filter	Optional filter to apply to layer titles

Value

A list of layer data

get_project *Get Project*

Description

Get a specific project by ID

Usage

```
get_project(project_id)
```

Arguments

project_id	The project ID

Value

A Project object

get_run *Get Run*

Description

Get a specific run by ID

Usage

```
get_run(run_id)
```

Arguments

run_id	The run ID

Value

A Run object

get_run_status *Get Run Status*

Description

Get the status of a specific run

Usage

```
get_run_status(run_id)
```

Arguments

run_id	The run ID

Value

The run status as a string

get_tables *Get Tables*

Description

Get tables from a run

Usage

```
get_tables(run_id, filter = NULL)
```

Arguments

run_id	The run ID
filter	Optional filter to apply to table titles

Value

A list of table data

`list_projects` *List Projects*

Description

List all available projects

Usage

```
list_projects()
```

Value

A list of Project objects

`list_runs` *List Runs*

Description

List runs with optional limit

Usage

```
list_runs(limit = 10)
```

Arguments

`limit`	The maximum number of runs to return (default: 10)

Value

A list of Run objects

load_auth_token *Load Auth Token from disk*

Description

Load an auth token from a saved file

Usage

```
load_auth_token(config, filename)
```

Arguments

config	The client config
filename	The filename to load from

Value

An AuthToken object

load_oauth_client *Load OAuth Client from disk*

Description

Load an OAuth client from a saved file

Usage

```
load_oauth_client(config = NULL, filename = API_SECRETS_FILE)
```

Arguments

config	The client config (optional)
filename	The filename to load from (optional)

Value

An OAuthClient object

`LocalFilePersistence` *LocalFilePersistence*

Description

R6 class for persisting credentials to local filesystem

Public fields

`path` The directory path for storing files

Methods

Public methods:

- `LocalFilePersistence$new()`
- `LocalFilePersistence$save()`
- `LocalFilePersistence$load()`
- `LocalFilePersistence$exists()`
- `LocalFilePersistence$clone()`

Method `new()`**:** Create a new LocalFilePersistence object

Usage:

```
LocalFilePersistence$new(path)
```

Arguments:

`path` The directory path for storing files. Must be supplied explicitly; no default is provided to comply with CRAN policy.

Returns: A new LocalFilePersistence object

Method `save()`**:** Save data to a file

Usage:

```
LocalFilePersistence$save(filename, data)
```

Arguments:

`filename` The name of the file

`data` The data to save (will be converted to JSON)

Returns: NULL (invisible)

Method `load()`**:** Load data from a file

Usage:

```
LocalFilePersistence$load(filename)
```

Arguments:

`filename` The name of the file

Returns: The loaded data as a list

Method `exists()`**:** Check if a file exists

Usage:

```
LocalFilePersistence$exists(filename)
```

Arguments:

`filename` The name of the file

Returns: TRUE if the file exists, FALSE otherwise

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`MemoryPersistence` *MemoryPersistence*

Description

R6 class for persisting credentials in memory (e.g. on hosted environments such as shinyapps.io where the local filesystem is not writable). Data is lost when the R session ends.

Public fields

`store` Named list used as an in-memory store

Methods

Public methods:

- `MemoryPersistence$new()`
- `MemoryPersistence$save()`
- `MemoryPersistence$load()`
- `MemoryPersistence$exists()`
- `MemoryPersistence$clone()`

Method `new()`**:** Create a new MemoryPersistence object

Usage:

```
MemoryPersistence$new()
```

Returns: A new MemoryPersistence object

Method `save()`**:** Save data to memory

Usage:

```
MemoryPersistence$save(filename, data)
```

Arguments:

`filename` The key to store data under

`data` The data to save (stored in its native R form)

Returns: NULL (invisible)

Method `load()`**:** Load data from memory

Usage:

```
MemoryPersistence$load(filename)
```

Arguments:

`filename` The key to load data from

Returns: The stored data as a list

Method `exists()`**:** Check if a key exists in memory

Usage:

```
MemoryPersistence$exists(filename)
```

Arguments:

`filename` The key to check

Returns: TRUE if the key exists, FALSE otherwise

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`OAuthAuthentication` *OAuthAuthentication*

Description

Authentication using OAuth client credentials

Super class

`ebx::AbstractAuthentication` -> `OAuthAuthentication`

Public fields

`client_id` The client ID

`client_secret` The client secret

Methods

Public methods:

- OAuthAuthentication$new()
- OAuthAuthentication$get_token_filename()
- OAuthAuthentication$load_saved_credentials()
- OAuthAuthentication$save_credentials()
- OAuthAuthentication$refresh()
- OAuthAuthentication$get_headers()
- OAuthAuthentication$clone()

Method `new()`**:** Create a new OAuthAuthentication object

Usage:

```
OAuthAuthentication$new(config, client_id = NULL, client_secret = NULL)
```

Arguments:

`config` The client configuration

`client_id` The client ID

`client_secret` The client secret

Returns: A new OAuthAuthentication object

Method `get_token_filename()`**:** Get the token filename

Usage:

```
OAuthAuthentication$get_token_filename()
```

Returns: The filename for storing this client's token

Method `load_saved_credentials()`**:** Load saved credentials from disk

Usage:

```
OAuthAuthentication$load_saved_credentials()
```

Returns: self

Method `save_credentials()`**:** Save credentials to disk

Usage:

```
OAuthAuthentication$save_credentials()
```

Returns: self

Method `refresh()`**:** Refresh the OAuth token

Usage:

```
OAuthAuthentication$refresh()
```

Returns: self

Method `get_headers()`**:** Get the headers for HTTP requests

Usage:

```
OAuthAuthentication$get_headers()
```

Returns: A named character vector of headers

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`OAuthClient` *OAuthClient*

Description

R6 class for OAuth client credentials

Public fields

`name` The name of the client

`description` The description of the client

`client_id` The client ID

`client_secret` The client secret

`enabled` Whether the client is enabled

Methods

Public methods:

- `OAuthClient$new()`
- `OAuthClient$save()`
- `OAuthClient$to_list()`
- `OAuthClient$to_json()`
- `OAuthClient$clone()`

Method `new()`**:** Create a new OAuthClient object

Usage:

```
OAuthClient$new(
  name = NULL,
  description = NULL,
  client_id = NULL,
  client_secret = NULL,
  enabled = NULL
)
```

Arguments:

`name` The name of the client

`description` The description of the client

`client_id` The client ID

`client_secret` The client secret

`enabled` Whether the client is enabled

Returns: A new OAuthClient object

Method `save()`**:** Save the OAuth client to disk

Usage:

```
OAuthClient$save(config = NULL, filename = API_SECRETS_FILE)
```

Arguments:

`config` The client config (optional)

`filename` The filename to save to (optional)

Returns: self (invisible)

Method `to_list()`**:** Convert OAuthClient to a plain list

Usage:

```
OAuthClient$to_list(include_secret = FALSE)
```

Arguments:

`include_secret` Whether to include client_secret (default: FALSE for security)

Returns: A list containing OAuth client fields

Method `to_json()`**:** Convert OAuthClient to JSON string

Usage:

```
OAuthClient$to_json(pretty = TRUE, include_secret = FALSE, ...)
```

Arguments:

`pretty` Whether to pretty-print the JSON (default: TRUE)

`include_secret` Whether to include client_secret (default: FALSE for security)

`...` Additional arguments passed to jsonlite::toJSON

Returns: A JSON string representation of the OAuth client

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

oauth_credentials_exist *Check if saved OAuth credentials exist*

Description

Check if a credentials file exists

Usage

```
oauth_credentials_exist(config = NULL, filename = API_SECRETS_FILE)
```

Arguments

config	The client config (optional)
filename	The filename to check (optional)

Value

TRUE if the file exists, FALSE otherwise

Project *Project*

Description

R6 class representing a project

Public fields

`id` The project ID

`name` The project name (mapped from API 'title' field)

`description` The project description

`version` The project version

`api_version` The API version

`api_access` Whether API access is enabled

`variables` List of project variables

`exec_parameters` Execution parameters configuration

Methods

Public methods:

- Project$new()
- Project$print()
- Project$to_list()
- Project$to_json()
- Project$clone()

Method `new()`**:** Create a new Project object

Usage:

```
Project$new(
  id = NULL,
  name = NULL,
  description = NULL,
  version = NULL,
  api_version = NULL,
  api_access = NULL,
  variables = NULL,
  exec_parameters = NULL,
  title = NULL,
  ...
)
```

Arguments:

`id` The project ID

`name` The project name

`description` The project description

`version` The project version

`api_version` The API version

`api_access` Whether API access is enabled

`variables` List of project variables

`exec_parameters` Execution parameters

`title` Alternative name for 'name' field (API compatibility)

`...` Additional fields (ignored)

Returns: A new Project object

Method `print()`**:** Print method for Project

Usage:

```
Project$print(...)
```

Arguments:

`...` Additional arguments (ignored)

Method `to_list()`**:** Convert Project to a plain list

Usage:

```
Project$to_list()
```

Returns: A list containing all project fields

Method `to_json()`**:** Convert Project to JSON string

Usage:

```
Project$to_json(pretty = TRUE, ...)
```

Arguments:

`pretty` Whether to pretty-print the JSON (default: TRUE)

`...` Additional arguments passed to jsonlite::toJSON

Returns: A JSON string representation of the project

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Run` *Run*

Description

R6 class representing a run

Public fields

`id` The run ID

`project_id` The associated project ID

`status` The run status

`started_at` The run start timestamp

`completed_at` The run completion timestamp

`exec_parameters` The execution parameters

`layers` The layers associated with the run

`outputs` The outputs (charts/tables) associated with the run

`name` The run name (optional)

Methods

Public methods:

- Run$new()
- Run$print()
- Run$to_list()
- Run$to_json()
- Run$clone()

Method `new()`**:** Create a new Run object

Usage:

```
Run$new(
  id = NULL,
  project_id = NULL,
  status = NULL,
  started_at = NULL,
  completed_at = NULL,
  exec_parameters = NULL,
  layers = NULL,
  outputs = NULL,
  name = NULL,
  ...
)
```

Arguments:

`id` The run ID

`project_id` The project ID

`status` The run status

`started_at` The run start timestamp

`completed_at` The run completion timestamp

`exec_parameters` The execution parameters

`layers` The layers list

`outputs` The outputs list

`name` The run name (optional)

`...` Additional fields (ignored)

Returns: A new Run object

Method `print()`**:** Print method for Run

Usage:

```
Run$print(...)
```

Arguments:

`...` Additional arguments (ignored)

Method `to_list()`**:** Convert Run to a plain list

Usage:

```
Run$to_list()
```

Returns: A list containing all run fields

Method `to_json()`**:** Convert Run to JSON string

Usage:

```
Run$to_json(pretty = TRUE, ...)
```

Arguments:

`pretty` Whether to pretty-print the JSON (default: TRUE)

`...` Additional arguments passed to jsonlite::toJSON

Returns: A JSON string representation of the run

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ServiceClientConfig` *ServiceClientConfig*

Description

R6 class for Earth Blox API client configuration for service endpoints

Super class

`ebx::ClientConfig` -> `ServiceClientConfig`

Methods

Public methods:

- `ServiceClientConfig$new()`
- `ServiceClientConfig$clone()`

Method `new()`**:** Create a new ServiceClientConfig object

Usage:

```
ServiceClientConfig$new(persistence_driver = NULL)
```

Arguments:

`persistence_driver` Optional persistence driver (see `ClientConfig`).

Returns: A new ServiceClientConfig object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Index