Package: ctoclient (via r-universe)

May 13, 2026

Type Package

Title A Modern and Flexible Data Pipeline for 'SurveyCTO'

Version 0.1.0

Date 2026-03-28

Description A modern and flexible R client for the 'SurveyCTO', a mobile and offline data collection platform, providing a modern and consistent interface for programmatic access to server resources. Built on top of the 'httr2' package, it enables secure and efficient data retrieval and returns analysis-ready data through optional tidying. It includes functions to create, upload, and download server datasets, in addition to fetching form data, files, and submission attachments. Robust authentication and request handling make the package suitable for automated survey monitoring and downstream analysis.

License MIT + file LICENSE

Language en-US

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports checkmate, cli, dplyr, httr2, purrr, readr, readxl, rlang, stringr, tidyr

Suggests curl, httptest2, jsonlite, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://guturago.github.io/ctoclient/, https://github.com/guturago/ctoclient/

BugReports https://github.com/guturago/ctoclient/issues

Depends R (>= 4.1.0)

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev libssl-dev libx11-dev

Repository https://guturago.r-universe.dev

Date/Publication 2026-05-13 15:53:02 UTC

RemoteUrl https://github.com/guturago/ctoclient

RemoteRef HEAD

RemoteSha 1f72f2758a1677871f39990eaf7c44a4bee17815

Contents

`cto_connect` *Connect to and manage a SurveyCTO Server connection*

Description

- `cto_connect()` authenticates against a SurveyCTO server, verifies credentials, and handles cookies.
- `cto_set_connection()` manually sets or restores an existing session object.
- `cto_is_connected()` checks if an active session currently exists in the internal environment.

Usage

```
cto_connect(server, username, password = NULL, cookies = TRUE)

cto_set_connection(session)

cto_is_connected()
```

Arguments

server	String. The subdomain of your SurveyCTO server. For example, if the full URL is https://my-org.surveycto.com, set this to "my-org".
username	String. The username or email address associated with the account.
password	String. The user password. If left NULL (recommended), it prompts you for the password interactively.
cookies	Logical. If TRUE (default), the client preserves cookies across requests and handles CSRF tokens automatically. This is required for maintaining stateful sessions to access endpoints that not available through the REST API.
session	A cto_session object previously created by cto_connect().

Details

Session Management:

By default, this package operates **statefully** by preserving cookies if the connection is established with the cookies = TRUE argument. Upon successful authentication, the request object (.session) is assigned to an internal package environment (.ctoclient_env). Therefore, you do not need to pass a request object to other functions in this package; they will automatically use the active session. If you are working with multiple servers, please use cto_set_connection() to switch the server connection.

Security Best Practices:

It is highly recommended to avoid hard-coding passwords in your scripts.

- **Interactive Session:** Pass the password securely via console input (leave it NULL) or keychain management tools.
- **Automation/Scripts:** Store your credentials in your .Renviron file (e.g., SCTO_PASSWORD) and retrieve them with Sys.getenv().

Value

- cto_connect(): The session object (invisibly).
- cto_set_connection(): NULL (invisibly), called for its side effect of setting the session.
- cto_is_connected(): A logical TRUE or FALSE.

See Also

httr2::req_auth_basic(), usethis::edit_r_environ()

Examples

```
## Not run:
# 1. Standard authentication
cto_connect("my-org", "user@org.com", Sys.getenv("SCTO_PASSWORD"))

# 2. Check if connected
cto_is_connected()
```

```
# 3. Recommended for interactive use
con <- cto_connect("my-org", "user@org.com")

# 4. Restore and existing connection
cto_set_connection(con)

## End(Not run)
```

`cto_dataset_create` *Create or Upload to Server Datasets*

Description

These functions manage the lifecycle of SurveyCTO server datasets: creating the container definition and populating it with data.

- `cto_dataset_create()`: Creates a new dataset with the specified configuration.
- `cto_dataset_upload()`: Uploads records from a CSV file to the specified dataset. Supports:
  - APPEND: add new records
  - MERGE: update existing records based on unique field
  - CLEAR: replace all data

Usage

```
cto_dataset_create(
  id,
  title = id,
  discriminator = NULL,
  unique_record_field = NULL,
  allow_offline_updates = NULL,
 id_format_options = list(prefix = NULL, allowCapitalLetters = NULL, suffix = NULL,
    numberOfDigits = NULL),
 cases_management_options = list(otherUserCode = NULL, showFinalizedSentWhenTree = NULL,
   enumeratorDatasetId = NULL, showColumnsWhenTable = NULL, displayMode = NULL,
    entryMode = NULL),
 location_context = list(parentGroupId = 1, siblingBelow = list(itemClass = NULL, id =
    NULL), siblingAbove = list(itemClass = NULL, id = NULL))
)

cto_dataset_upload(
  id,
  file,
  upload_mode = c("APPEND", "MERGE", "CLEAR"),
  joining_field = NULL
)
```

Arguments

id	String. The unique identifier for the dataset (e.g., "household_data").
title	String. The display title of the dataset. Defaults to id.
discriminator	String. The type of dataset to create.
unique_record_field	String. The name of the field that uniquely identifies records. Required if upload_mode is "merge".
allow_offline_updates	Logical. Whether the dataset allows updates while offline.
id_format_options	List. Options for formatting IDs within the dataset.
cases_management_options	List. Specific configurations for case management
location_context	List. Metadata regarding where the dataset resides.
file	String. Path to the local CSV file to upload.
upload_mode	String. How the data should be handled.
joining_field	String. The column name used to match records during a "merge". Often the same as unique_record_field.

Value

A list containing the API response (metadata for creation, or job summary for upload).

See Also

Other Dataset Management Functions: cto_dataset_delete(), cto_dataset_download(), cto_dataset_info(), cto_dataset_list()

Examples

```
## Not run:
# 1. Create the container
cto_dataset_create(
id = "hh_data",
title = "Household Data",
unique_record_field = "hh_id"
)

# 2. Upload data to it
cto_dataset_upload(
file = "data.csv",
id = "hh_data",
upload_mode = "merge",
joining_field = "hh_id"
)

## End(Not run)
```

`cto_dataset_delete` *Delete or Purge a Dataset*

Description

Functions to permanently remove data from the server.

- `cto_delete_dataset()`: **Permanently deletes** a dataset and all its associated data. This operation cannot be undone
- `cto_purge_dataset()`: Removes **all records** from the dataset but keeps the dataset definition (schema/ID) intact.

Usage

```
cto_dataset_delete(id)

cto_dataset_purge(id)
```

Arguments

`id`	String. The unique identifier of the dataset.

Value

A list confirming the operation status.

See Also

Other Dataset Management Functions: `cto_dataset_create()`, `cto_dataset_download()`, `cto_dataset_info()`, `cto_dataset_list()`

Examples

```
## Not run:
# 1. Delete dataset
cto_dataset_delete(id = "hh_data")

# 2. Purge dataset
cto_dataset_purge(id = "hh_data")

## End(Not run)
```

`cto_dataset_download` *Download SurveyCTO Server Datasets*

Description

Downloads one or more datasets from a SurveyCTO server to a local directory as CSV files.

Usage

```
cto_dataset_download(id = NULL, dir = getwd(), overwrite = FALSE)
```

Arguments

`id`	A character vector of dataset IDs to download. If `NULL` (the default), the function queries the server for a list of all available datasets and downloads them all.
`dir`	A string specifying the directory where CSV files will be saved. Defaults to the current working directory.
`overwrite`	Logical. If `TRUE`, existing files in `dir` will be overwritten. If `FALSE` (the default), existing files are skipped to conserve bandwidth.

Details

- **Smart Downloading:** If `overwrite = FALSE`, the function checks if the target file already exists in `dir` and skips the download.
- **Error Handling:** If a specific dataset fails to download, a warning is printed with the dataset name, but the function continues processing the remaining list.

Value

(Invisibly) A character vector of file paths to the successfully downloaded CSVs. Returns `NULL` if no datasets were found.

See Also

Other Dataset Management Functions: `cto_dataset_create()`, `cto_dataset_delete()`, `cto_dataset_info()`, `cto_dataset_list()`

Examples

```
## Not run:
# --- Example 1: Download a specific dataset ---
paths <- cto_dataset_download(id = "household_data", dir = tempdir())
df <- read.csv(paths[1])

# --- Example 2: Download all datasets, skip existing files ---
paths <- cto_dataset_download(dir = "my_data_folder", overwrite = FALSE)

## End(Not run)
```

cto_dataset_info *Get Dataset Properties*

Description

Retrieves detailed metadata for a specific dataset, including its configuration, schema, and status.

Usage

```
cto_dataset_info(id)
```

Arguments

id	String. The unique identifier of the dataset.

Value

A list containing the dataset properties.

See Also

Other Dataset Management Functions: `cto_dataset_create()`, `cto_dataset_delete()`, `cto_dataset_download()`, `cto_dataset_list()`

Examples

```
## Not run:
ds_info <- cto_dataset_info(id = "hh_data")

## End(Not run)
```

cto_dataset_list *List Available Server Datasets*

Description

Retrieves a list of datasets that the authenticated user has access to. Results can be filtered by team and ordered by specified fields.

Usage

```
cto_dataset_list(
  order_by = "createdOn",
  sort = c("ASC", "DESC"),
  team_id = NULL
)
```

Arguments

order_by	String. The field to sort the results by. Options are: "id", "title", "createdOn", "modifiedOn", "status", "version", or "discriminator". Defaults to "createdOn".
sort	String. The direction of the sort: "asc" (ascending) or "desc" (descending). Defaults to "asc".
team_id	String (Optional). Filter datasets by a specific Team ID. If provided, only datasets accessible to that team are returned. Example: 'team-456'.

Value

A data frame containing the metadata of available datasets.

See Also

Other Dataset Management Functions: cto_dataset_create(), cto_dataset_delete(), cto_dataset_download(), cto_dataset_info()

Examples

```
## Not run:
# List all datasets sorted by creation date
ds_list <- cto_dataset_list()

# List datasets for a specific team, ordered by title
team_ds <- cto_dataset_list(team_id = "team-123", order_by = "title")

## End(Not run)
```

---

cto_form_attachment *Download Attachments from a SurveyCTO Form*

Description

Downloads files attached to a deployed SurveyCTO form, such as preloaded CSV files, media assets, or other server-side attachments.

Usage

```
cto_form_attachment(form_id, filename = NULL, dir = getwd(), overwrite = FALSE)
```

Arguments

form_id	A string specifying the SurveyCTO form ID.
filename	Optional character vector of specific filenames to download (e.g., "prices.csv"). If NULL (default), all available attachments associated with the form are downloaded.
dir	A string giving the directory where files will be saved. Defaults to getwd().
overwrite	Logical; if TRUE, existing files in dir will be overwritten. If FALSE (the default), existing files are skipped.

Details

This function first calls `cto_form_metadata()` to retrieve metadata for the deployed form, including the list of available attachments.

- **File types**: Any attached file type can be downloaded (for example, images, audio, or CSV files).
- **Progress reporting**: When `options(scto.verbose = TRUE)` (default), progress messages are printed using the CLI framework.
- **Caching**: Files are not re-downloaded if they already exist in `dir` unless `overwrite = TRUE`.

If all requested files are not available, the function aborts with an informative message suggesting how to inspect the form metadata.

Value

A character vector of file paths to all available attachments that exist locally after the function completes (invisibly).

Returns `invisible(NULL)` if the form has no attachments.

See Also

Other Form Management Functions: `cto_form_data()`, `cto_form_data_attachment()`, `cto_form_dofile()`, `cto_form_languages()`, `cto_form_metadata()`

Examples

```
## Not run:
files <- cto_form_attachment("household_survey")

# 2. Download specific files to a local directory
cto_form_attachment(
  form_id  = "household_survey",
  filename = c("item_list.csv", "logo.png"),
  dir      = "data/raw"
)

# 3. Force re-download of a file
p <- cto_form_attachment(
  form_id  = "household_survey",
  filename = "prices.csv",
  overwrite = TRUE
)

prices <- read.csv(p)

## End(Not run)
```

`cto_form_data` *Download and Tidy SurveyCTO Form Data*

Description

Downloads submission data from a SurveyCTO server in wide JSON format. Encrypted forms are supported via a private key. When `tidy = TRUE` (default), the function uses the form's XLSForm definition to convert variables to appropriate R types, drop structural fields, and organize columns for analysis.

Usage

```
cto_form_data(
  form_id,
  private_key = NULL,
  start_date = as.POSIXct("2000-01-01"),
  status = c("approved", "rejected", "pending"),
  tidy = TRUE
)
```

Arguments

`form_id`	A string specifying the SurveyCTO form ID.
`private_key`	An optional path to a `.pem` private key file. Required if the form is encrypted.
`start_date`	A POSIXct timestamp. Only submissions received after this date/time are requested. Defaults to `"2000-01-01"`.
`status`	A character vector of submission statuses to include. Must be a subset of `"approved"`, `"rejected"`, and `"pending"`. Defaults to all three.
`tidy`	Logical; if `TRUE`, attempts to clean and restructure the raw SurveyCTO output using the XLSForm definition.

Details

When `tidy = TRUE`, the function performs several common post-processing steps:

- **Type conversion:** Converts numeric, date, and datetime fields to native R types based on question types in the XLSForm.
- **Structural cleanup:** Removes layout-only fields such as notes, group markers, and repeat delimiters.
- **Column ordering:** Places key submission metadata (for example, completion and submission dates) first, followed by survey variables in form order.
- **Media fields:** Strips URLs from image, audio, and video fields, leaving only the filename.
- **Geopoints:** Splits geopoint variables into four columns with `_latitude`, `_longitude`, `_altitude`, and `_accuracy` suffixes when not already present.

Value

A `data.frame` containing the downloaded submissions.

If `tidy = FALSE`, the raw parsed JSON response is returned. If `tidy = TRUE`, a cleaned version with standardized column types and ordering is returned.

Returns an empty `data.frame` when no submissions are available.

See Also

Other Form Management Functions: `cto_form_attachment()`, `cto_form_data_attachment()`, `cto_form_dofile()`, `cto_form_languages()`, `cto_form_metadata()`

Examples

```
## Not run:
# Download raw submissions
raw <- cto_form_data("my_form_id", tidy = FALSE)

# Download and tidy encrypted data
clean <- cto_form_data("my_form_id", private_key = "keys/my_key.pem")

## End(Not run)
```

---

cto_form_data_attachment

Download Attachments from SurveyCTO Form Data

Description

Extracts attachment URLs (images, audio, video, signatures) from SurveyCTO form data and downloads the files to a local directory. This function handles encrypted forms if a private key is provided.

Usage

```
cto_form_data_attachment(
  form_id,
  fields = everything(),
  private_key = NULL,
  dir = file.path(getwd(), "media"),
  overwrite = FALSE
)
```

Arguments

`form_id`	A string specifying the SurveyCTO form ID to inspect.
`fields`	A `tidy-select` expression (e.g., `everything()`, `starts_with("img_")`) specifying which columns should be scanned for attachment URLs. Defaults to `everything()`.

private_key	Optional. A character string specifying the path to a local RSA private key file. Required if the form is encrypted.
dir	A character string specifying the local directory where files should be saved. Defaults to "media". The directory must exist.
overwrite	Logical. If TRUE, existing files with the same name in dir will be overwritten. If FALSE (the default), existing files are skipped.

Details

This function performs the following steps:

1. Fetches the form data using cto_form_data.
2. Scans the selected fields for values matching the standard SurveyCTO API attachment URL pattern.
3. Downloads the identified files sequentially to the specified dir.

Value

Returns a vector of file paths (invisibly). The function is called for its side effect of downloading files to the local disk.

See Also

Other Form Management Functions: cto_form_attachment(), cto_form_data(), cto_form_dofile(), cto_form_languages(), cto_form_metadata()

Examples

```
## Not run:
# 1. Download all attachments from the form submissions
cto_form_data_attachment(
  form_id = "household_survey_v1",
  dir = "downloads/medias"
)

# 2. Download only specific image fields from an encrypted form
cto_form_data_attachment(
  form_id = "encrypted_health_survey",
  fields = starts_with("image_"),
  private_key = "keys/my_priv_key.pem",
  overwrite = TRUE
)

## End(Not run)
```

`cto_form_dofile` *Generate a Stata Do-File with Variable and Value Labels from a SurveyCTO Form*

Description

Creates a Stata `.do` file that applies variable labels, value labels, and notes to a dataset based on the XLSForm definition of a SurveyCTO form. The function supports multi-language forms, repeat groups, and `select_multiple` questions, and generates Stata-compatible regular expressions so labels are applied to all indexed variables.

Usage

```
cto_form_dofile(form_id, path = NULL)
```

Arguments

`form_id`	A character string specifying the SurveyCTO form ID.
`path`	Optional character string giving the output file path for the generated `.do` file. Must end in `.do`. If `NULL`, the file is not written to disk and the generated commands are returned invisibly.

Details

The function performs several processing steps:

- **Language selection:** Automatically chooses the default language defined in the XLSForm, or falls back to English when multiple label columns are present.
- **Value labels:** Generates Stata `label define` commands for all `select_one` choice lists and a binary label set for `select_multiple` variables.
- **Repeat handling:** For variables inside repeat groups, Stata loops and regex matching are created so labels apply to all indexed copies (for example, `child_age_1`, `child_age_2`).
- **Select-multiple expansion:** Produces conditional labeling logic for binary indicator variables derived from `select_multiple` questions.
- **Label cleaning:** Removes HTML markup, escapes Stata-special characters, normalizes whitespace, and preserves SurveyCTO interpolation strings such as `${var}`.

Value

A character vector containing the lines of the generated Stata `.do` file. The value is returned invisibly.

See Also

Other Form Management Functions: `cto_form_attachment()`, `cto_form_data()`, `cto_form_data_attachment()`, `cto_form_languages()`, `cto_form_metadata()`

Examples

```
## Not run:
# Generate a Stata do-file and write it to disk
cto_form_dofile("household_survey", path = "labels.do")

# Generate without writing to a file
cmds <- cto_form_dofile("household_survey")

## End(Not run)
```

`cto_form_languages` *Download SurveyCTO Form Files and Templates*

Description

These functions retrieve auxiliary files and templates associated with a deployed SurveyCTO form. All these functions require a stateful session to work.

- `cto_form_languages()` retrieves the list of languages defined in the form.
- `cto_form_stata_template()` downloads a Stata `.do` file template for importing submitted data.
- `cto_form_printable()` downloads a printable (HTML) version of the form definition.
- `cto_form_mail_template()` downloads a mail merge template for the form.

All downloads are saved locally and their file paths are returned invisibly.

Usage

```
cto_form_languages(form_id)

cto_form_stata_template(
  form_id,
  dir = getwd(),
  lang = NULL,
  csv_dir = NULL,
  dta_dir = NULL
)

cto_form_printable(
  form_id,
  dir = getwd(),
  lang = NULL,
  relevancies = FALSE,
  constraints = FALSE
)

cto_form_mail_template(form_id, dir = getwd(), type = 2, group_names = FALSE)
```

Arguments

form_id	A character string giving the unique SurveyCTO form ID.
dir	A character string specifying the directory where downloaded files will be saved. Defaults to the current working directory.
lang	Optional character string giving the language identifier (for example, "English"). If NULL, the form's default language is used.
csv_dir	Optional character string giving the directory where the CSV dataset will eventually be stored. This value is embedded in the generated Stata .do file to automate data loading.
dta_dir	Optional character string giving the directory where the Stata .dta file should be written by the template.
relevancies	Logical; if TRUE, relevance logic (skip patterns) is included in the printable form. Defaults to FALSE.
constraints	Logical; if TRUE, constraint logic is included in the printable form. Defaults to FALSE.
type	Integer (0–2) specifying the format of the mail merge template: • 0: Field names only. • 1: Field labels only. • 2: Both field names and labels.
group_names	Logical; if TRUE, group names are included in variable headers. Defaults to FALSE.

Value

- cto_form_languages() returns a list containing the available languages and the index of the default language (1-based).
- All other functions return the local file path of the downloaded file, invisibly.

See Also

Other Form Management Functions: cto_form_attachment(), cto_form_data(), cto_form_data_attachment(), cto_form_dofile(), cto_form_metadata()

Examples

```
## Not run:
form <- "household_survey"

# 1. List available form languages
langs <- cto_form_languages(form)
print(langs)

# 2. Download a Stata import template
# Provide future CSV/DTA locations so the .do file is ready to run
cto_form_stata_template(
  form_id = form,
```

```
  dir     = "downloads/",
  csv_dir = "C:/Data",
  dta_dir = "C:/Data"
)

# 3. Download a printable form with logic displayed
cto_form_printable(
  form_id      = form,
  dir          = "documentation/",
  relevancies  = TRUE,
  constraints  = TRUE
)

# 4. Download a mail-merge template
cto_form_mail_template(
  form_id = form,
  dir     = "templates/",
  type    = 2
)

## End(Not run)
```

`cto_form_metadata` *Download SurveyCTO Form Metadata and Definitions*

Description

Functions for interacting with SurveyCTO form definitions.

- `cto_form_metadata()` retrieves raw metadata for a form, including available definition files, version identifiers, and download URLs.
- `cto_form_definition()` downloads a specific XLSForm definition (Excel file) to a local directory.

Usage

```
cto_form_metadata(form_id)

cto_form_definition(form_id, version = NULL, dir = getwd(), overwrite = FALSE)
```

Arguments

`form_id`	A string giving the unique SurveyCTO form ID.
`version`	Optional string specifying a particular form version to download. If `NULL` (default), the currently deployed version is used.
`dir`	Directory where the XLSForm should be saved. Defaults to `getwd()`.
`overwrite`	Logical; if `TRUE`, an existing file in `dir` will be overwritten. If `FALSE` (default), the existing file is used.

Details

- **Version Handling:** When `version` is supplied, it is validated against the available versions from `cto_form_metadata()`. An informative error is raised if the requested version does not exist.
- **Caching:** If the file already exists in `dir`, it will not be re-downloaded unless `overwrite = TRUE`.

Value

- `cto_form_metadata()` returns a list containing the metadata, including keys for `deployedGroupFiles` and `previousDefinitionFiles`.
- `cto_form_definition()` returns a character string with the path to the downloaded Excel file.

See Also

Other Form Management Functions: `cto_form_attachment()`, `cto_form_data()`, `cto_form_data_attachment()`, `cto_form_dofile()`, `cto_form_languages()`

Examples

```
## Not run:
# --- 1. Get raw metadata ---
meta <- cto_form_metadata("household_survey")

# --- 2. Download the current form definition ---
file_path <- cto_form_definition("household_survey")

# --- 3. Download a specific historical version ---
file_path_v <- cto_form_definition(
  "household_survey",
  version = "20231001"
)

# --- 4. Read XLSForm manually with readxl ---
library(readxl)
survey <- read_excel(file_path, sheet = "survey")
choices <- read_excel(file_path, sheet = "choices")
settings <- read_excel(file_path, sheet = "settings")

## End(Not run)
```

`cto_metadata` *Retrieve Server Metadata and Resource Lists*

Description

These functions retrieve various metadata and lists of resources (forms, groups, teams, roles, users) from the SurveyCTO server.

- `cto_metadata()`: Retrieves a combined structure of forms, groups, and datasets (legacy console endpoint).
- `cto_form_ids()`: Returns a simple vector of all form IDs.
- `cto_group_list()`: Lists all form groups.
- `cto_team_list()`: Lists all available team IDs.
- `cto_role_list()`: Lists all defined user roles.
- `cto_user_list()`: Lists all users on the server.

Usage

```
cto_form_ids()

cto_metadata(which = c("all", "datasets", "forms", "groups"))

cto_group_list(
  order_by = c("createdOn", "id", "title"),
  sort = c("ASC", "DESC"),
  parent_group_id = NULL
)

cto_team_list()

cto_role_list(
  order_by = c("createdOn", "id", "title", "createdBy"),
  sort = c("ASC", "DESC")
)

cto_user_list(
  order_by = c("createdOn", "username", "roleId", "modifiedOn"),
  sort = c("ASC", "DESC"),
  role_id = NULL
)
```

Arguments

`which`	String. Specifies which subset of metadata to return for `cto_metadata()`. One of: • `"all"` (default): Returns a list containing groups, datasets, and forms. • `"groups"`: Returns a data frame of form groups. • `"datasets"`: Returns a data frame of server datasets. • `"forms"`: Returns a data frame of deployed forms.
`order_by`	String. Field to sort the results by. Available fields vary by function (e.g., `"createdOn"`, `"id"`, `"title"`, or `"username"`).

sort	String. Sort direction: "ASC" (ascending) or "DESC" (descending).
parent_group_id	Number (Optional). Filter groups by their parent group ID.
role_id	String (Optional). Filter users by a specific Role ID.

Value

The return value depends on the function:

- cto_form_ids() and cto_team_list() return a **character vector** of IDs.
- cto_metadata() returns a **list** (if which = "all") or a **data frame**.
- cto_group_list(), cto_role_list(), and cto_user_list() return a **list** or **data frame** of the requested resources (depending on pagination handling).

Examples

```
## Not run:
# --- 1. Basic Metadata ---
# Get all form IDs as a vector
ids <- cto_form_ids()

# Get detailed metadata about forms
meta_forms <- cto_metadata("forms")

# --- 2. Resource Lists ---
# List all groups, sorted by title
groups <- cto_group_list(order_by = "title", sort = "asc")

# List all users with a specific role
admins <- cto_user_list(role_id = "admin_role_id")

## End(Not run)
```

Index