Package 'zentraR'

August 7, 2026

Title R Client for the ZENTRA Cloud V5 API

Version 0.1.3

Description Downloads environmental sensor data from the ZENTRA Cloud V5 API (<https://api.zentracloud.io>) into tidy data frames. Provides device discovery, reading retrieval with automatic pagination and rate-limit handling, tidy long output with a wide-format helper, and an incremental 'sync' engine with pluggable local storage (RDS files, CSV files, or return-only) so that new readings can be fetched on a schedule and appended to a growing local record.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports cli, httr2 (>= 1.0.0), rlang (>= 1.0.0), tibble, tidyr, vctrs

Suggests jsonlite, knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

URL https://gitlab.com/meter-group-inc/pubpackages/zentraR

BugReports https://gitlab.com/meter-group-inc/pubpackages/zentraR/-/work_items

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Travis Bates [aut, cre],
METER Group, Inc. [cph, fnd]

Maintainer Travis Bates <travis@metergroup.com>

Repository CRAN

Date/Publication 2026-08-07 17:20:02 UTC

Contents

`zc_error_codes` *ZENTRA device and sensor error codes*

Description

Returns the lookup table of `error_code` values that can appear on readings. A code of `0` means the reading is valid; any non-zero code flags a device or sensor problem.

Usage

```
zc_error_codes()
```

Value

A tibble with columns `error_code` and `error_label`.

Examples

```
zc_error_codes()
```

`zc_get_readings` *Download sensor readings for a device*

Description

Fetches time-series readings for a single device and returns them in a tidy "long" data frame: one row per (port, measurement, timestamp). All pages are retrieved automatically, and the request is paced to respect the API rate limit.

Usage

```
zc_get_readings(
  device_id,
  start = NULL,
  end = NULL,
  start_timestamp = NULL,
  end_timestamp = NULL,
  direction = "ascending",
  units = "metric",
  max_pages = Inf,
  progress = NULL,
  key = NULL
)
```

Arguments

`device_id`	Device identifier (string), e.g. `"z6-00930"` or `"A4100110"`. See `zc_list_devices()`.
`start, end`	Start/end of the window. Accepts a Date, POSIXct, numeric epoch seconds, or an ISO 8601 string. Interpreted as UTC when no timezone is present.
`start_timestamp, end_timestamp`	Alternative window as Unix epoch seconds (UTC). Mutually exclusive with `start`/`end`.
`direction`	Sort order by time: `"ascending"` (default) or `"descending"`.
`units`	`"metric"` (default) or `"imperial"`.
`max_pages`	Maximum number of pages (calendar-month windows) to fetch. Defaults to `Inf` (all available). Useful to cap very large backfills.
`progress`	Show a live progress indicator while pages are fetched? `NULL` (default) shows one in interactive sessions; set `TRUE`/`FALSE` to force it, or set `options(zentraR.progress = FALSE)` to turn it off globally. During a rate-limit pause the request waits and the underlying HTTP client reports the wait.
`key`	Optional API key. Defaults to the key set with `zc_set_key()` or the `ZENTRACLOUD_API_KEY` environment variable.

Details

Specify the time window with `start`/`end` (any of a Date, POSIXct, numeric epoch seconds, or an ISO 8601 string) **or** with `start_timestamp`/ `end_timestamp` (epoch seconds) - but not both. All datetimes are normalised to UTC. If you omit the window, the API returns its default (most recent) window.

Value

A tibble with columns `device_id`, `datetime` (POSIXct, UTC), `timestamp` (numeric epoch seconds), `port_num`, `sensor_name`, `measurement`, `value`, `unit`, and `error_code` (`0` = valid; see `zc_error_codes()`).

Examples

```
## Not run:
zc_set_key("my-secret-key")

# Last 7 days:
zc_get_readings("z6-00930", start = Sys.Date() - 7)

# An explicit window:
zc_get_readings(
  "z6-00930",
  start = as.POSIXct("2026-05-01", tz = "UTC"),
  end   = as.POSIXct("2026-06-01", tz = "UTC")
)

## End(Not run)
```

---

zc_has_key *Check whether an API key is available*

Description

Check whether an API key is available

Usage

```
zc_has_key()
```

Value

`TRUE` if a key has been set with `zc_set_key()` or is present in the `ZENTRACLOUD_API_KEY` environment variable, otherwise `FALSE`.

zc_label_errors *Add human-readable error labels to readings*

Description

Joins the zc_error_codes() table onto a readings data frame, adding an error_label column so quality flags are easy to read and filter.

Usage

```
zc_label_errors(readings)
```

Arguments

readings	A data frame from zc_get_readings() with an error_code column.

Value

readings with an added error_label character column. Unknown codes get an "Unknown error code (<n>)" label.

Examples

```
## Not run:
r <- zc_get_readings("z6-00930", start = Sys.Date() - 2)
zc_label_errors(r)

## End(Not run)
```

zc_list_devices *List the devices your API key can access*

Description

Returns one row per device the authenticated key is allowed to see. Use this to discover device ids before calling zc_get_readings() or zc_sync(). All pages are fetched automatically.

Usage

```
zc_list_devices(
  organization_id = NULL,
  expand = NULL,
  limit = 1000L,
  max_pages = Inf,
  key = NULL
)
```

Arguments

organization_id	Optional organization id (UUID string) to restrict the list to a single organization.
expand	Optional character vector of extra detail to attach to each device. Any of "max_min_timestamp" (first/last measurement times), "settings", "hardware", "connectivity", "subscription". Adds extra columns to the result.
limit	Devices requested per API page (1-1000, default 1000). Affects only the number of underlying requests, not the returned data.
max_pages	Maximum number of pages to fetch. Defaults to Inf (all).
key	Optional API key. Defaults to the key set with zc_set_key() or the ZENTRACLOUD_API_KEY environment variable.

Value

A tibble with one row per device. Always includes device_id, name, organization_id, organization_name, device_type, and model. Requesting expand adds further columns (for example first_measurement and last_measurement as POSIXct times when "max_min_timestamp" is requested).

Examples

```
## Not run:
zc_set_key("my-secret-key")
zc_list_devices()
zc_list_devices(expand = "max_min_timestamp")

## End(Not run)
```

---

zc_pivot_wider *Reshape tidy readings into wide format*

Description

Turns the tidy "long" output of zc_get_readings() (one row per measurement) into a wide table with one row per timestamp and one column per measurement. This is the familiar "spreadsheet" layout for a single device.

Usage

```
zc_pivot_wider(readings, values_from = "value", names_sep = "_")
```

Arguments

readings	A data frame from zc_get_readings() (tidy long format).
values_from	Name of the column to spread into the wide cells. Defaults to "value".
names_sep	Separator used when a measurement name is combined with a port number. Defaults to "_".

Details

Note that the wide layout drops the per-reading `unit` and `error_code` columns (a single measurement column cannot carry them). Keep the tidy long form if you need units or quality flags alongside each value. When a device reports the same measurement on more than one port, the port number is appended to the column name to keep them distinct.

Value

A tibble with `device_id`, `datetime`, `timestamp`, and one column per measurement.

Examples

```
## Not run:
r <- zc_get_readings("z6-00930", start = Sys.Date() - 2)
zc_pivot_wider(r)

## End(Not run)
```

`zc_set_key` *Set your ZENTRA Cloud API key*

Description

Stores your ZENTRA Cloud v5 API key so that the other `zc_*()` functions can authenticate. By default the key is kept for the current R session only. Set `install = TRUE` to also save it to your user `.Renviron` file so it is available in every future session.

Usage

```
zc_set_key(key, install = FALSE, overwrite = FALSE)
```

Arguments

`key`	Your API key, as a string.
`install`	If `TRUE`, write the key to your user `.Renviron` as `ZENTRACLOUD_API_KEY` so it persists across sessions. Defaults to `FALSE`.
`overwrite`	If `TRUE`, replace an existing `ZENTRACLOUD_API_KEY` entry in `.Renviron` when `install = TRUE`. Defaults to `FALSE`.

Details

Get your key from ZENTRA Cloud: **User Account -> Integrations -> Show Token** (https://app.zentracloud.io/profile/integrations). Treat it like a password. Regenerating it in ZENTRA Cloud immediately invalidates the old key.

Value

Invisibly, the key.

Examples

```
## Not run:
# Keep the key for this session only:
zc_set_key("my-secret-key")

# Persist it across sessions:
zc_set_key("my-secret-key", install = TRUE)

## End(Not run)
```

`zc_store_access` *Read, write, and inspect a zentraR store*

Description

Low-level accessors for a store created by `zc_store_rds()`, `zc_store_csv()`, or `zc_store_memory()`. Most users interact with stores only through `zc_sync()`, but these are exported for direct access to the stored data.

Usage

```
zc_store_read(store, device_id = NULL)

zc_store_write(store, readings)

zc_store_last_time(store, device_id)
```

Arguments

`store`	A store object.
`device_id`	Optional device id. For `zc_store_read()`, `NULL` reads all devices. For `zc_store_last_time()` a single device id is required.
`readings`	A tidy readings data frame (as returned by `zc_get_readings()`).

Value

`zc_store_read()` returns a tidy readings tibble. `zc_store_write()` invisibly returns the store. `zc_store_last_time()` returns the latest stored timestamp (numeric epoch seconds) for the device, or `NA_real_` if the device has no stored data.

Examples

```
## Not run:
store <- zc_store_csv("data")
zc_store_read(store)
zc_store_last_time(store, "z6-00930")

## End(Not run)
```

zc_store_csv *Store readings as CSV files*

Description

Creates a store that persists readings as plain CSV files (one per device) inside `path`. The most accessible option for collaborators who open data in a spreadsheet or a tool other than R. Datetimes are written as ISO 8601 UTC strings and re-parsed on read.

Usage

```
zc_store_csv(path)
```

Arguments

`path`	Directory to hold the `.csv` files. Created if it does not exist.

Value

A store object for use with `zc_sync()` and `zc_store_read()`.

Examples

```
## Not run:
store <- zc_store_csv("data/zentra")
zc_sync("z6-00930", store = store, start = Sys.Date() - 30)

## End(Not run)
```

zc_store_memory *Store readings in memory (session only)*

Description

Creates a store that keeps readings in memory for the current R session. Data is not written to disk. Useful for experimentation or when another process (for example a database loader) will consume the returned data.

Usage

```
zc_store_memory()
```

Value

A store object for use with `zc_sync()` and `zc_store_read()`.

zc_store_rds *Store readings as RDS files*

Description

Creates a store that persists readings as native R `.rds` files (one per device) inside `path`. This is the most faithful option for RStudio project workflows: types (including `POSIXct` datetimes) round-trip exactly.

Usage

```
zc_store_rds(path)
```

Arguments

`path`	Directory to hold the `.rds` files. Created if it does not exist.

Value

A store object for use with `zc_sync()` and `zc_store_read()`.

Examples

```
## Not run:
store <- zc_store_rds("data/zentra")
zc_sync("z6-00930", store = store, start = Sys.Date() - 30)

## End(Not run)
```

zc_sync *Incrementally fetch and store readings*

Description

`zc_sync()` is the workhorse for keeping a local record up to date. For each device it looks at what you already have, fetches only the readings newer than that, and appends them to your chosen store - so you can run it on a schedule (weekly, daily, or on file open) and always pull just the new data.

Usage

```
zc_sync(
  device_id = NULL,
  store = NULL,
  start = NULL,
  end = NULL,
  units = "metric",
```

```
  direction = "ascending",
  max_pages = Inf,
  max_active = 1L,
  quiet = FALSE,
  progress = NULL,
  key = NULL
)
```

Arguments

device_id	Device id, or a character vector of device ids. NULL (default) discovers and syncs every device your key can access.
store	Where to persist readings: a store from zc_store_rds(), zc_store_csv(), or zc_store_memory(). Pass NULL (default) for "return only" - readings are fetched and returned but not stored (useful when you load them into your own database).
start	Optional start of the window for devices with no stored data yet (a Date, POSIXct, epoch seconds, or ISO 8601 string). Ignored for devices that already have stored readings.
end	Optional end of the window. Defaults to now (the API's latest).
units	"metric" (default) or "imperial".
direction	Fetch order: "ascending" (default) or "descending".
max_pages	Maximum pages (calendar-month windows) to fetch per device. Defaults to Inf.
max_active	Maximum number of devices to fetch concurrently. Defaults to 1 (sequential). Because the readings rate limit is enforced *per device*, values above 1 fetch several devices at once, which can be dramatically faster for multi-device syncs. Ignored when syncing a single device.
quiet	If TRUE, suppress progress messages. Defaults to FALSE.
progress	Show a live per-device progress indicator while fetching? NULL (default) shows one in interactive sessions (unless quiet); set TRUE/FALSE to force it, or use options(zentraR.progress = FALSE).
key	Optional API key. Defaults to the key set with zc_set_key() or the ZENTRACLOUD_API_KEY environment variable.

Details

The window for each device is chosen automatically:

- If the store already has readings for the device, fetching resumes just after the latest stored timestamp.
- Otherwise, if you pass start, that is used.
- Otherwise, the device's first-ever measurement is used (a full backfill).

Pages are written to the store as they arrive, so an interruption (network drop, rate limit, cancelled run) never loses the pages already committed - the next run simply resumes where it left off.

Value

If `store` is `NULL`, a tidy readings tibble (the combined new readings across devices). Otherwise, a summary tibble with one row per device: `device_id`, `rows_added`, `last_time` (latest stored POSIXct), `status` (`"ok"` or `"error"`), and `message`.

Examples

```
## Not run:
zc_set_key("my-secret-key")
store <- zc_store_csv("data/zentra")

# First run backfills from each device's first measurement (or `start`):
zc_sync("z6-00930", store = store, start = Sys.Date() - 30)

# Later runs fetch only what is new:
zc_sync("z6-00930", store = store)

# Sync every accessible device:
zc_sync(store = store)

# Return-only (no persistence), e.g. to load into your own database:
new_data <- zc_sync("z6-00930", start = Sys.Date() - 7)

## End(Not run)
```

Index